MODELING THE TEMPERATURE DEPENDANCE OF PKA AND INTEGRATION OF CHEMICAL PROCESS MODELS USING SPARC

by

SARAVANARAJ NALLAGOUNDEN AYYAMPALAYAM

(Under the Direction of Lionel A. Carreira)

ABSTRACT

SPARC (SPARC Performs Automated Reasoning in Chemistry) is a computer program developed to mimic an expert chemist in estimating chemical reactivity parameters and physical properties. SPARC strictly uses structural information of molecules to calculate these properties for a broad range of organic compounds. SPARC currently calculates the following chemical reactivity parameters for organic compounds: ionization pKa, molecular speciation in gaseous, aqueous and non-aqueous medium, hydration constants, hydrolysis constants, and tautomeric equilibrium constants.

A temperature dependence of pKa model was implemented to enhance the existing pKa model. The temperature dependence of pKa is modeled using Van't Hoff's equation. The Van't Hoff's coefficients for the different reaction centers are determined by plotting observed pKa versus the inverse of temperature. The slope of the line is the enthalpic coefficient and the intercept is the entropic coefficient. The temperature dependence was determined and modeled for most of SPARC's pKa reaction centers. However for reaction centers like amine acting as an acid, the temperature dependence was not modeled due to lack of sufficient experimental temperature dependence data.

In the natural environment there are many chemical processes acting on a molecule. In order to model the fate of compounds in nature, the chemical process models in SPARC were integrated. The integrated chemical process model includes hydration, tautomer network and molecular speciation models. This integration was challenging because of the enormous increase in the number of calculations that needed to be performed and the amount of data generated. Intelligent filters, based on the reliability of the calculations performed and identification of unproductive paths, were designed and implemented.

In the process of implementing cis-trans isomer and chirality information decoders for SMILES string handling in SPARC, the weakness and incompleteness of the SMILES encoding / decoding rules were discovered. A set of rules for encoding / decoding such information in SMILES was developed and implemented in SPARC. These new rules will be published and a request to collaboratively develop a complete standard for the encoding / decoding SMILES will be made.

INDEX WORDS: SPARC, Temperature Dependence, pKa, Tautomer, Chemical Fate, SMILES

MODELING THE TEMPERATURE DEPENDANCE OF PKA AND INTEGRATION OF CHEMICAL PROCESS MODELS USING SPARC

by

SARAVANARAJ NALLAGOUNDEN AYYAMPALAYAM

B.Tech, Madurai Kamarajar University, India, 1998

A Dissertation Submitted to the Graduate Faculty of The University of Georgia in Partial Fulfillment of the Requirements for the Degree

DOCTOR OF PHILOSOPHY

ATHENS, GEORGIA

2004

MODELING THE TEMPERATURE DEPENDANCE OF PKA AND INTEGRATION OF CHEMICAL PROCESS MODELS USING SPARC

by

SARAVANARAJ NALLAGOUNDEN AYYAMPALAYAM

Major Professor: Lionel A. Carreira

Committee: John Stickney
James Anderson
Allen D. King

Electronic Version Approved:

Maureen Grasso
Dean of the Graduate School
The University of Georgia
December 2004

DEDICATION

I dedicate this work to my wonderful family. My father who taught me to always aim high, believe in people and treat everybody justly. My mother who taught me kindness, patience and to have an open mind in life. My sister for always being there for me and supporting my decisions. Thanks Mom and Dad.

ACKNOWLEDGEMENTS

I would like to thank my advisor Butch Carreira for being a friend, philosopher and guide. I also thank Dr. John Stickney, Dr. Allen King and Dr. James Anderson for all their help and support. I would also like to thank Dr. Said Hilal for all the help, support and friendship he gave me. I thank my friends Tad Whiteside, Madhivanan Muthuvel and Dinesh Pillai for their support and friendship.

TABLE OF CONTENTS

Page

LIST OF TABLES

Page

LIST OF FIGURES

Page

Chapter 1

Introduction

1.1 Introduction

In these modern industrial times the use of chemicals has increased by leaps and bounds. In this context it has become very important to regulate and monitor the usage of these chemicals, and to study the impact of these chemicals on the environment. Most industrial countries have monitoring and control agencies which are responsible for regulating the use of these chemicals. As the number of potentially hazardous substances increases, maximizing the efficiency for assessment of these chemicals becomes important.

The major differences between behavior profiles of organic molecules in the environment are attributable to their physicochemical properties. Although considerable progress has been made in process elucidation and modeling for chemical and physical processes, determination of the values for the fundamental thermodynamic and physicochemical properties (i.e., rate/equilibrium constants, Henry's constant, solubility, etc.) have been achieved for only a small number of molecular structures. For most chemicals only fragmentary knowledge exists about those properties which determine their fate in the environment. The physical and chemical properties have actually been measured for only about one percent of the chemicals in the Office of Toxic Substances (OTS) inventory. These properties, in most instances, must be obtained from measurements or from the judgment of expert chemists. The cost of measuring the required properties of these chemicals is very high and time consuming. In any case, trained personnel

and adequate facilities are not available for measurement efforts involving the tens of thousands of chemicals introduced each year.

After 1990 the philosophy of pollution control changed. The philosophy dictated that the pollution should be controlled at the source. This meant that the manufactures and users of these chemicals have to switch over to less toxic and less polluting chemicals. New chemicals have to be developed to replace the existing ones and there will be a large development and screening process to find these replacements. In the process, many of the physical and chemical properties of these chemicals have to be determined costing a lot of money for the manufacturers. Property estimation can play a pivotal role in this process. It will make the process of screening faster and much cheaper.

For these reasons, during the past four decades, considerable effort has been made to develop empirical and non-empirical methods that will enable us to accurately and rapidly estimate physicochemical properties for organic molecules. Most of the property estimation methods developed during this period are based on empirical structure-activity relationship (SAR) models. The disadvantage of SAR models are that the relationships hold only for a limited family of chemical and are specific to a particular property query. They fail miserably when faced with novel compounds.

1.2 History of property prediction

Since the early 20th century it has been known that the property of a compound is determined by its chemical structure and the structure of the system with which it is interacting. It has been noted that a change in the substituent property (the electronegativity for example) in a compound brings about a change in the compound's property itself. It has been established by the work of Hammett that the effect a substituent has on a reaction, reaction1, will also be seen in

another reaction, reaction2, where the same substituent is present, except the magnitude may be different depending on the different substrates[1].

These observations led Hammett to propose a general quantitative relation between the nature of the substituent, S, and the reactivity of the side chain, C. This relation has become known as the Hammett equation, and is widely applied in this form

$$\log \frac{K_s}{K} = \sigma_s \rho_R \qquad 1.1$$

Where σ_s is a substituent constant which depends solely on the nature and the position of the substituent and is independent of the reaction. It is sum of the total electrical field (resonance and electrostatic) effects. ρ_R is the susceptibility of the reaction to electrical fields, determined by the reaction and its conditions (e.g., solvent, temperature, etc.) and is independent of substituent. K_s denotes the ionization equilibrium constants for the substituted compound, K denotes a statistical quantity corresponding approximately to the equilibrium constants of the unsubstituted compound. Hammett developed the equations specifically for benzoic acid and ρ_R for benzoic acid was set as 1. Others have modified the parameters and used the relationship for other compounds.

The expression on the left-hand side of equation 1 is proportional to the differences in the free energies of substituted and unsubstituted compounds as indicated in the next equation. For this reason equation 1.1 is often referred to as a "linear free-energy relationship" (LFER)[2]. Redefining equation 1.1 in terms of free energy change, it can be rewritten as equation 1.2.

$$\Delta G^0 - \Delta G_o^0 = \rho (\Delta G_o^{'} - \Delta G_o^{'0}) \qquad 1.2$$

Where ΔG^0 and ΔG_o^0 are the free energy changes for dissociation for a substituted and un-substituted compound. In the case of benzoic acid, $\Delta G^{'0}$ and $\Delta G_o^{'0}$ are the free energy of dissociation for substituted and un-substituted benzoic acid; $\Delta G^{'0} - \Delta G_o^{'0}$ gives the value of σ_s, the substituent constant.

Scientists were very successful in applying Hammett's approach to meta and para substituted compounds. But when it was applied to ortho substituted compounds the approach failed[2]. Ortho substituted compounds have more complicated interactions and effects. Interactions like hydrogen bonding and the orientation of dipoles have a very pronounced effect on the rates.

Later, using LFER, Taft and co-workers developed parameters based on the separation of substituent effects depending on the type of effects like electronegativity, field, polarizablity and resonance[2]. These parameters were successful in corelating a number of physicochemical properties involving solute/solvent interactions. LFER also led to the development of QSAR (Qualitative Structure Activity Relationships) an empirically determined correlation between structure and the compound's activity.

QSAR is widely used to estimate properties of compounds based on their structure. QSAR based predections perform adequately when the compound is analogous to the compounds that were used to determine the QSAR parameters. Most often when a new compound is seen by the system it does a poor job of estimating its property. Unfortunately, most of these calculational methods are based on the Hammett relation or LFER. They are purely empirical in nature and are usually only good with a particular class of compounds[1].

Dewar and his coworkers proposed a general treatment of substituent effects, based on the assumption that a substituent can affect a distant reaction center either by direct electrostatic

interactions across space (the direct field effect) or by polarization of intervening π electrons (mesomeric and π-inductive). Using a point charge model to calculate the field effect, and a simple Hückel Molecular Orbital (HMO) treatment of π polarization, a general treatment of substituent effects (direct field-indirect field effects) was developed[2]. This allowed for the calculation of many kinds of conjugated molecules in terms of just two parameters per substituent and one ρ constant for each type of reaction center.

The disadvantage of most of these estimation techniques was that they were based heavily on pattern matching and correlational inferencing in predictive strategy. They fell short of implementing actual scientific theory. SPARC has been developed to address this problem. It is an expert system that embeds theoretical knowledge as well as calculation algorithms. It implements scientific theory when calculating interactions.

1.3 SPARC

Chemical properties describe molecules in transition, that is, the conversion of a reactant molecule to a different state or structure. For a given chemical property, the transition of interest may involve electron redistribution within a single molecule or bimolecular union to form a transition state or distinct product. The behavior of chemicals depends on the differences in electronic properties of the initial state of the system and the final state of interest. For example, in chemical equilibrium, ionization chemical equilibrium constants depend on the energy differences between the protonated state and unprotonated state of the molecule. Reaction rates, on the other hand, may depend on the energies of a transition state relative to either reactants or products.

In every case, these differences are usually small compared to the overall energy of the molecule. *Ab initio* methods calculating absolute energies have a difficult time in predicting the

small energy differences between the chemical states of interest. Approaches based on LFER and QSAR methods have proven to offer good prediction within a limited number of molecules and within a particular class of molecules, but failed to predict either chemical or physical properties for a large number of molecules. In most cases the number of data does not exceed the number of parameters by as large an amount as would be desired.

Perturbation methods, however, can be used to compute differences in reactivity. These methods treat the final state as a perturbed initial state and the energy differences then are determined by quantifying the perturbation. These perturbation methods are ideally suited for expert system application due to their extreme flexibility and computational simplicity. The requisite conditions for applicability, as well as the selection of appropriate reference structures or reactions, can be easily built into the computation control portion of the expert system.

The SPARC program is a prototype computer system and it uses computational algorithms based on fundamental chemical structure theory to estimate chemical reactivity parameters and physical properties strictly from molecular structure for a broad range of organic molecules. SPARC stands for Sparc Performs Automated Reasoning in Chemistry.

The ultimate goal for SPARC is to develop mechanistic models to predict physicochemical properties for the universe of organic molecules strictly from molecular structure. Hence, SPARC mechanistic models are designed and parameterized so as to be portable, in principle, to any type of physical/chemical property or molecular structure.

1.4 Current SPARC Capabilities

SPARC as a result of its multiuse mechanistic toolbox design is quite capable of calculating many of the physical and chemical reactivity properties of molecules. Following is a list of SPARC capabilities:

I. Chemical Reaction Properties

A. Ionization pKa

1. Gas phase, aqueous and non-aqueous pKa
2. Temperature dependent pKa
3. Full molecular speciation as a function of pH
4. Full abundance calculation at given pH
5. Calculate iso-electric point for a molecule
6. Macroscopic and microscopic pKa

B. Tautomerization

1. Intelligent algorithm to determine all favorable tautomers
2. Calculate the tautomeric equilibrium constants for all possible tautomers

C. Hydrolysis

1. Acid, Base and Neutral catalyzed ester hydrolysis rate constants

D. Hydration

E. Integrated chemical process models

II. Physical Properties

A. Condensed phase properties

1. Solvent properties for mixed and user defined solvents
2. Solubility, concentration dependent activity, Henry's constant, distribution coefficients and GC/LC retention time (all as function of T or P)

B. Gas phase properties

1. Vapor pressure, boiling point, heat of vaporization, heats of formation and diffusion coefficient (all as function of T or P)

2. Molecular descriptors: Density, polarizablity, index of refractions and hydrogen bonding

1.5 SPARC Computational Procedure

SPARC does not do "first principles" computation. SPARC computes physical and chemical reactivity properties solely based on chemical structure. It does rigorous structural analysis of the compound based on the type of query from the user. SPARC tries to analyze the compound like an expert chemist. The intermolecular interactions are computed based on structure and are expressed by a set of molecular descriptors (density-based volume, molecular polarizability, molecular dipole and H-bonding parameters). For chemical reactivity the molecule is separated into the reaction center (the smallest molecule that can take part in a similar reaction) and the perturber (rest of the molecule). The perturbation of the reaction center with an intrinsic reactivity is calculated. The impact on reactivity of the reaction center due to the appended molecular structure (perturber) is quantified using mechanistic perturbation models[3]. For physical properties, intermolecular interactions are expressed as a summation over all the interaction forces between molecules (i.e, dispersion, induction, dipole and H-bonding). Each of these interaction forces are expressed in terms of a set of molecular descriptors (density based volume, molecular polarizability, molecular dipole, and H-bonding parameters). SPARC calculates molecular descriptors for the molecule and uses them in interaction models to calculate a physical property of interest.

A "toolbox" of mechanistic perturbation models has been developed that can be implemented where needed for a specific reactivity query. Resonance models were developed and calibrated on light absorption spectra whereas electrostatic models were developed on ionization equilibrium constants. Solvation models (i.e. dispersion, induction, H-bonding, dipole,

etc.) have been developed on physical properties (i.e. vapor pressure, solubilities, distribution coefficient, gas chromatographic retention times, etc.).

SPARC is a combination of LFER, SAR and Perturbed Molecular Orbital (PMO) methods. It uses LFER to calculate thermodynamic properties and PMO to calculate quantum effects such as resonance and polarization of π electrons. In reality, all chemical properties involve both quantum and thermodynamic contributions and require the use of both perturbation methods for prediction.

1.6 SPARC pKa calculation models

As described earlier, in the SPARC calculator molecular structures are broken into two functional units called the reaction center and the perturber. The reaction center is the smallest unit of the molecule that can undergo the ionization. The rest of the molecule forms the perturber. The reference pKa of the reaction center is a standard value for that reaction center and is inferred from measured values. The reference pKa of the reaction center $(pKa)_c$ is adjusted based on the differential perturbation effects of the substituents $\delta_p(pKa)_c$. The pKa can be calculated using equation 1.3.

$$pKa = (pKa)_c + \delta_p (pKa)_c \qquad 1.3$$

$\delta_p(pKa)c$ is the differential perturbation effect. The two differential states are the ionized and unionized reaction center. The perturbations are calculated as the difference in the interaction of the perturber with the ionized and unionized state of the reaction center. $\delta_p(pKa)c$ can be broken down into individual contributions as:

$$\delta_p(pK_a)_c = \delta_{ele}\, pK_a + \delta_{res}\, pK_a + \delta_{sol}\, pK_a + \ldots \qquad 1.4$$

where $\delta_{ele}\ pK_a$, $\delta_{res}\ pK_a$, $\delta_{sol}\ pK_a$ describe the differential electrostatic, resonance and solvation effects of the perturber with the initial and final states of the reaction center. Additional effects other than those listed include hydrogen bonding to the reaction center from the rest of the molecule.

The SPARC approach to pKa modeling is based on perturbation models. Any compound can be represented in the following format S—iRj—C, where S-iRj is the perturber structure attached to the reaction center, C. S is the substituent group that perturbs the reaction center[3]. The perturbation could be in the form of electric effects and/or resonance effects. R is the conductor network that conducts the perturbation. i and j denote the connection positions for S and C to the R network, respectively.

SPARC has a database of parameters describing the characteristics of all reaction centers and substituents. The term substituents are applied to all non-carbon atoms and aliphatic carbon atoms that are contiguous to the reaction center or a π-unit. There are some groups like ($-NO_2$, $-C{\equiv}N$, $-C{=}O$, $-CO_2H$, -NCO) which are treated as one whole unit. These groups are self contained structurally and electronically. Their properties, with respect to the rest of the structure of the molecule and their interactions, are also defined by SPARC after inferring them from observed data.

All perturbations in SPARC pKa models can be factored into three independent components.

1. Substituent strength, which describes the potential of a particular substituent to exert an effect.
2. Molecular network conduction, which describes the conduction properties of the molecular network, R, connecting S and C for a particular effect.

3. Reaction center susceptibility, which described the extent to which a reaction center is affected by a particular interaction.

Each of the above mentioned components are independent of each other. The potential of a substituent is gauged independent of the reaction center and the molecular network it is attached to. This is done by studying the effect of a particular substituent's effect on different reaction centers. Similarly, the parameters for the molecular network are determined by varying the substituents and reaction centers. Reaction center susceptibility is a differential value, which quantifies the differential response of the initial and final state for a particular effect.

1.6.1 Field Effects Model

The direct field effect for charged and dipolar substituents interacting with the reaction center is expressed as a multipole expansion, equation 1.5.

$$\delta(\Delta E)_{field} = \frac{\delta q_c q_s}{r_{cs}^{'} D_e} + \frac{\delta q_c \mu_s \cos\Theta_{cs}}{r_{cs}^{2} D_e} + \frac{\delta \mu_c q_s \cos\Theta_{cs}}{r_{cs}^{'2} D_e} + \frac{\delta \mu_c \mu_s \cos\Theta_{cs'} \cos\Theta_{cs}}{r_{cs}^{3} D_e} \quad 1.5$$

Where q_s is the charge on the substituent, approximated as a charge located at point s'; μ_s is the substituent dipole located at point s; δq_c ($\delta\mu_c$) is the charge (dipole moment) change at the reaction center accompanying the reaction, both are assumed to be located at point c; Θ_{cs} is the angle the dipole subtends to the reaction center; D_e is the effective dielectric constant for the medium through which the effect is propagated; and r_{cs} ($r_{cs}^{'}$) is the distance from the substituent dipole (charge) center to the reaction center.

1.6.2 Resonance Effects Model

Resonance involves the de-localization of charge between the reaction center and the π system, and between the substituent and the π system. The direction of charge movement is determined by the type of reaction center and substituent. Resonance in SPARC pKa model is a

differential effect. The difference in the delocalization in the initial and the final state gives the final perturbation to the pKa. If the initial state is more delocalized than the final then the pKa is increased, and visa versa.

SPARC uses PMO theory to determine the distribution of charge in a π system. To model this distribution, SPARC moves charge out of the reaction center after replacing it with a CH_2^- ion. All reaction center susceptibilities are expressed as a differential quantity compared to the CH_2^- ion. The resonance perturbation of the initial versus the final state for a reaction center is given by: equation 1.6.

$$\delta_{res}(pKa)_c = \rho_{res}(\Delta q)_c \qquad 1.6$$

Where $(\Delta q)_c$ is the fraction of NBMO charge lost from the surrogate reaction center (CH_2^-). ρ_{res} is the susceptibility of a given reaction center to resonance interactions. It quantifies the differential donor capability of the two states of the reaction center relative to the reference donor. Resonance strength (ability to receive or donate) is defined for all substituents; resonance susceptibility for all reaction centers is also defined in SPARC. Figure 1.1 shows a plot of SPARC's pKa performance. The test consists of about 4000 pKas. A R^2 value of 0.9876 is observed for the whole set.

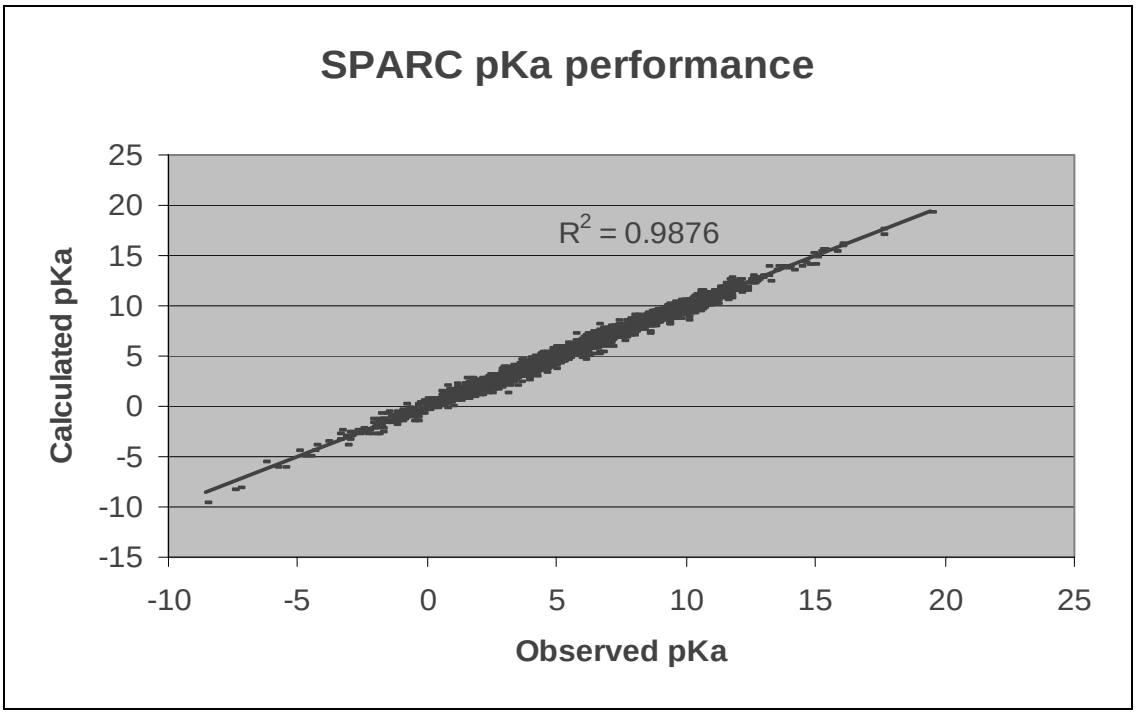


Figure 1.1 SPARC pKa performance plot of 4076 pKa calculations in water

1.7 Structural Input in SPARC

SPARC uses SMILES (Simplified Molecular Line Entry System) as the final form of input. SMILES was developed by David Weininger for EPA as a simple form of molecular structure input. It is a linear representation of a molecular structure using atomic symbols and special characters (=,-,#) to represent a molecular structure. SMILES is fully capable of specifying molecular structures. SMILES encoding grammar is very simple; hence it is easy for any chemist to encode it without using computer software.

SMILES has its own disadvantages, one of which is that there is no published and accepted standard which governs the rules for encoding it. Another problem is that SMILES does not provide absolute 3-D structural information i.e. it does not provide ways to specify 3-D coordinates. It does provide rules to encode structural isomer information (cis-trans, tetrahedral chirality).

An excellent tutorial on the rules of SMILES encoding is available at http://www.daylight.com/smiles/

Chapter 2

Temperature dependence of pKa

2.1 Introduction

The effect of temperature on the equilibrium of chemical systems is well known. SPARC pKa models are used to predict the ionization states of chemicals in the environment. The temperature in the earth's environment varies from about 0^{o} C to 30^{o} C, hence a model to calculate temperature dependent pKa is very important. In chemical processes, ionization plays a very important role. In coupled processes, a small change in the equilibrium rate constant could bring about an enormous change in the products. In chemical industries, reactions are performed in large scale using large quantities of energy to heat to cool the reaction chambers. To run such systems efficiently a chemical process engineer would want to study the effects of temperature on the reactions. For reactions involving ionizations, this SPARC model would be an invaluable tool in the hands of a chemical process engineer.

Temperature dependence models for pKa are very useful for drug designers. Modern drug technologies utilize the concept of controlled release. An active drug compound is released in a controlled manner by attaching long chained molecules with multiple ionization sites to control the overall pKa of the structure to maintain a thermodynamic equilibrium at a specific pH. By varying the pKa they can control the abundance of the active molecule in the blood stream. The disadvantage of this technique is the precipitation of the molecule out of the solution before injection into the body. There is a quite a difference between the storage temperature of the drug and the body temperature. The designers of these drugs have to make sure that their drugs can

stay in the proper form over this temperature range. Since pKa plays an important part in keeping the molecule soluble, it is important to know the temperature effect on the pKa for these molecules. In the preliminary phase of drug research SPARC models can be used to determine such relationships to aid in refining the wide selection of drugs.

2.2 Theory

For any thermodynamic equilibrium process the rate of change of free energy change with respect to temperature is represented by Van't Hoff's equation[2] (Eqn 2.1).

$$\frac{d \ln K}{d(1/T)} = -\frac{\Delta H}{R} \qquad 2.1$$

Integrating the above equation we get equation 2.2.

$$pKa = \frac{\Delta H}{2.303RT} + C \qquad 2.2$$

Rewriting as a linear function of 1/T w get it of the form

$$pKa = \frac{M}{T} + C$$

$$Here \quad M = \frac{\Delta H}{2.303R} is\ called\ the\ enthalpic\ coefficient \qquad 2.3$$

$$and\ C\ is\ known\ as\ the\ Entropic\ Contribution\ (Consant)$$

By plotting equation 2.3 as pKa vs 1/T we can get the slope (M) and the intercept (C) and develop the relationship for a compound.

SPARC's approach to modeling temperature dependence of pKa is to model the reaction center temperature dependence and the perturbation effects separately. In SPARC, for most reaction centers, the perturbations effects are assumed to be enthalpic and that the reaction center encapsulates the entropic contributions. However, for reaction centers which have a large susceptibility to perturbation effects, the perturbations are split into enthalpic and entropic

contributions. Equations 2.4 is a combination of the temperature effect model for reaction center and perturbations, to give the pKa of the compound.

$$pKa = A_c + \delta_s(\Delta E)_c + \frac{B_c + \delta_h(\Delta E)_c}{T}$$

Where A_c *and* B_c *are the entropic and enthalpic Van't Hoff coefficients for the reaction center* δ_s *and* δ_h *are entropic and enthalpic pertubation contributions* Eq 2.4

The Van't Hoff coefficients can be determined from temperature dependent pKa data for reaction center or inferred from a simple structure with minimal perturbations. The temperature dependence of enthalpic contribution of perturbations are calculated by an equation of the form y=mx + c assuming 'c' the entropic contribution to be zero. This assumption works well for almost all reaction centers. But in molecules where the reference pKa and the perturbations are high, for example the reaction center methyl acid has a reference pKa of 48 for the reference molecule methyl. In compounds where the methyl acid pKa are measurable like 2,4 pentadione (pKa 8.9) the perturbations have to about 39 pKa units. From temperature dependence data for methyl acids we can infer that all of the perturbation cannot be enthalpic. A factor to split the perturbations into enthalpic and entropic parts is introduced for methyl acid. For the rest of the reaction centers, the factor does not do much and perturbations are assumed to be purely enthalpic.

After splitting methyl acid perturbations into enthalpic and entropic parts to match the slope from the observed data, the enthalpic part was found to be 1% of the total perturbation. 99% of the perturbation cannot be entropic contributions. It is known that methyl acids are very slow in reaching equilibrium. Therefore the observed data could actually be measuring the abundances before the system reached equilibrium.

2.3 Procedure

To determine the Van't Hoff's coefficients, a large set of temperature dependent pKa data for each of the SPARC defined reaction centers were gathered and plotted Vs 1/T. The slope of the relation is calculated from the best-fit line. This slope and a first guess of B_c, A_c is used to train the SPARC model parameters to apply temperature correction on the reference value for that particular reaction center. The entropic contributions are assumed to be captured by the reaction center, A_c, and the perturbations are assumed to be all enthalpic (δ_H) with negligible entropic contribution (δ_S=0). This is the model we use to estimate the temperature correction for the perturbation contributions. Figure 2.1 and 2.2 show plot of pKa Vs 1/T for methyl amine and sulphuric acid.

A list of Van't Hoff's coefficients for a number of SPARC pKa reaction centers is given in Table 2.1. The amount of perturbation on the reaction center determines the temperature dependence of a compound. For example, the compound 2-nitro-aniline has a smaller slope compared to the slope of its reaction center (amine). This change in slope can go either way, perturbation can increase the slope or decrease the slope. Table 2.2 lists the 'B' Van't Hoff's coefficient of interesting molecules and their corresponding reaction centers.

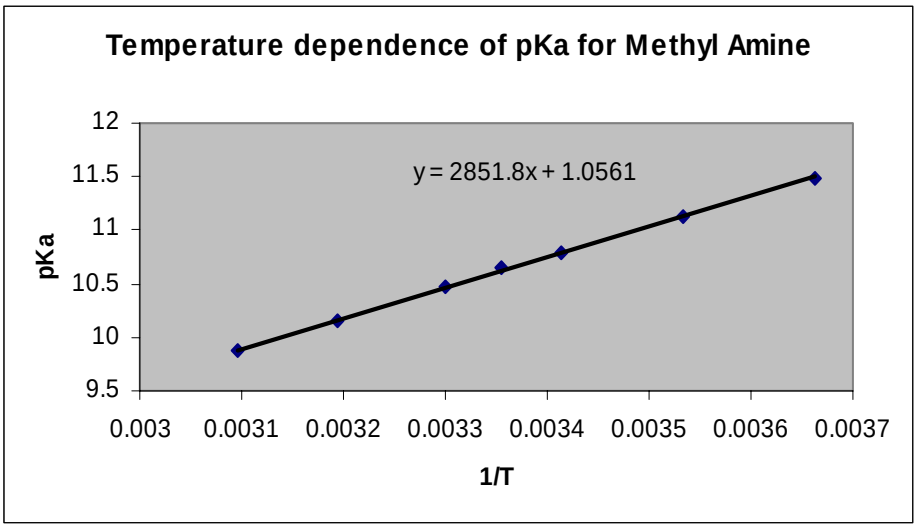


Figure 2.1 Plot of pKa Vs 1/T for methyl amine

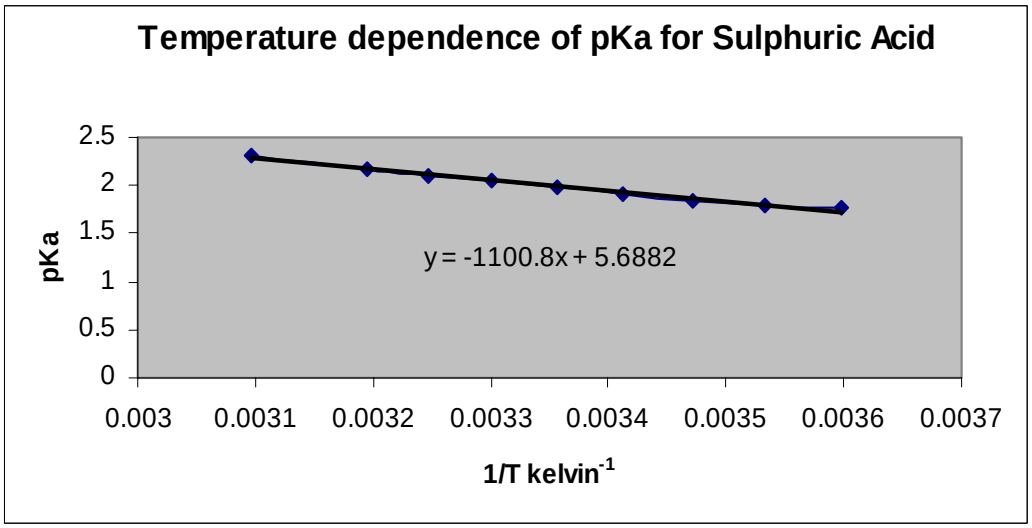


Figure 2.2 Plot of pKa Vs 1/T for sulphuric acid

Table 2.1 List of Van't Hoff's coefficients for various SPARC pKa reaction centers

Reaction Center	Bc	Ac
Amines (nr2)	2682.06	0.8345
Hydroxy (oh)	2775.91	4.987
Carboxlic Acid (co2h)	108.62	3.386
Thio (sh)	804.502	4.64
Carbon Acid (methyl_a)	783.452	45.55
Sulphonic (so3h)	-1729.26	5.708
Aromatic Nitrogen (n)	489.48	1.07

Table 2.2 Comparison of Van't Hoff's Co-efficient B of some select compounds and their reaction centers

Molecule	SMILES	B	Bc
2-nitro aniline	Nc1c(N(=O)(=O))cccc1	366.76	2682.06
2-chloro Phenol	c1(O)c(Cl)cccc1	900.02	2775.91
4-Amino Pyridine	n1ccc(N)cc1	2457.6	489.48

2.4 Discussion

Using the temperature dependent pKa data of selected molecules the Van't Hoff coefficients were determined for seven of SPARC's pKa reaction centers. This enables SPARC to calculate pKa's at any temperature for these reaction centers. There are other reaction centers in SPARC for which the temperature dependence could not be calculated, the reason being that sufficient temperature dependent pKa data are not available for those reaction centers. These include neth (C=N), nethR (neth as part of a ring), nr2_a (amine acting as acid) to list a few.

The factor to split the perturbations into enthalpic and entropic contributions was determined for methyl acting as an acid. The SPARC determined enthalpic factor for the methyl acid reaction center is 0.0108 and is set to 1 for the rest of the reaction centers. Figure 2.3 plots the performance of SPARC's temperature dependence model for pKa. The calculated values used for plotting have been corrected for the deviation of pKa values calculated by the pKa model. This correction projects only the performance of the temperature dependence model and takes the error from the base pKa models out of it.

2.5 Conclusion

A model for calculating temperature dependence of pKa was developed and implemented into SPARC. The model was not implemented for some reaction centers, nr2_a (amine acting as acid) for example, as there are no temperature dependent pKa data for such ionizations. Moreover these molecules are difficult to measure like methyl acids and are some times very unreliable.

Future work in this area would be to find measured enthalpy of ionizations for molecules which have not been addressed in this model. This enthalpy can be used to model the temperature dependence of pKa.

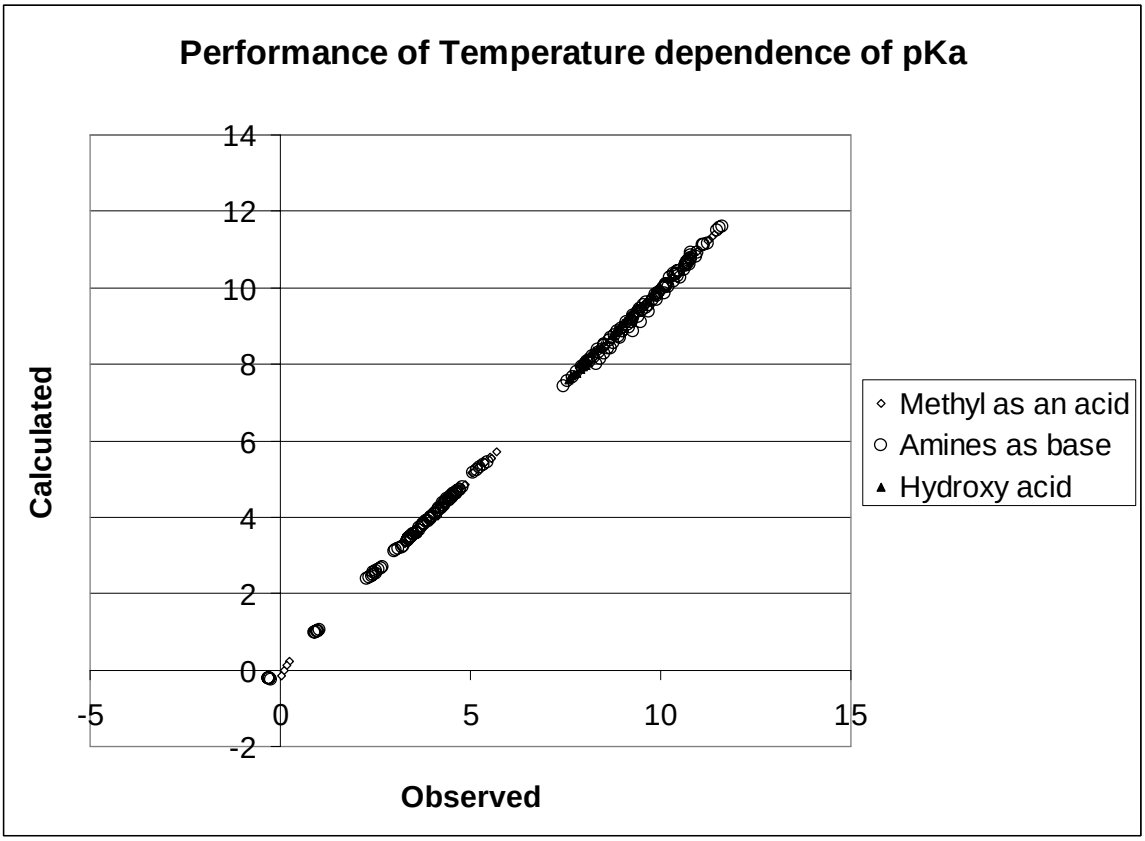


Figure 2.3 Plot of Observed vs Calculated pKa's at different temperature to show the performance of SPARC temperature models.

Chapter 3

SPARC Chemical Process Integration

3.1 Introduction

All chemical processes occurring in natural environments are a complex mixture of multiple processes involving the compound of interest. For predicting the fate of compounds in the environment, it is essential to take into account all the processes affecting the fate of the compound. For chemicals that speciate or exist in multiple forms (ions, zwitterions, tautomers, hydrates), observed chemical behavior may reflect integration over discrete chemical species or processes.

It is convenient to designate as 'macro', the observed equilibrium or kinetic constants, and designate as 'micro', a constant for a single chemical event (which may or may not be resolved experimentally). As an example, for ionization, a micro constant describes the loss or gain of a proton at a site whereas a macro constant may involve poly-protonic events relating to (1) loss or gain of protons from different sites on separate molecules that are integrated in the measurement, or (2) synchronous loss/gain of protons from different sites on the same molecule resulting in one unit change in total charge (e.g., gain of one and loss of two protons)[4].

SPARC currently has the molecular speciation, tautomer and hydration processes modeled. The speciation model determines the single and multiple ionization events occurring in a compound. It computes the abundance of each species at equilibrium as a function of pH, from which the macro constants, which are observed by experiments, can be determined. The tautomer model determines intelligently all possible tautomers of a compound and then computes the

tautomeric equilibrium constant for each tautomer. The tautomer model is a combination of the SPARC's chemical and physical process models.

The combination of these models will form the chemical fate prediction model in SPARC. Here I present the results of integration of the speciation, tautomer and hydration models. Further progress would be to incorporate the solid partition model, Henry's constant model, vapor pressure and solubility model to have a complete fate prediction model.

3.2 Molecular Speciation

Compounds with multiple ionization sites exist in various charged states in the environment. The charges on a molecule are dependent on the nature of the ionization sites and the environment in which the molecule exists. In a molecule with multiple ionization sites, the state of ionization of a site, affects the pKa of other ionizable sites. Hence it is important to know the state of all reaction sites in such molecules. In order to fully understand the molecular speciation in a given system, a complete calculation of all microscopic equilibrium constants has to be performed. The SPARC speciation model can intelligently determine all possible ionization species for a compound and determine the equilibrium constants for the individual reactions.

The reaction equilibrium constants for these individual reactions are known as micro constants. In molecules with a complex mix of such micro constants, simple experiments might not be able to measure or see these constants since there might be other dominating reactions. The name macro constants are assigned to measurable equilibrium constants. These macro constants may be a combination of one or more micro constants or micro steps. In the end, depending on the application, the user may want micro or macro constants.

For example, let's consider the compound N-Phenyl-Glycine (c1ccccc1NCC(=O)O), the experimentally observed pKas for the molecule are 2.1 and 4.4. Logically we would assume that

the pKa at 2.1 is due to the OH group in the carboxlic acid ionizing, and that the pKa at 4.4 is the basic pKa of the amine, both reactions taking place on the neutral molecule. But in reality there are four distinct ionization events taking place. They are, the neutral molecule ionizing at the carboxlic acid to form the negatively charged ion, the protonated amine carrying a positive charge, loosing the proton to form the neutral molecule and the same positively charged molecule ionizing at the carboxlic acid to form the zwitter ion (having zero total charge on the molecule) and lastly the protonated amine in the zwitter ion form loosing its proton to form the negatively charged molecule. Each of these individual reactions is represented by a micro constant. For N-Phenyl Glycine the individual reactions and their micro constants are listed in Table 3.1. The observed macro constants are a combination of the two or more micro constants, grouped by similar change in charge. SPARC also calculates the species fraction as a function of pH. Figure 3.1 plots the species fraction as a function of pH for all species possible from N-Phenyl-Glycine. The micro constants can be inferred from the intersection of the reactant and product species curves[5].

For any molecule with N ionization sites there will be 2^N species possible and N macro constants (Macro pKas). The number of individual pKa calculations needed to generate all the micro constants is $N*(2)^{N-1}$ calculations. SPARC calculates the macro pKa by summing the individual species curves of similar charge, and the intersection points of these summed curves determines the macro pKas[4].

Table 3.1 Table listing the micro and macro constants for N-Phenyl Glycine

Rxn	Reactant	Product	pKa	pKa Type
1	c1ccccc1NCC(=O)O	c1ccccc1NCC(=O)[O-1]	3.953	micro
2	c1ccccc1[N+1]CC(=O)[O-1]	c1ccccc1NCC(=O)[O-1]	4.158	micro
3	c1ccccc1[N+1]CC(=O)O	c1ccccc1NCC(=O)O	2.555	micro
4	c1ccccc1[N+1]CC(=O)O	c1ccccc1[N+1]CC(=O)[O-1]	2.364	micro
5	c1ccccc1[N+1]CC(=O)O	c1ccccc1NCC(=O)O c1ccccc1[N+1]CC(=O)[O-1]	2.051	macro
6	c1ccccc1NCC(=O)O c1ccccc1[N+1]CC(=O)[O-1]	c1ccccc1NCC(=O)[O-1]	4.381	macro

To obtain the fraction of each species as a function of pH from the calculated micro constants, the system needs to calculate the total abundance of all species at that pH. SPARC assumes that the neutral molecule has an abundance of 1 at equilibrium. Given all micro constants for all species and the pH, SPARC uses equation 3.1 to determine the total abundance of all species at that pH. Each individual term in the equation represents the abundance of a species and can be used to calculate it, and finally calculate the species fraction.

$$D = \frac{1}{0!} + \frac{\sum_i k_i [H]^{L_i}}{1!} + \frac{\sum_i \sum_{j \neq i} k_i k_{ij} [H]^{L_{ij}}}{2!} + \ldots + \frac{\sum_i \sum_{j \neq i} \ldots \sum_{k \neq ij\ldots} k_i k_{ij} \ldots k_{ij\ldots k} [H]^{L_{ij\ldots k}}}{N!} \quad \text{Eq 3.1}$$

D is the total abundance of all species in solution at a pH. K_{ij} is the equilibrium constant of species of state ij. L_{ij} is the total charge on the molecule at state ij. [H] hydrogen ion concentration is determined by the pH of the solution. The factorial is number of different pathways that lead to that particular state[4].

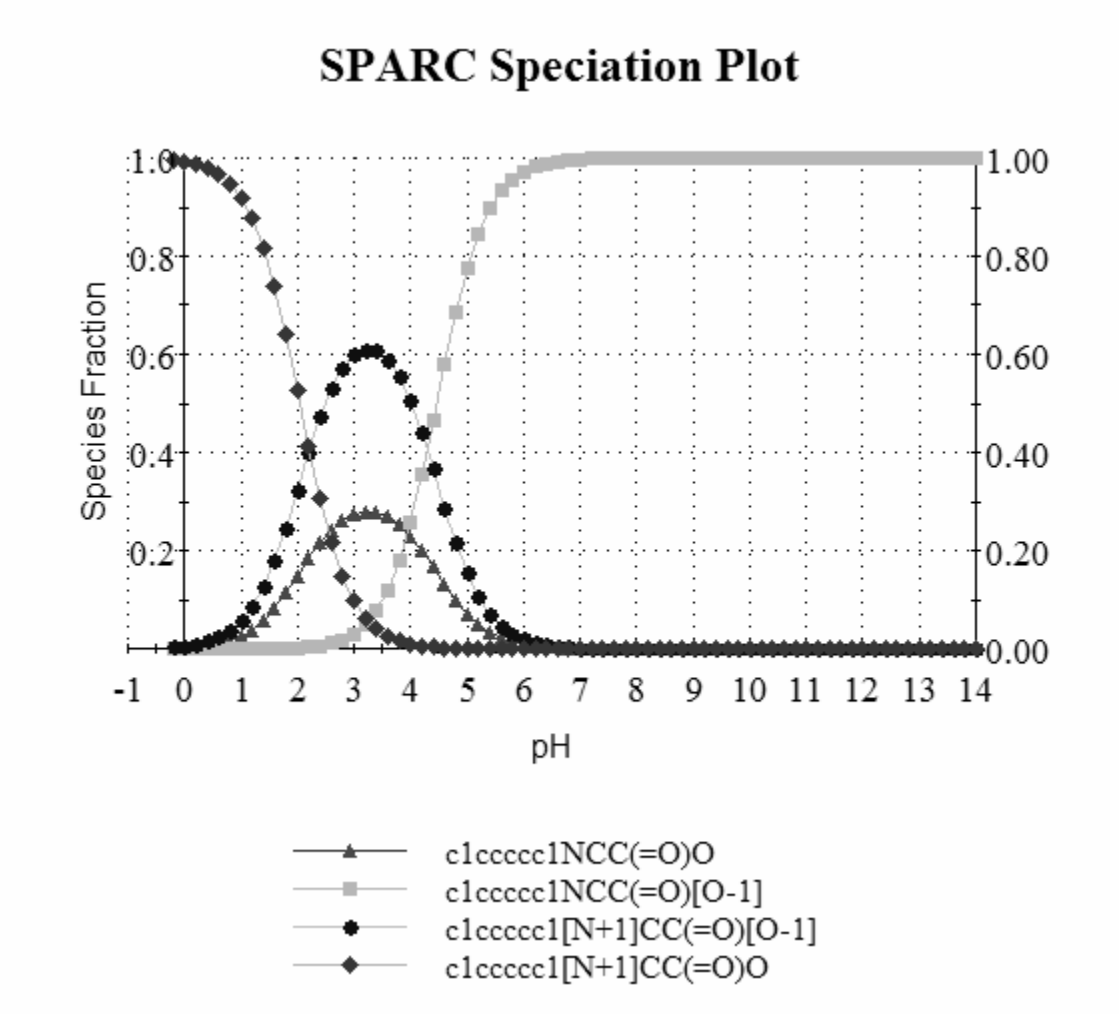


Figure 3.1 Plot of species fraction as a function of pH for N-Phenyl Glycine

3.3 Tautomer Equilibrium Constants

Tautomers are rapidly converting isomers of a structure. Tautomerism is a chemical process in which the double bonds in a molecule are rearranged with synchronized hydrogen atom shift to form an isomer. The most common form of tautomerism is the keto-enol tautomers. Tautomers play a very important role in terms of a compounds ability to exhibit a particular chemical reactivity[6]. It is important in the study of biological activity of chemicals and

researchers have been studying tautomerism in nucleic acids as for an explanation for self mutation in RNA.

Tautomerism is a two step process; first the molecule is attacked by an acid or base for the gain or loss of a proton respectively and a double bond rearrangement. Second step is the loss or the gain of the proton from the first step[6]. Hence it is catalyzed by acid or base. This makes the process dependent on the solvent properties.

To model the tautomeric equilibrium constant (K_T), the energy difference between the two isomeric structures, the ketone forms and the enol form has to be modeled. SPARC operates as a perturbation calculator to a reference structure, hence cannot directly model the tautomer equilibrium constant. Rather SPARC uses an indirect thermodynamic loop to calculate K_T.

3.3.1 SPARC Tautomer Models

SPARC does not calculate absolute energies of molecular structures. To calculate tautomeric equilibrium constants, SPARC uses its pKa and Henry's constant models to calculate K_T using a thermodynamic loop. The model is illustrated in figure 3.2, where the tautomeric equilibrium is calculated for 2-amino-1H-Indole.

The process flow in modeling tautomer equilibrium constants can be expressed as follows[7] :

1) Analyze the molecule and determine possible tautomers
2) Generate all the necessary structural information for representing all the tautomers of the starting molecule
3) Select a tautomer product and determine the atom to be ionized in the reactant to get the ionized structure. Apply similar logic to determine the atom to be ionized in the product
4) Calculate the pKa of both the ionizations

5) Calculate the energy required to move the ions into vacuum (Using Henry's Constant model)
6) Rearrange the reactant ion to form the product ion (zero energy)
7) Sum all the energies to determine the pK_T

In the process of modeling tautomer equilibrium constants using experimentally observed data, it was realized that the solvation energy difference between the two ionic forms is negligible and is within the noise range of the system. So under normal operation circumstances the differential Henry's energy term would be ignored. Tautomeric equilibrium models can also calculate the equlibrium constants in various arbitrary solvents. This is also done using the Henry's constant calculator to calculate the difference in the solvation of the reactant and the product.

One effect of integrating multiple process models is the dramatic increase in the number of calculations performed and the large quantities of data generated. To make sense of the data and to reduce the use of computational power, the output information needs to be filtered. In the grand scheme of things, a filter for the tautomer calculations was designed and implemented. The filter is designed to filter unproductive tautomeric pathways and to ignore un-reliable pK_T calculations. In order to decide un-reliable calculations a scoring system was developed and implemented. Every pKa calculation was given a reliability score based on the type of the reaction center. Currently the reliability for a carbon acid calculation is set as 0.5 and for the rest it is set to 1, the reason being the RMS value of the SPARC carbon acid model (1.1 pKa Units) is high compared to other reaction centers. The reliability of a tautomer equilibrium calculation is the product of the reliabilities of the two pKa calculations involved in its calculation. The filter,

filters the tautomeric equilibrium calculations with reliability less than 0.15 from being included when averaging the equilibrium constants for a tautomer states, this filter is only applied to tautomer states with more than one pathway.

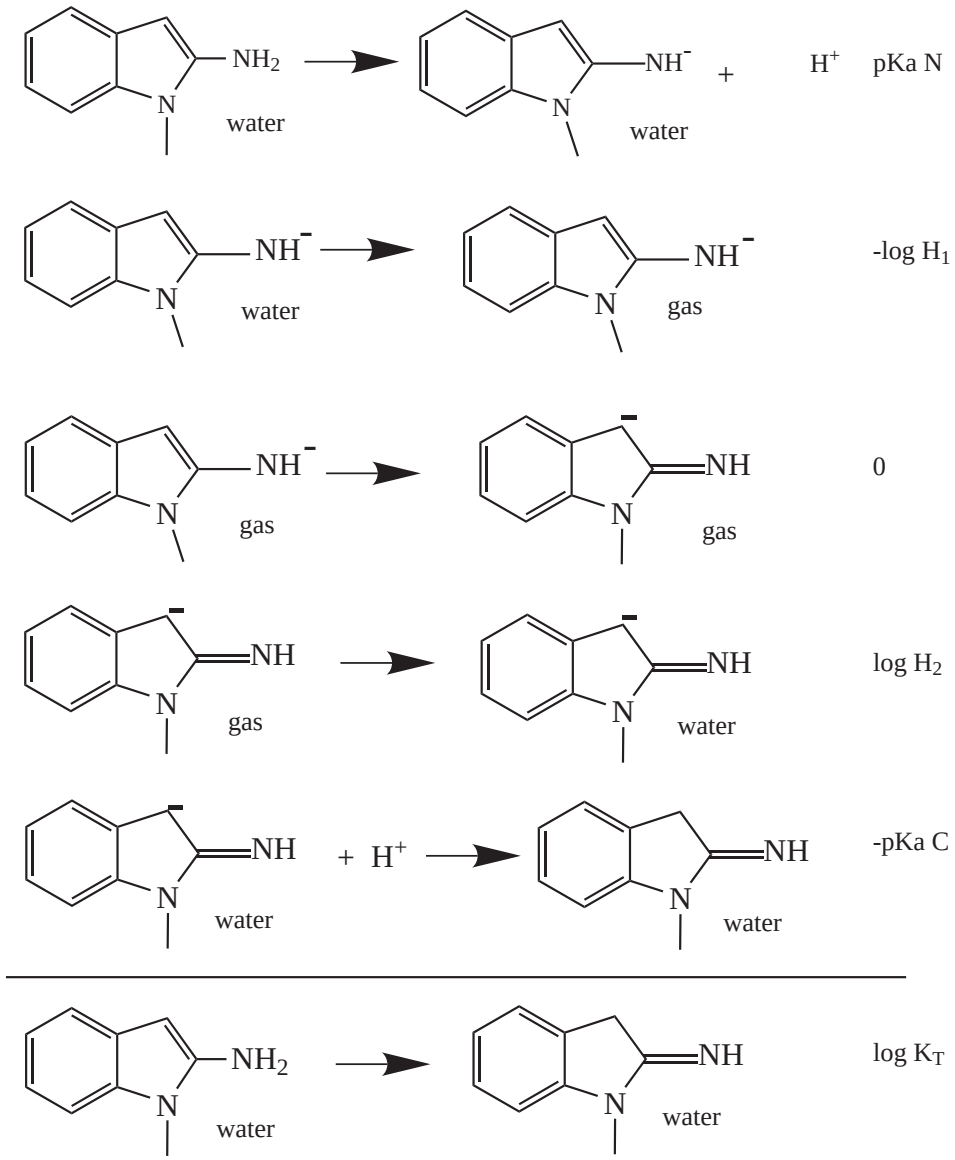


Figure 3.2 SPARC thermodynamic loop to calculate the tautomeric equilibrium constant

A list of SPARC calculated tautomeric equilibrium constants and the observed values are listed in Table 3.2[7,8]. A plot of the observed and calculated pK_T values is given in the form of Figure 3.3. Based on the very small number of published experimental investigations of the tautomeric properties of chemicals, the R^2 value for the plot at 0.895 is not bad. The effect of other processes involving the compounds of interest bring about a large margin of error in the observed data available in literature. When such coupled processes are involved simple experiments fail to measure these equilibrium constants accurately.

Table 3.2 A list of Observed and SPARC Calculated tautomeric equilibrium constants (pK_T)

Reactant	Product	Observed	Calculated
CC=O	C=CO	4.66	3.57839607
CCC=O	CC=CO	3.9	2.74232143
CCCC=O	CCC=CO	5.2	3.09963287
CC(C)C=O	CC(C)=CO	2.8	2.86012091
CC(=O)C	C=C(O)C	8.22	8.2798407
CCC(=O)CC	CC=C(O)CC	7.44	7.32422166
CC(C)C(=O)C(C)C	CC(C)=C(O)C(C)C	7.52	7.39685563
CC(=O)CC	C=C(O)CC	8.76	8.2823295
CC(=O)CC	CC(O)=CC	7.51	7.32239305
CC(=O)C(C)C	C=C(O)C(C)C	8.61	8.2823295
CC(=O)C(C)C	CC(O)=C(C)C	7.33	7.39469495
CC(=O)C(C)(C)C	C=C(O)C(C)(C)C	8.76	8.2823295
c1cc(OC)ccc1C(=O)C	c1cc(OC)ccc1C(O)=C	7.31	6.77728353
c1cc(C)ccc1C(=O)C	c1cc(C)ccc1C(O)=C	6.95	6.74714697
c1ccccc1C(=O)C	c1ccccc1C(O)=C	6.63	6.84466396
c1cc(Cl)ccc1C(=O)C	c1cc(Cl)ccc1C(O)=C	7.77	7.01818139
c1ccc(Cl)cc1C(=O)C	c1ccc(Cl)cc1C(O)=C	7.57	7.01322827
c1ccc(C(F)(F)F)cc1C(=O)C	c1ccc(C(F)(F)F)cc1C(O)=C	7.55	7.24412514
c1ccc(N(=O)=O)cc1C(=O)C	c1ccc(N(=O)=O)cc1C(O)=C	7.13	7.23358715
c1cc(N(=O)=O)ccc1C(=O)C	c1cc(N(=O)=O)ccc1C(O)=C	6.95	7.3205721
c1(C)cc(C)cc(C)c1C(=O)C	c1(C)cc(C)cc(C)c1C(O)=C	6.92	6.66554625
c1ccccc1C(=O)C(C)C	c1ccccc1C(O)=C(C)C	6.48	6.1408617

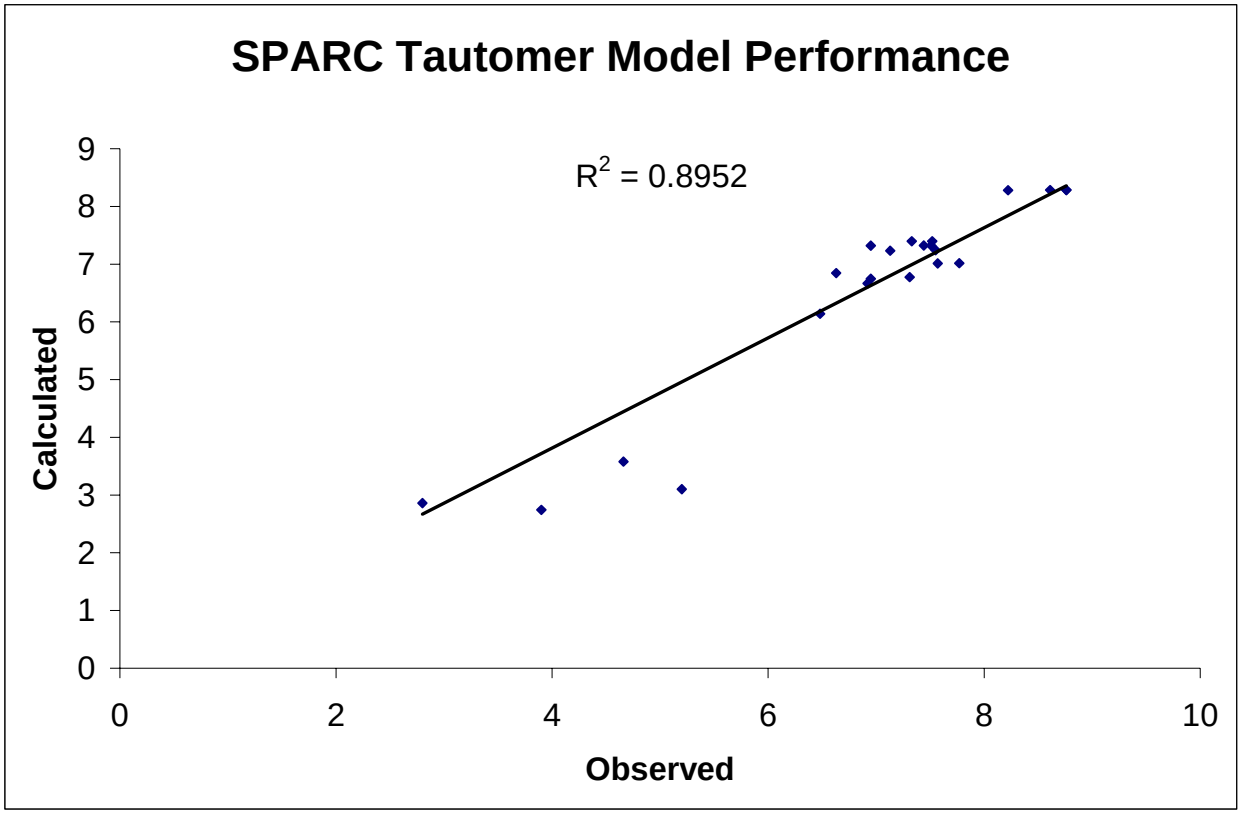


Figure 3.3 Plot of Observed vs SPARC Calculated tautomeric equilibrium values

3.4 Hydration

Hydration is the process of adding a water molecule across a pi-electron functional group. The two structural units where this is known to occur are the carbonyl and imine functional groups. Hydration follows the Markovnikov's rule of addition. In each case, a hydroxyl group attaches to the base carbon and a hydrogen atom to the heteroatom[6].

As described previously, in the SPARC modeling approach these functional groups will be reaction centers and any molecular structure(s) appended thereto designated perturber structure.

$$P\text{-}C_i \rightarrow P\text{-}C_f$$

In the case of hydration, differential solvation of the two species will play a major role. In this case we will start with the following thermodynamic cycles to model the reaction.

$P\text{-}C{=}O\ (g) \rightarrow P\text{-}C(OH)_2\ (g)$ $\quad \Delta G_{hydration}(g)$

$P\text{-}C(OH)_2\ (g) \rightarrow P\text{-}C(OH)_2\ (l)$ $\quad \Delta G_{transfer}(O)$

$P\text{-}C{=}O\ (l) \rightarrow P\text{-}C{=}O\ (g)$ $\quad -\Delta G_{transfer}(=O)$

$P\text{-}C{=}O\ (l) \rightarrow P\text{-}C(OH)_2\ (l)$ $\quad \Delta G_{hydration}(l)$

The top reaction will be modeled using the usual SPARC perturbation approach.

$$\Delta G_{hydration} = (\Delta G_{hydration})_c + \delta_p(\Delta G_{hydration})_c \qquad \text{Eq. 3.2}$$

where the reaction center ΔG (in this case formaldehyde) is perturbed by appended molecular structure. This perturbation is further factored into mechanistic components such as:

$$\delta_p(\Delta G_{hydration})_c = \delta_{ele}\Delta G_{hydration} + \delta_{res}\Delta G_{hydration} + \delta_{steric}\Delta G_{hydration} + \ldots \qquad \text{Eq. 3.3}$$

From structure theory of organic chemistry, it is known that nucleophilic addition reactions across pi bonds are sensitive to inductive and steric effects from atoms contiguous to the pi group. Also, it is known that functional groups containing non-bonded electrons (-oh, -or, $-nr_2$) attached to the base carbon will prohibit hydration (via induction and resonance). This model can confirm the failure of esters, amides, ureas, and carboxylic acids to hydrate and can project other structures to be readily hydrated. The biggest perturbations are the direct field effect (increase), sigma induction (decrease), resonance (decrease) and steric (decrease)[9]. SPARC hydration models now calculate the hydration of ketones, aldehydes and quinazolines. A performance plot of observed vs calculated for SPARC hydration models is shown in figure 3.4.

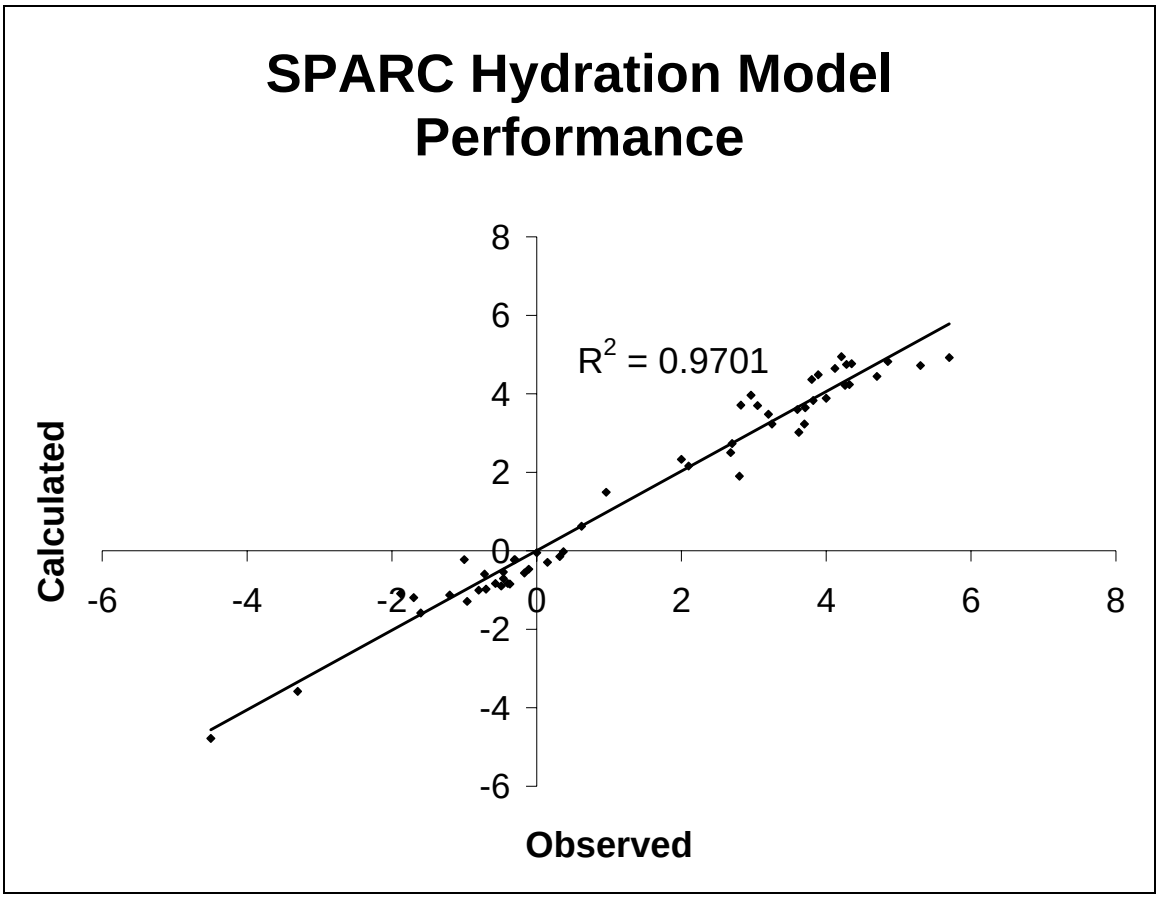


Figure 3.4 Plot of Observed vs SPARC Calculated hydration constants

3.5 SPARC Process Integration

The above sections explained the implementation of the various chemical process models modeled in SPARC. One of the main problems associated with the integration of these individual processes is the number of computations needed to determine all the various combinations of pathways. Another by-product of all these calculations is the amount of data that is generated by the calculators. To bring some sort of meaning and control to the whole system a set of filters were implemented.

The filters are based on the cumulative reliability of the calculations along the path and the cumulative equilibrium constant at that point. The first step in reducing the amount of information generated is to reduce the number of unwanted / unproductive reaction pathways from the network. The logic behind deciding unproductive / unwanted reaction pathways is the cumulative equilibrium constant, it provides a way of deciding whether a particular node in the network would be present in sufficient quantity to warrant a progression along that direction. This decision cannot be taken based on the cumulative equilibrium constant after just one step. SPARC only applies this rule after two or more steps have been taken from the initial state.

The reliability of calculations in a pathway is very important, as, the equilibrium constants are averaged for each tautomer form. As discussed before, a reliability value is assigned to all tautomeric equilibrium calculation. In the case of a pK_T calculation with low reliability the pathway is not included in the equilibrium constant averaging for that node.

The first step in building an integrated process model is to use the hydration model to determine the hydration constant and the hydrated product. The next step would be to find all possible tautomers of the starting compound and its hydrated product, if any, and determine the combined equilibrium constant for all the different products. After all the tautomeric forms and equilibrium constants of the initial compound and its hydrated product are determined, the pKa ionization / speciation model is used to determine their speciation products and the combined equilibrium constants. The speciation model calculates the species fractions as a function of pH, determines the macro constants and the coupled micro constants for all the species.

3.5.1 Building Tautomer Network

The SPARC tautomer model is used to recursively determine all possible tautomers from the starting compound and tautomers of the tautomers themselves, thereby creating a tautomer

network map with the different tautomer forms forming nodes and the equilibrium constants between these nodes are computed. A doubly recursive algorithm is used to perform these calculations and the progression down the network is of the depth first search type. This algorithm enables SPARC to identify all tautomer forms and their equilibrium constants irrespective of the starting compound. Once again filters are applied based on reliabilities and cumulative equilibrium constants for nodes more than two steps away from the start. The current threshold reliability value for tautomer equilibrium constant for inclusion into averaging is 0.15. The logic based filter evaluates the aggregate K_T for reaction pathways two or more steps long. Filtering decisions are based on a threshold value, anything below threshold is filtered as an unproductive pathway. This threshold is set at 2E-06 for the tautomer network model. This helps in the reduction of the number of calculations by removing unproductive reaction pathways.

Figure 3.5 is a flow diagram illustrating the logic behind determining tautomer network maps. This model has been tested using complex compounds capable of producing multiple tautomers. The accuracy of the calculated tautomer equilibrium would go down as the network grows, but the numbers give a very good idea about the feasibility of that tautomer forms existing. Sample output of the tautomer network model for acetyl acetone is provided in the form of table 3.3 and is illustrated as a mapped network in figure 3.6. The table lists the different tautomeric forms and their fraction at equilibrium assuming, acetyl acetone ('CC(=O)CC(=O)C') the starting point has a concentration of 1. In the column for individual equilibrium constants, the equilibrium constant for each tautomeric step taken to reach the current node is listed as a dashed pair with its reliability value. Let's consider the tautomer no 4 (CC(O)=CC(=O)C), the list of individual equilibriums for this form are

1) [0.1348436469935751-0.5,1-1]

2) [11303823096407.127-0.25,3.1711564854704756E-08-0.5,1-1]

The 1st path consists of one step with a K_T value of 0.134 and a reliability of 0.5, 1-1 is the entry for the initial starting point and does not count as a step. The 2nd path has two steps with K_T for the first step 3.17E-08 with a reliability of 0.5 and 1.13E+13 with reliability of 0.25 for the second step. The combined reliability for the 2nd path is the product of the reliabilities of the individual steps and is listed as path reliability. The total path reliability for the second path is 0.125. The observed tautomer equilibrium constant for tautomer 4 is 0.16 An example for filtering based on aggregated K_T is the termination of reaction progression from compound IV (aggregate K_T = 1.13E-16 very lower than threshold of 2E-06) in the example illustrated in figure 3.6.

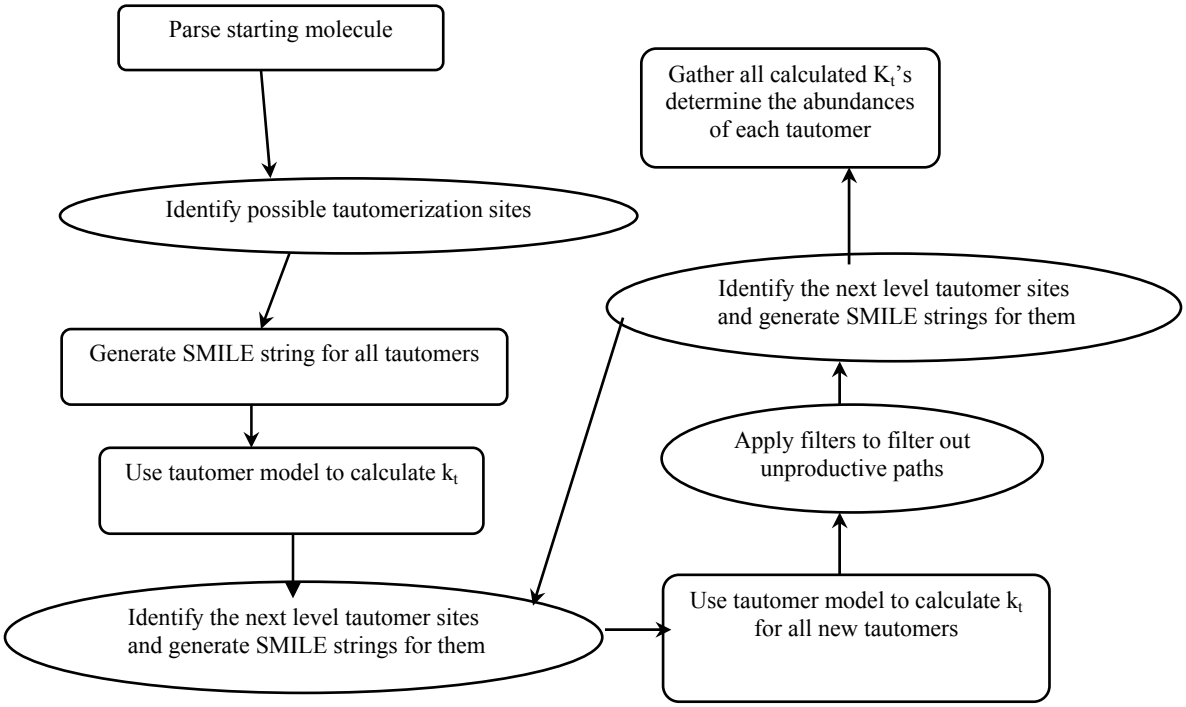


Figure 3.5 Flow Diagram for Determining Tautomer Networks

Table 3.3 Output from SPARC Tautomer Network Model

No.	Compound	Species Fraction	Individual Equlibrium Constants and Reliabilities	Path Reliability
1	C=C(O)CC(O)=C	1.1391 E-16	[6.091130108395312E-08-0.5, 3.1711 E-08-0.5,1-1]	0.25
2	C=C(O)C=C(O)C	1.9505 E-08	[0.001272564344873305-0.5, 3.1711 E-08-0.5,1-1]	0.25
			[6.107683459346353E-08-0.5, 0.13484364-0.5,1-1]	0.25
3	C=C(O)CC(=O)C	3.1711 E-08	[3.1711 E-08-0.5,1-1]	0.5
4	CC(O)=CC(=O)C	0.1348436	[1.13E+13-0.25, 3.1711 E-08-0.5,1-1]	0.125
			[0.1348436-0.5,1-1]	0.5
5	CC(=O)CC(=O)C	1	[[1-1]	1

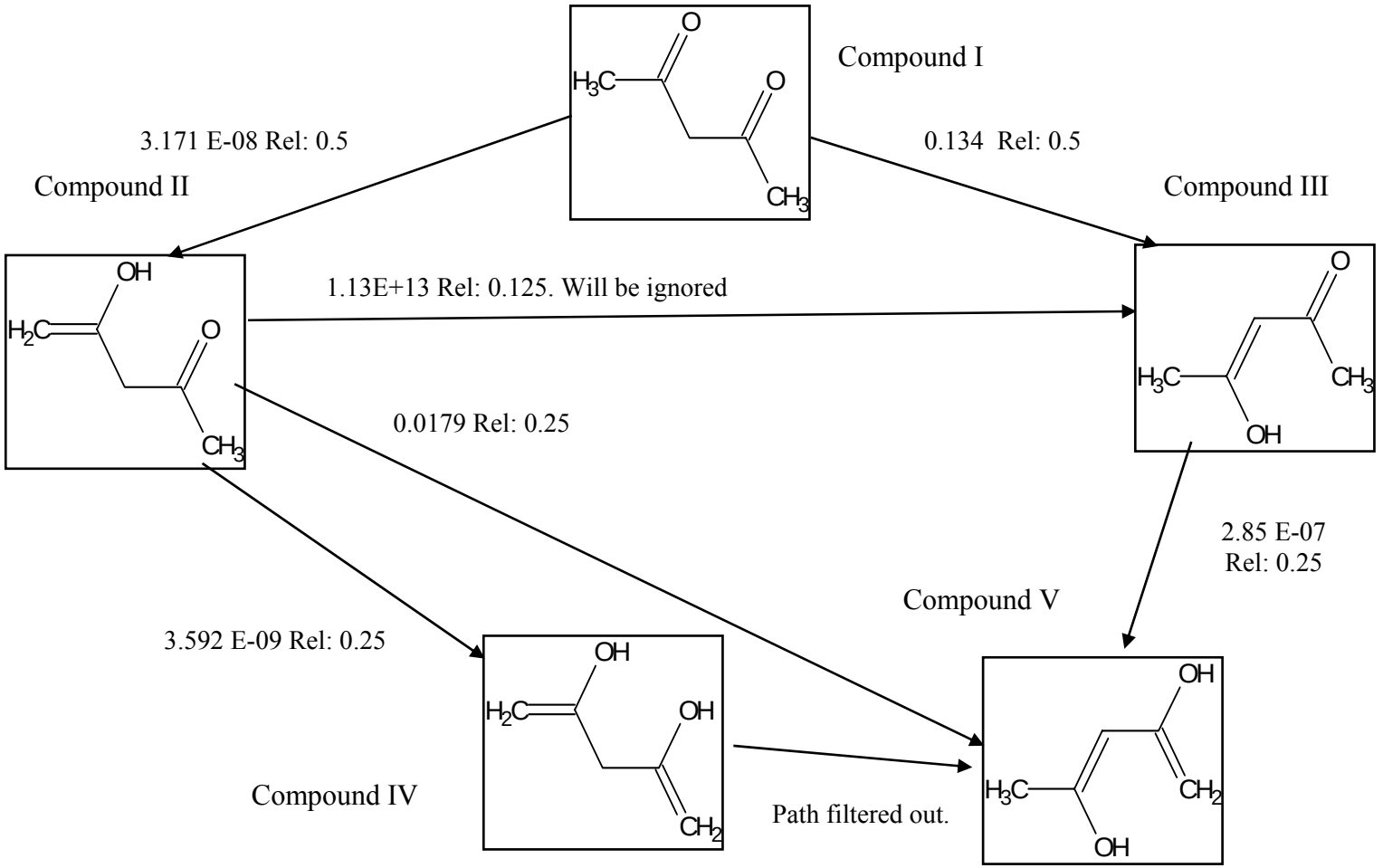


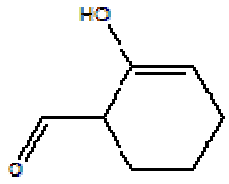
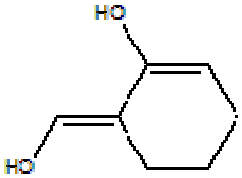
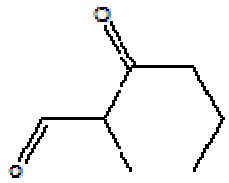
Figure 3.6 SPARC generated tautomer map for acetyl acetone. The equilibrium constants are indicated next to the arrows with the reliability of the calculation

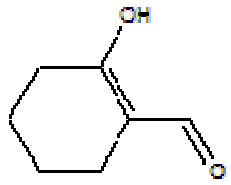
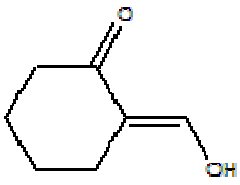
3.5.2 Integration of Hydration and Tautomerization

The hydration model is used to generate all the hydrated forms of the starting molecule and determine their hydration constants. These molecules and their respective equilibrium constants are combined and fed into the tautomer network model to determine the possible tautomer and their integrated constants.

Let us use the model to determine the reaction pathways of 2-Aceto-Cyclohexanone. To illustrate the effect of coupling hydration to Tautomerization, two different calculations, one without hydration and one with hydration, is reported. Table 3.4 lists the output of the model without hydration. Table 3.5 lists the output of the model with hydration turned on. Both calculations are performed in water as solvent.

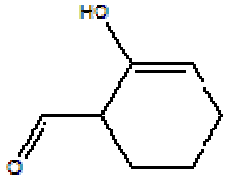
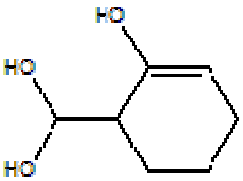
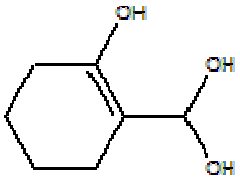
Table 3.4 Results of integrated tautomer network model without hydration

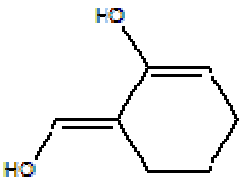
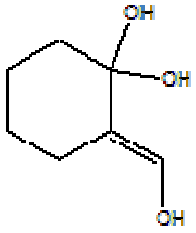
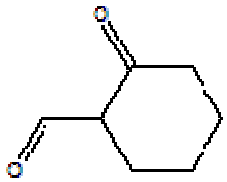
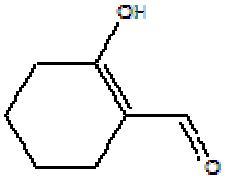
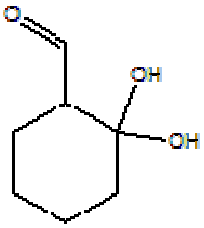
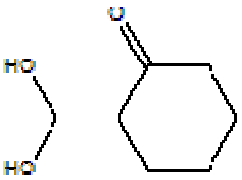
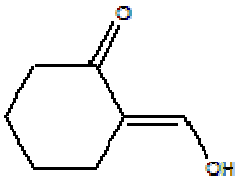
No	Tautomer	Relative Abundance	Molecule
1	C1(O)=CCCCC1C=O	7.35E-07	
2	C1(O)=CCCCC1=CO	2.00E-03	
3	C1(CCCCC1C=O)=O	1.00E+00	

4	C1(O)=C(C=O)CCCC1	4.25E+00	
5	C1(C(=CO)CCCC1)=O	4.66E+01	

From the results, the conjugated enol-keto forms are found to be the most stable tautomeric form in water. This is justified as the double bond conjugation stabilizes the molecules better than the di-carbonyl form.

Table 3.5 Results of the integrated tautomer network model with hydration

No	Tautomer	Relative Abundance	Molecule
1	C1(O)=CCCCC1C=O	7.35E-07	
2	C1(O)=CCCCC1C(O)O	5.85E-06	
3	C1(O)=C(C(O)O)CCCC1	9.57E-05	

4	C1(O)=CCCCC1=CO	2.00E-03	
5	C1(O)(C(=CO)CCCC1)O	3.53E-03	
6	C1(CCCCC1C=O)=O	1.00E+00	
7	C1(O)=C(C=O)CCCC1	4.25E+00	
8	C1(O)(CCCCC1C=O)O	9.94E+00	
9	C1(CCCCC1C(O)O)=O	2.44E+01	
10	C1(C(=CO)CCCC1)=O	4.66E+01	

Compared to the results without hydration we see more species present in significant quantities. The relative stabilities of two hydrated forms 8, 9 are explained by the ease of hydration of the aldehyde versus that of the ketone. These species are present in significant quantities to affect chemical behavior.

3.5.3 Integration of Speciation, Hydration and Tautomer Network

Using the hydration coupled tautomer network model the equilibrium constants for the neutral species has been developed. The next step is to determine all the different ionization process as applied to these molecules and determine their pKas.

A full speciation calculation is performed on the different species from the tautomer network model. The respective equilibrium constants of the tautomer forms are integrated used to determine the final species fraction as a function of pH. The molecular speciation model is the same model used for pKa speciation; hence it provides a wealth of information to the user. It calculates the macro constant and micro constants for all the involved chemical process and determines species fraction as a function of pH.

Let us analyze the results of the fully integrated chemical process model using the same compound. The plot of the species fraction as a function pH if shown in figure 3.7. From the plot we see that below pH 9 the dominant species are the exo-enol form (species 3) and the di-hydro form (species 2) with a little bit of the other di-hydro form (species 1). At about pH 9 these three species disappear and the ionized form of the di-carbonyl form, and the two keto-enol forms dominate (species 6,7,8). The species number is the order in which they are listed in the figure.

As the pH increases the concentration of the di-hydro form is reduced, the force that drives this reduction is the ionization of the tautomer of the un-hydrated form which in turn drives equilibrium of hydration in favor of the reactant, thus decreasing the concentration of the

hydrated forms. This compound is a very good test for coupled reaction models and is performing very well. A list of the reactions together with the micro constants is listed below. A total of 59 micro constants and one macro constant were determined by the speciation calculator.

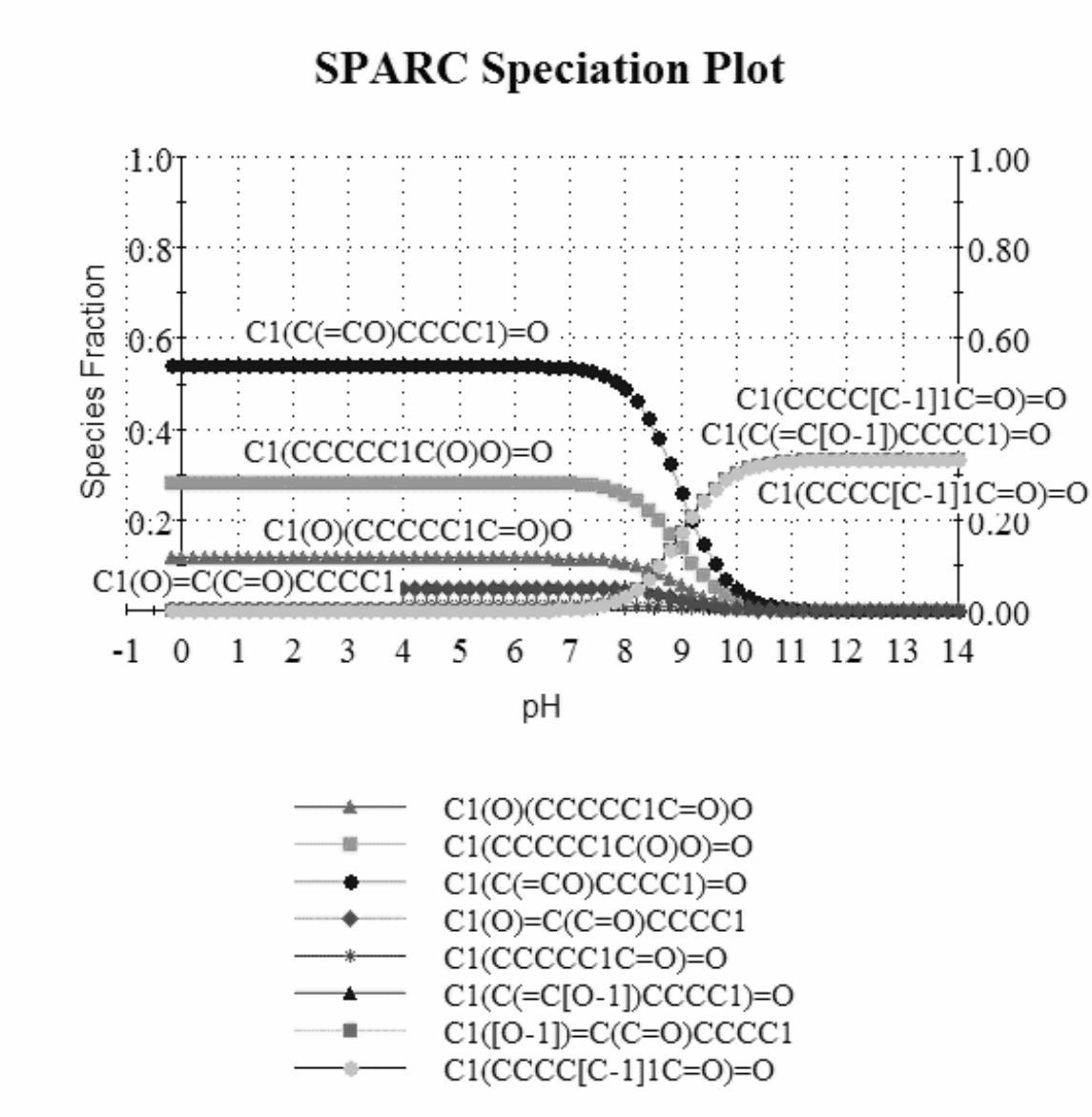


Figure 3.7 Plot of species fraction as a function of pH for 2-Aceto-Cyclohexanone

3.5.4 Thermodynamic Loops

A solution to fix the low reliability problem in the tautomeric equilibrium calculations is to use thermodynamic loops to infer the actual equilibrium constants for those calculations with low reliability. When there is more than one path to a species / tautomer form the change in free energy being independent of path taken should be the same for both pathways, such loops are known as thermodynamic loops. In order to infer these constants and use them to train SPARC parameters the rest of the calculations involved in the loop has to be reliable.

For example, in the calculation of the tautomer network for acetyl acetone we saw that certain pathways were rejected due to poor reliability of 0.125 (i.e. path with K_T [11303823096407.127-0.25, 3.1711564854704756E-08-0.5, 1-1]). This is due to the K_T for the tautomerization of compound II 'C=C(O)CC(=O)C' to compound III 'CC(O)=CC(=O)C' being unreliable. Using thermodynamic loops and assuming that the first step is reliable we can determine that the K_T for the second step should be 5E+06 rather than 1.13E+13. This value will be used to infer the involved, un-observer pKas taking part in the calculation for that tautomer equilibrium constant.

3.5.5 Microscopic Reactions

Listed below is a list of microscopic reactions and their equilibrium constants:

Micro reactions and constants:
C1(CCCCC1C=O)=O -> C1(O)(C(=CO)CCCC1)O
Kt is 0.0035310905141420637

C1(CCCCC1C=O)=O -> C1(O)(CCCCC1C=O)O
Kt is 9.941725439243926

C1(CCCCC1C=O)=O -> C1(O)=C(C(O)O)CCCC1
Kt is 9.569754217721239E-05

C1(CCCCC1C=O)=O -> C1(CCCCC1C(O)O)=O
Kt is 24.37683088627839

C1(CCCCC1C=O)=O -> C1(O)=CCCCC1=CO
Kt is 0.0020005816586784316

C1(CCCCC1C=O)=O -> C1(C(=CO)CCCC1)=O
Kt is 46.62888703024625

C1(CCCCC1C=O)=O -> C1(O)=C(C=O)CCCC1
Kt is 4.247989918782148

C1(CCCCC1C=O)=O -> C1(CCCCC1C=O)=O
Kt is 1.0

C1(O)(C(=CO)CCCC1)O -> C1(O)(C(=C[O-1])CCCC1)O
pKa is 11.290990995289015

C1(O)(C(=CO)CCCC1)[O-1] -> C1(O)(C(=C[O-1])CCCC1)[O-1]
pKa is 13.02423611669241

C1(O)=C(C(O)O)CCCC1 -> C1([O-1])=C(C(O)O)CCCC1
pKa is 12.223746259259778

C1(O)=C(C([O-1])O)CCCC1 -> C1([O-1])=C(C([O-1])O)CCCC1
pKa is 13.974357570407976

C1(O)=C(C(O)[O-1])CCCC1 -> C1([O-1])=C(C(O)[O-1])CCCC1
pKa is 13.974357570407976

C1(O)=CCCCC1C(O)O -> C1([O-1])=CCCCC1C(O)O
pKa is 10.92533207205689

C1(O)=CCCCC1C([O-1])O -> C1([O-1])=CCCCC1C([O-1])O
pKa is 13.055136185567525

C1(O)=CCCCC1C([O-1])[O-1] -> C1([O-1])=CCCCC1C([O-1])[O-1]
pKa is 14.73343604534849

C1(O)=CCCCC1C(O)[O-1] -> C1([O-1])=CCCCC1C(O)[O-1]
pKa is 13.055136185567525

C1(O)=CCCCC1=CO -> C1([O-1])=CCCCC1=CO
pKa is 11.402019238648172

C1([O-1])=CCCCC1=CO -> C1([O-1])=CCCCC1=C[O-1]
pKa is 12.934205115884826

C1(O)=CCCCC1=CO -> C1(O)=CCCCC1=C[O-1]
pKa is 11.164266323173813

C1(O)=CCCCC1=C[O-1] -> C1([O-1])=CCCCC1=C[O-1]
pKa is 12.948329205548161

C1(C(=CO)CCCC1)=O -> C1(C(=C[O-1])CCCC1)=O
pKa is 9.179621017995222

C1(O)=C(C=O)CCCC1 -> C1([O-1])=C(C=O)CCCC1
pKa is 8.1391494458693

C1(O)=CCCCC1C=O -> C1(O)=CCCC[C-1]1C=O
pKa is 11.807832985774404

C1(O)=CCCC[C-1]1C=O -> C1([O-1])=CCCC[C-1]1C=O
pKa is 12.815532631320009

C1(O)=CCCCC1C=O -> C1([O-1])=CCCCC1C=O
pKa is 10.780590232640959

C1([O-1])=CCCCC1C=O -> C1([O-1])=CCCC[C-1]1C=O
pKa is 12.980583901142545

C1(CCCCC1C=O)=O -> C1(CCCC[C-1]1C=O)=O
pKa is 7.510965968450897

Macro pKa:

C1(C(=CO)CCCC1)=O, C1(O)=C(C=O)CCCC1, C1(O)(CCCCC1C=O)O, C1(CCCCC1C(O)O)=O,
C1(CCCCC1C=O)=O ----> C1(CCCC[C-1]1C=O)=O, C1([O-1])=C(C=O)CCCC1, C1(C(=C[O-1])CCCC1)=O
Macro Pka = 8.969012

3.6 Conclusion

The three chemical process models, hydration, tautomerization and speciation models were integrated. Filters were designed and implemented to reduce the complexity of the calculations, the filters are based on threshold values deciding productive paths based on aggregated equilibrium constants. The reliability of the calculations are computed and unreliable calculations are ignored.

The lack of through quantitative analysis of complex coupled process is a major stumbling block to modeling such systems. SPARC's models do a very good job at qualitative analysis of these systems. For modeling tautomers one of the important pKas is the carbon acid pKa. Due to the magnitude of the perturbations the RMS of the model is high compared to other pKa reaction models. Another way to test these models is to let the researcher use the system and report any error they find. A web based interface for the integrated process models has also been implemented and will be available for public use in December of 2004.

The speciation models in SPARC are very well tuned and are getting even better. Recently Pfizer Chemical have started using the more current version of SPARC, in doing so they have started to compare and benchmark SPARC models with the large collection of pKa data they have accumulated. They are collaborating with us to improve SPARC; one of the recent changes in SPARC pKa models has been the complete rewrite of the nitrogen models. This was done after feedback from Pfizer researchers, who provided us with information, which convinced us to handle nitrogen in rings in a different manner.

The next phase of this research would be to integrate more physical process models like Henry's constant, solubility and vapor pressure models into this existing coupled process. Models for auto tautomerization (tautomerization in itself) have to be developed and tested.

Chapter 4

Interpretation of Cis-trans isomer information in SMILES representation

4.1 Introduction

SMILES (Simplified Molecular Input Line Entry System) was developed by David Weininger as a method to encode chemical molecules based on simple grammar and is capable of specifying complete structural information[10, 11, 12]. This representation is based on expressing the molecule by atomic symbols and other symbols to represent bonds and other features in a linear fashion[11] . As a result of its simple grammar, it proved to be easy to encode by hand as well as easy for computer programs to interpret. This led to its usage in a number of computer programs and also as a way to store chemical structures in computer databases.

The SMILES encoding rules do not place any restriction on the starting point in a molecule when encoding it. This leads to multiple ways of encoding a molecule in SMILES. This poses a problem when SMILES notation is used to identify the molecule by a simple string comparison. The solution is to generate a unique SMILES notation[10]. A unique SMILES is a unique way of encoding a molecule in SMILES. The uniqueness is due to the weighting of the atoms in the molecule based on their connectivity and nature of the atoms themselves. This process is known as the generation of a unique SMILES for that molecule. The simplicity of SMILES specification rules led to its adaptation as a form of molecular input/output for chemistry software. One of the side effects of this adaptation is the development of incompatible specification rules for interpretation of cis-trans isomer information. The atom weighting system for the molecules to

generate unique SMILES is also not standardized. This leads to the generation of different unique SMILES depending on the software and weighting system used[13, 14, 15].

In SPARC models SMILES notation is used to represent the molecule internally. Hence we have to convert all other form of molecular representation into SMILES notation. In the initial form SPARC did not use structural isomerism information to model physical and chemical property. But when the heat of formation calculator was added to SPARC the specification of double bond isomerism and tetrahedral chirality became very important. In the process of incorporating structural isomer models we discovered several problems with SMILES notation and set out to fix them.

The SMILES grammar in its published state was sufficient in defining simple cis-trans and tetrahedral chirality. However when applied to complex ring systems with multiple conjugations of double bonds such as (18-Annulene), this grammar fails to represent all structural information. Specification of cis-trans across a ring break also was not addressed in the originally published grammar. We have attempted to work out the rules to specify complete structural information using SMILES while adhering to the original symbols.

In table 4.1 a few SMILES and their interpretation by different chemistry software are given. The table illustrates that all the software follow the basic cis-trans interpretation rules as specified by the daylight specification. This is illustrated by the molecules trans-2-butene and cis-2-butene. The SMILES representation of the other four molecules involves the specification of cis-trans isomer information through a ring break. In the case of these molecules, it is possible to write a SMILES representation without passing through a ring break, but to illustrate the issues with interpretation rules we have written them as such. For the four ring compounds ChemDraw, ACD/ChemSketch and depict seem to ignore the cis-trans specification. The

interpretation rules of Marvin and SPARC interpret in some case, opposite of each other and the same for the other.

Table 4.1 SMILES strings and its interpretation by different software

Molecule	SMILES	MARVIN	ACD/ChemSketch	ChemDraw	SPARC	Depict
trans-2-butene	C\C=C\C	Trans	Trans	Trans	Trans	Trans
trans-2-butene	C(/C)=C\C	Trans	Trans	Trans	Trans	Trans
cis-2-butene	C\C=C/C	Cis	Cis	Cis	Cis	Cis
cis-2-butene	C(\C)=C\C	Cis	Cis	Cis	Cis	Cis
Trans-1-methyl-octene	C\C1CCCCCC/C=1	Cis	Trans	Trans	Trans	Trans
cis-1-methyl-octene	C/C1CCCCCC/C=1	Trans	Trans	Trans	Cis	Trans
trans octene	C1=C\CCCCCC\1	Cis	Cis	Cis	Cis	Cis
cis octene	C1=C\CCCCCC/1	Trans	Cis	Cis	Trans	Cis

The above example clearly illustrates the need for a more compressive and standard cis-trans interpretation rules for SMILES. The interpretation rules developed for SPARC have been tested for over a 1000 compounds containing cis-trans information. All these compounds were downloaded from NIST Webook[16] as MDL mol files and SMILES strings were encoded for them. These SMILES strings were then interpreted according to the same rules and were verified to be self consistent. A standard weighting scheme for the generation of unique SMILES also need to be developed. The algorithm used by SPARC is based on the original algorithm

published by David Weininger. This algorithm is by no means efficient. A standard algorithm will help the developers of software use a more portable chemical information database. In the generation of unique SMILES for molecules with cis-trans and tetrahedral chirality we had to impose some restriction on the use of symbols. Specifically the restriction determines the sense of the first slash used in specifying an isomer. When developing a standard this area also has to be addressed. In the following pages we attempt to propose a set of standard rules which needs to be examined and tested and worked into an acceptable standard.

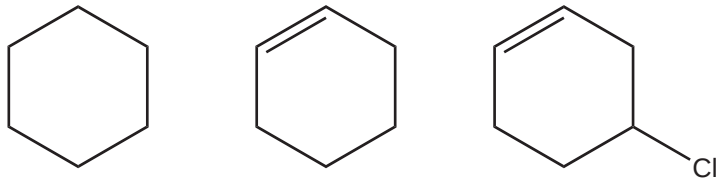
4.2 SMILES Grammar Review

Let us review some SMILES grammars that are important for understanding the problems in specification of cis-trans isomerism for complex rings. A simple compound like 2-butene is represented in SMILES by 'CC=CC'. No information about the orientation of the molecule (cis or trans) is specified here. The orientation of atoms around a double bond is specified in SMILES using the characters '/' or '\'. The slashes represent directional single bonds connecting the *anchor atoms*. In this paper we will be using the terms "*anchor atoms"* and "*other atoms"*. *Anchor atoms* are the atoms that form the double bonds and *other atoms* are the atoms which specify the configuration and are connected to their respective *anchor atoms*. For example cis-2-butene will be represented by 'C\C=C/C' and trans-2-butene by 'C\C=C\C'. For the cis conformation the two slashes are pointing up from the anchor atoms. Likewise, for the trans conformation, one single bonds is pointing up, one is pointing down.

Since SMILES encodes molecules as a linear string of characters, rings have to be broken into linear representations. This is achieved by breaking the ring and tagging the ring-break-atoms with numbers (ring-tags) and writing them in linear fashion. For example cyclohexane is represented by 'C1CCCCC1'. When decoding this SMILES string we have to join the atoms

with the same ring tags to get the actual representation. When generating unique SMILES for rings, the occurrence of ring breaks cannot be predetermined. The ring break is determined by the order of atoms in the string. The order is determined by the weights calculated for individual atoms in the process of generating a unique SMILES string.

In Figure 4.1 the unique SMILES notation are given for cyclohexane, cyclohexene and 4-chlorocyclohexene. All three compounds are ring compounds and the ring-break atoms are labeled by number 1 in the notation. This example illustrates the change in the ring break atoms in the molecules with the addition of a substituent. As already stated, the change in ring-break position is due to the change in the calculated weights of the atoms. Since the same algorithm must be applied to all molecules in the generation of unique SMILES strings, the ring-break selection is independent of user control.



C(CCC1)CC1	C(=CCC1)CC1	ClC(CC=C1)CC1
cyclohexane	cyclohexene	4-chlorocyclohexene

Figure 4.1 Representing rings in SMILES string

4.3 Conjugated ring systems

When specifying isolated cis-trans isomers in a molecule the sense of slashes are independent from the influence of other slashes that might be used to specify other isomers. In

the case of conjugated ring systems once the sense of the first slash is set, the sense of all other slashes involved in that conjugated system is also fixed. This can be observed in figure 4.2. 18-anulene is shown with its SMILES notation. In the notation you can observer that every slash is responsible for defining two isomer specifications. This type of slash usage is common and by itself is not a problem; but in case of conjugated rings it could cause a conflict, where the specification of the last double bond in the conjugated system cannot be specified correctly. The reason is that the senses of the two slashes for that double bond are already fixed and cannot be changed with out disturbing all the other specifications. This conflict cannot be resolved based on the current rules. After processing a large number of molecules listed in the NIST Webbook and building 3D models[6], we did not find an example of trans double bond configuration in a ring smaller than eight atoms. From our own 3D models it appears molecules do not have the bond angles necessary to form trans double bonds in rings smaller than eight atoms. Hence we assume the default configuration for all double bonds in a ring smaller than eight atoms to be "cis". New rules have been developed for the interpretation of cis-trans information for conjugated ring systems. In combination with our weighting system for the atoms we can write self-consistent SMILES representations for conjugated ring systems.

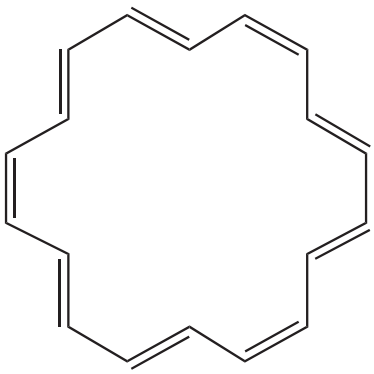


Figure 4.2 18-Annulene. A completely conjugated molecule with a mix of cis and trans isomers. "C(=C\C=C\C=C\C=C/C=1)\C=C\C=C\C=C/C=C/C1" is the SMILES representation of this molecule.

4.4 Approach to solve cis-trans specification across ring breaks

Our goal was the development of a rule set which will enable self consistent encoding / decoding of cis-trans isomer information across ring breaks. In order to represent the information in a computationally facile way, we have defined some terms to identify the participants. The two anchor atoms forming the double bond are identified *anchor1* and *anchor2*. The atoms attached to the anchor atoms and which define conformation, are identified as *other1* and *other2*. An example is given in Figure 4.3.

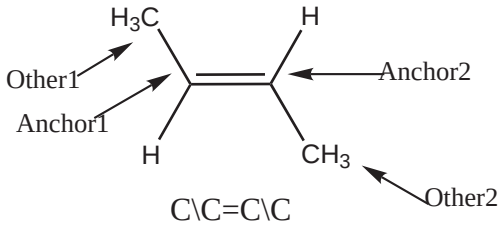


Figure 4.3 Trans 2-butene. Illustrating the cis-trans nomenclature.

The rules are based on the projection of the smiles in 3-D space before interpreting the conformation. Let us consider a simple molecule, trans-2-butene. The simple way to write the SMILES representation is to start from one end of the molecule and proceed to the other end. The SMILES is 'C\C=C\C'. We start from the *other1* atom (Methyl) and use a back slash to represent the direction of the single bond to the *anchor1* atom. The forward slash after the *anchor2* atom is representing the direction of the single bond, which in this case is pointing away from the *other1* atom and hence representing the trans conformation. As a general rule, when

both slashes are of the same "sense", they represent a trans conformation and when they are different, they represent cis conformation.

The example that we saw above was written in a simple straightforward way. Whenever the molecule is encoded with branching and or ring breaks the user has to take into account that the molecule is basically turning at the branch point or ring break as the case may be. This complicates the cis-trans specification through the branch or ring break in such molecules. The complication is due to the change in progression of the atoms involved in the specification. This leads to two different interpretations of the isomer configuration, the apparent sense that we get from interpreting the conformation just from the sense of the slashes and the real (actual) sense that we interpret taking into account the direction of progression of the molecule. After considering all possible ways of writing a SMILES representation of a double bond we have two cases where the real sense is an inverse of the apparent sense.

There are two cases where the apparent conformation has to be inversed to get the actual conformation. The reason behind this reversal is the relative position of the other atom with respect to its *anchor atom*. When they are not in the regular order (Other1/Anchor1=Anchor2\Other2) the sense of the slash (single bond) might have to be reversed to get the actual orientation. For example let us consider trans-2-butene now written with the relative position of *anchor1* and *other1* atoms reversed. The SMILES string would be "C(\C)=C/C". When decoding the SMILES string to get the right conformation, the sense of the first slash has to be reversed before interpreting the conformation. This is because the position of *other1 atom* is after *anchor1 atom*, which when projected in 3D will show the other1 atom away from the other2 atom.

The rules for reversing the apparent sense of slashes are given below with examples illustrating the situations.

1) When the *other1* atom and *anchor1* atom positions are reversed, the apparent sense should be reversed. For example trans-2-butene written as "C(\C)=C/C"
2) When there is a ring break between *anchor1* and *anchor2* atoms and the *other2* atom comes before the *anchor2* atom in the SMILES string, the apparent sense has to be reversed. For example trans-1-methyl-cyclooctene can be written as "C\C1CCCCCC/C=1"

For all cases other than the two described above, the real sense is the same as the apparent sense. The above rules have been tested on a large set of SMILES representations and can handle all possible ways of writing a cis-trans specification across a ring break except in the case of completely conjugated rings. The reason, as described above, is the conjugation which "locks down" the sense of the slashes in the string once one of the double bond in the conjugation is specified. This could pose a problem where the user might not be able to specify the correct isomer for the last double bond in the conjugated system. One way to resolve this deadlock is to prevent a ring-break from occurring across a double bond in the conjugated system and use two slashes to specify the isomer for both double bonds. These two slashes will be the leading and trailing slashes of the single bond. The example below will give a clearer picture of this idea.

Let us consider Cycloocta-1,3,5,7-tetraene's two isomers for encoding into SMILES. The first isomer is the one with two trans double bonds and the other is the all cis isomer.

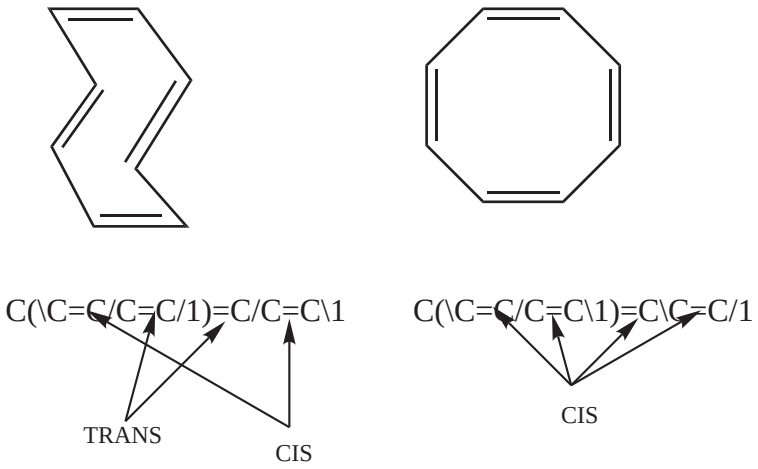


Figure 4.4 The two isomers of cycloocta-1, 3, 5, 7-tetraene.

The SMILES strings for the two isomers are given under each picture. In the first molecule there are two trans and two cis isomers. The conformation of the second double bond and the fourth is specified through a ring break and since this is a fully conjugated system a leading and trailing slash is used to separate the specification of the two double bonds. The leading and trailing slashes are both associated with the ring break. For this system to work, the ring break should not be a double bond. When unique SMILES are generated for this kind of systems, the algorithm to generate them has to take this rule into account.

4.5 Influence of Cis-trans on Unique SMILES string generation

In the original algorithm for the generation of unique SMILES strings, neither cis-trans isomers nor tetrahedral chirality was taken into account when weighting the atoms in the molecule. After working with many molecules we decided that change in the weights of the atoms due to cis-trans or other structural specification was not the best way to generate unique SMILES. We feel that the molecules should be similarly ordered (similarly weighted)

irrespective of their structural conformations, as this will help the user in identifying similar molecules.

But the question of generating unique SMILES with structural information still remains. To generate unique SMILES with structural information, a standard for the use of symbols has to be developed. In the case of cis-trans a standard encoding procedure would define selection rules to choose atoms to use in the definition of the isomer. It also defines a standard sense of the slash to be used when writing the very first conformation. We use two rules to determine the atoms participating in the isomer specification.

1) If there is a choice for the selection of the *other atom* the higher weighted of the two should be selected as the *other atom*.
2) If possible cis-trans specification through ring breaks should be avoided

4.6 Summary

The rules for complete structural specification in SMILES string were not defined very well in the original SMILES specification. This has led to the development of various implementations of the SMILES parser, each with its own set of rules. In the case of specifying cis-trans information, interpretation rules have been modified to allow proper and consistent interpretation. The problem with conjugated rings has also been addressed and rules developed to properly specify conformation for all double bonds involved in the conjugated system. Ideas for the development of an algorithm to generate unique SMILES involving cis-trans and chirality were also stated. Hopefully this improvement to SMILES will bring about an increase in its adoption as a standard molecular representation method.

Chapter 5

Development of Tools for SPARC

5.1 Introduction

One of the goals of the SPARC group is to provide free access to the research community to perform calculations using SPARC. It is not feasible to distribute a SPARC calculator to every person who wants to use it. Hence, a client, acting as a bridge to the actual SPARC server was designed and built. SPARC clients were built for the Sun Solaris and Windows operating systems. These clients are distributed to researchers wanting to use SPARC. Because of the chore of maintaining client releases for different versions of operating systems and for unifying access to SPARC, a web client application for SPARC was developed. Communication to SAPRC servers was handled by a custom protocol riding over the TCP/IP communication protocol. A Windows COM object was designed as a bridge to interface the web application server and SPARC server. This COM object was written in Visual Basic. As new calculators are added to SPARC, the communication protocol has to be expanded to add support for these new applications. This new protocol has to then be implemented in the COM object. Figure 5.1 is a visual illustration of the integration of the SPARC web application.

5.2 SPARC Server Manager

The SPARC website is capable of handling multiple users at one time. Although SPARC server are “blocking” when doing a calculation (does not respond to other requests) a mechanism to handle multiple users at the same time was designed and implemented inside the web

application server. This method was not very portable and could not be expanded to provide this management capability to other web servers. A Server Manager was designed and implemented to handle the assignment of calculations to a free server and in a way "load balance" the calculation requests.

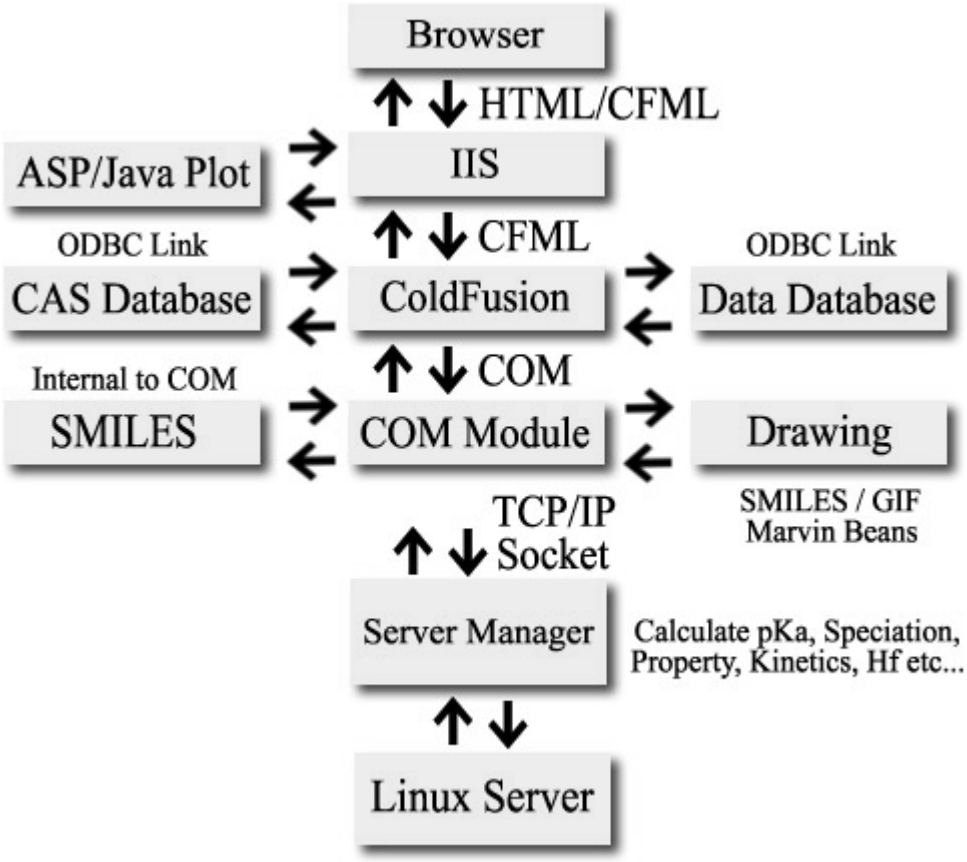


Figure 5.1 SPARC Web Application Integration Diagram

When designing the server manager, it was decided to make the system an "active". It is known as "active" because it probes its list of SPARC servers to determine the state of each

process and notifies the administrator in case of server failure. An authentication mechanism for access to the server manager was implemented based on the client addresses. The server was also responsible for furnishing login information for the clients. The server continuously maintains a list of all active connections to its servers and the clients who are using it. A special protocol was developed to communicate and negotiate a server request for all clients. A mechanism was also included to enquire about the status of the servers and to force an emergency rescan the status of all the SPARC servers.

5.2.1 Implementation

The server manager was designed to run as a Windows service so it can be started without user intervention. A Microsoft supplied ActiveX Control (NTSVC.OCX) was used to implement it as a service. All communication is done using TCP/IP protocol and the queries and replies are all in plain English.

The internal representation of the server information is handled by a custom data structure called “resource”. When the service is started, the server and client information is read from an initialization file (.ini file) and an array of the “resource” data structure is filled. A submodule is then called with the list of resources to determine the status of the SPARC servers by performing a login operation. The result of the login operation is then stored back in the resource structure. When a client logs on to the server manager and requests a connection by asking for a “password”, which is the protocol for requesting a server to perform calculations, the task is assigned to another submodule to perform. This looks at the list of resources and finds a free server, it then tests the connection by performing a test login operation using the correct username. If it is successful, it then sends the client the information for connecting to the free server and adds an entry in the resources structure that a server has been assigned to that

particular client. After the client is done using the services of the SPARC server, a call is made to the server manager and requests the release of the server resource that was assigned to it. The server manager then marks the SPARC server as a free resource.

The requests from clients are queued in a ring buffer until the worker process is ready to process them. The worker was designed to be a single threaded application to avoid race conditions when handling the resource data.

A complete listing of code for this program and its client code piece is listed in Appendix A.

5.2.2 Data Structures and Communication Protocol

Definition of resources structure:

```
Public Type resources
    Machine As String * 16
    Status As Boolean
    Operational As Boolean
    Port As Integer
    User As String * 16
    Client As String * 16
    Tries As Integer
    Type As String
    start_time As String
End Type
```

Client Communication Protocol:

- Request for status of all server:

  Client sends “status”

  Server sends back a formatted list of data in resource data structure

- Request for a free resource (SPARC Server)

  Client first sends the type of SPARC server it needs (“new” or “old”)

Client sends “password”

Server sends back “**password” and a comma delimited string with a unique connection id, IP address of the server, port number and user name

- Request for releasing a resource

  Client sends “release,” followed by a comma separated list of connection ids it wants released

5.2.3 Scope for Future Improvements

SPARC has always been of interest to pharmaceutical companies and other commercial ventures. These industries are very particular about privately maintaining their research information. Therefore, they do not want to use the SPARC Web Interface over the internet. Currently Pfizer is using the latest version of SPARC for calculating pKas, and would like to use it on their internal network to provide access to all Pfizer personnel. In these cases the SPARC server has to communicate with their web server. It is impossible for them to implement the whole SPARC communication protocol; the ideal situation would be to provide them an interface like the COM object which is currently used in SPARC web interface, implemented in a universal and platform independent language like Java. This would also require that the server manager be rewritten in Java.

In future versions, dynamically scalable SPARC servers will be very desirable. This would enable the server manager to dynamically spawn new SPARC servers on remote Unix machines based on the user load. Currently the SPARC sever resource is limited and is a fixed number and in the case of a heavy load it would be unable to handle it.

5.3 Multiple Remote Training Utility

SPARC's mechanistic toolbox models are based on parameters that define the properties of reaction groups, substituents and reactophores. These parameters are calculated based on observed functions using a process known as *training*. In *training* these parameters are fitted using a least-squares method to fit the observed data. The training sets are designed to optimize a batch of parameters using molecules that are affected by these parameters. The number of calculation increase proportionally to the product of the number of parameters and the number of molecules in a training set. In training large sets of parameters, the number of calculations is very large and in order to speed up the training a split training process was designed. In this split training process the training set is divided into smaller sets with a part of the parameters and all of the molecules. The smaller training sets are then sent to different machines and the co-variant matrices are calculated at the same time, this reduces training time. The training data, i.e. the results of the training, are then combined and a least squares fit is done.

The multiple split training was first done manually and was a tedious process. Hence it was decided to automate it. Based on a previous program that split the training files we, set out to write a fully automated version. The design goal was to provide a fully Windows based tool to perform fully automated split training. Since the SPARC servers are based on a Unix machine a program was written that would in collaboration with the Windows client will perform the necessary operations on the Unix side.

The information about the servers that are part of this training cluster is stored in a .ini file. This file also contains all the user information the program need to login and perform file transfers. Other information about the current training is collected from the user. This information contains the type of training to be performed, i.e. pKa, property, hydrolysis, Heats of

Formation training, the multipliers, if any, used to scale training data, and other options which switch on/off particular pieces of code. The SPARC training module has mechanisms to store training data and output into separate directories in the user area for each training cycle. This enables users to train multiple passes without intervention and later look at the change in each parameter for each cycle. The multiple training client was also designed to take advantage of this functionality. After each pass the system combines the data from all servers in the cluster and after performing the least-squares fit stores the data in a subfolder on each machine in the cluster. In doing so, the system can perform another training cycle based on the parameters optimized in the first cycle.

The next phase of development in a remote training utility is to design an interface to store all the trainable parameters in chronological order in a MySQL Database. This would let the SPARC developers to go back and take a look at the parameters as a function of time. A side effect of this database based storage of parameter is a chance to use this data to write analysis tools to look at the behavior of parameters, etc. Developers could also write parameter constraints so when a parameter exceeds set boundaries, the system can monitor this and notify the developers.

5.4 Conclusion

Multiple tools and miscellaneous code snippets were written to enhance research methods and provide access to the SPARC severs. A new version of a standalone pKa client is being developed for industries interested in using SPARC to calculate the properties for chemicals used in their research. A new Java version of the client will soon be developed to facilitate the client to be platform independent.

References

1. Said H. Hilal. 1992. Prediction of Physical Properties and Chemical Reactivity Parameters From Molecular Structure by SPARC. PhD diss., The University of Georgia. .

2. Neil Isaacs. 1996. Physical Organic Chemistry. England: Addison Wesley Longman, .

3. S. H. Hilal, L. A. Carreira and S. W. Karickhoff. 1994. Estimation of Chemical Reactivity Parameters and Physical Properties of Organic Molecules Using SPARC. Chap. in Quantitative Treatments of Solute / Solvent Interactions. 291. New York: Elsevier, .

4. S. H. Hilal, S. W. Karickhoff, L. A. Carreira. 1999. Estimation of Microscopic, Zwitter Ionization Constants, Isoelectric Point and Molecular Speciation of Organic Compounds. Talanta 50: 827.

5. L. A. Carreira. Discussion. Athens, Ga.: Dept. of Chemistry.

6. Robert T. Morrison and Robert N. Boyd. 1963. Organic Chemistry. Boston: Allyn and Bacon, Inc., .

7. Souad A. M. Shaaban. 1998. Prediction of Keto - Enol Equilibrium Constants by Computer. M.S. thesis, The University of Georgia. .

8. J. Toullec. 1990. Keto-Enol Equilibrium Constants. Chap. in The Chemistry of Enols. New York: John Weily and Sons Ltd., .

9. S. H. Hilal, L. L. Bornander and L. A. Carreira. Calculation of Hydration Constants Using SPARC. Submitted for Publication.

10. David, Arthur Weininger and Joseph L. Weininger. 1989. Smiles, 2. Algorithm for Generation of Unique SMILES Notation. Journal of Chemical Information and Computer Sciences, 29: 97.

11. David Weininger. 1988. Smiles, A Chemical Language and Information System. 1. Introduction to Methodology and Encoding Rules. Journal of Chemical Information and Computer Sciences, 28: 31.

12. David Weininger. http://www.daylight.com: Daulight Chemical Information Systems, Inv..

13. Chemdraw. http://www.cambridgesoft.com: Cambridgesoft Corporation.

14. Depict. http://www.daylight.com/depict: Daylight Chemical Information Systems, Inv..

15. MarvinSketch. http://www.chemaxon.com/marvin: ChemAxon, Ltd..

16. NIST Webbook. http://webbook.nist.gov: NIST.

Appendix A

Code Listing for SPARC Server Manager

```
" File SSM.vbp
Type=Exe
Reference=*\G{00020430-0000-0000-C000-
000000000046}#2.0#0#C:\WINNT\System32\Stdole2.tlb#OLE Automation
Object={33335123-F789-11CE-86F8-0020AFD8C6DB}#2.0#0; IPDAEM35.Ocx
Object={33335113-F789-11CE-86F8-0020AFD8C6DB}#2.0#0; IPPORT35.Ocx
Object={33335233-F789-11CE-86F8-0020AFD8C6DB}#2.0#0; SMTP35.Ocx
Object={E7BC34A0-BA86-11CF-84B1-CBC2DA68BF6C}#1.0#0; ntsvc.ocx
Form=Main_frm.frm
Module=INISET; Iniset.bas
Module=Worker; Worker.bas
IconForm="Main_frm"
Startup="Main_frm"
HelpFile=""
Title="Server Manager"
ExeName32="SSM.exe"
Command32="-debug"
Name="SSM"
HelpContextID="0"
CompatibleMode="0"
CompatibleEXE32="SSM.dll"
MajorVer=3
MinorVer=0
RevisionVer=16
AutoIncrementVer=1
ServerSupportFiles=0
VersionCompanyName="UGA"
CompilationType=0
OptimizationType=2
FavorPentiumPro(tm)=0
CodeViewDebugInfo=0
NoAliasing=0
BoundsCheck=0
OverflowCheck=0
FlPointCheck=0
FDIVCheck=0
UnroundedFP=0
StartMode=0
Unattended=0
Retained=0
```

```
ThreadPerObject=0
MaxNumberOfThreads=1

[MS Transaction Server]
AutoRefresh=1


" File main_frm.frm
VERSION 5.00
Object = "{33335123-F789-11CE-86F8-0020AFD8C6DB}#2.0#0"; "IPDAEM35.Ocx"
Object = "{33335113-F789-11CE-86F8-0020AFD8C6DB}#2.0#0"; "IPPORT35.Ocx"
Object = "{33335233-F789-11CE-86F8-0020AFD8C6DB}#2.0#0"; "SMTP35.Ocx"
Object = "{E7BC34A0-BA86-11CF-84B1-CBC2DA68BF6C}#1.0#0"; "ntsvc.ocx"
Begin VB.Form Main_frm
  Caption        =   "Server Manager"
  ClientHeight   =   4050
  ClientLeft     =   3465
  ClientTop      =   3765
  ClientWidth    =   6225
  LinkTopic      =   "Form1"
  ScaleHeight    =   4050
  ScaleWidth     =   6225
  Begin VB.Timer res_timer
    Enabled        =   0   'False
    Interval       =   60000
    Left           =   3480
    Top            =   2880
  End
  Begin VB.Timer Timer
    Enabled        =   0   'False
    Left           =   2520
    Top            =   1800
  End
  Begin NTService.NTService NTService1
    Left           =   840
    Top            =   600
    _Version       =   65536
    _ExtentX       =   741
    _ExtentY       =   741
    _StockProps    =   0
    DisplayName    =   "Sparc Server Manager"
    ServiceName    =   "SparcSSM"
    StartMode      =   3
  End
  Begin VB.Timer rescan_timer
    Enabled        =   0   'False
```

```
    Left            =   1320
    Top             =   3000
  End
  Begin VB.Timer Timer1
    Left            =   5400
    Top             =   1320
  End
  Begin SMTPLib.SMTP SMTP1
    Left            =   2520
    Top             =   3120
    _ExtentX        =   741
    _ExtentY        =   741
    MailServer      =   ""
    From            =   ""
    To              =   ""
    Cc              =   ""
    BCc             =   ""
    ReplyTo         =   ""
    Date            =   ""
    Subject         =   ""
    MessageText     =   ""
  End
  Begin IPPortLib.IPPort IPPort1
    Left            =   5400
    Top             =   1920
    _ExtentX        =   741
    _ExtentY        =   741
    RemoteHost      =   ""
    Linger          =   -1  'True
    EOL             =   ""
  End
  Begin IPDaemonLib.IPDaemon IPDaemon1
    Left            =   5400
    Top             =   600
    _ExtentX        =   741
    _ExtentY        =   741
    Linger          =   -1  'True
  End
End
Attribute VB_Name = "Main_frm"
Attribute VB_GlobalNameSpace = False
Attribute VB_Creatable = False
Attribute VB_PredeclaredId = True
Attribute VB_Exposed = False

Private Sub Form_Load()
```

```
On Error GoTo Err_Load
   Me.Hide
   Dim a$
   a$ = App.Path & "\server.log"
   Open a$ For Append As #1
   Print #1, Date$
   'Close #1
   Dim strDisplayName As String
   Dim bStarted As Boolean

   strDisplayName = NTService1.DisplayName


   If Command = "-install" Then
      ' enable interaction with desktop
      NTService1.Interactive = False

      If NTService1.Install Then
         Call NTService1.SaveSetting("Parameters", "TimerInterval", "150")
         MsgBox strDisplayName & " installed successfully"
      Else
         MsgBox strDisplayName & " failed to install"
      End If
      End
   ElseIf Command = "-uninstall" Then
      If NTService1.Uninstall Then
         MsgBox strDisplayName & " uninstalled successfully"
      Else
         MsgBox strDisplayName & " failed to uninstall"
      End If
      End
   ElseIf Command = "-debug" Then
      NTService1.Debug = True
   ElseIf Command <> "" Then
      MsgBox "Invalid command option"
      End
   End If

   Dim parmInterval As String
   parmInterval = NTService1.GetSetting("Parameters", "TimerInterval", "150")
   Timer.Interval = CInt(parmInterval)

   ' enable Pause/Continue. Must be set before StartService
   ' is called or in design mode
   NTService1.ControlsAccepted = svcCtrlPauseContinue
```

```
    ' connect service to Windows NT services controller
    NTService1.StartService
    Exit Sub
Err_Load:
    If NTService1.Interactive Then
        MsgBox "[" & Err.Number & "] " & Err.Description
        End
    Else
        Call NTService1.LogEvent(svcMessageError, svcEventError, "[" & Err.Number & "] " &
Err.Description)
    End If
End Sub

Private Sub IPDaemon1_Connected(ConnectionID As Integer, StatusCode As Integer,
Description As String)
 'Accept / Reject connection from clients based on valid IP address
    a$ = IPDaemon1.RemoteHost(ConnectionID)
    For i = 1 To numClients
        If Trim(Clients(i)) = Trim(a$) Then
            IPDaemon1.EOL(ConnectionID) = Chr$(10)
            Exit Sub
        End If
    Next i
    IPDaemon1.Connected(ConnectionID) = False
End Sub

Private Sub IPDaemon1_DataIn(ConnectionID As Integer, Text As String, EOL As Integer)
On Error GoTo err_data_in
If InStr(Trim(Text), "release") Then 'This is a request for release of resources
    j = InStr(Trim(Text), ",")       'Filter all resource index and put it in Value$
    Value$ = Mid$(Trim(Text), j + 1)
    Text = "release"
End If
Select Case Trim(Text)
    Case "status" 'Client is asking for status of resources
        ActionQ(ActionPointer) = "status!" & Trim(Str(ConnectionID))
        IncActionPointer
    Case "emergency_reset" 'Client is asking for a reset of resources
        ActionQ(ActionPointer) = "emergency_reset!" & Trim(Str(ConnectionID))
        IncActionPointer
    Case "rescan" 'Client has requested for a rescan of all resource and validation
        ActionQ(ActionPointer) = "rescan!" & Trim(Str(ConnectionID))
        IncActionPointer
    Case "old" 'Client has requested for a old server
        ActionQ(ActionPointer) = "old!" & Trim(Str(ConnectionID))
        IncActionPointer
```

```
    Case "new" 'Client has requested for a new server
        ActionQ(ActionPointer) = "new!" & Trim(Str(ConnectionID))
        IncActionPointer
    Case "password" 'Client has requested for a valid resource
        ActionQ(ActionPointer) = "password!" & Trim(Str(ConnectionID))
        IncActionPointer
    Case "release" 'Client has requested a release of the resource it had used
        ActionQ(ActionPointer) = "release!" & Trim(Str(ConnectionID)) & "!" & Value$
        IncActionPointer
    Case "logoff" 'Client is disconnecting
        IPDaemon1.Connected(ConnectionID) = False
    Case Else
        a$ = "Unknown request from client: " & Trim(Text)
        'Call NTService1.LogEvent(svcEventInformation, svcMessageInfo, a$)
        logMsg (a$)
End Select
Exit Sub
err_data_in:
        a$ = "General Failure in Data in " & Err.Number & "  " & Err.Description
        'Call NTService1.LogEvent(svcEventInformation, svcMessageInfo, a$)
        logMsg (a$)
End Sub

Private Sub IPDaemon1_Disconnected(ConnectionID As Integer, StatusCode As Integer,
Description As String)
  'Dim x As Long
    'Client pulled the plug
'    Timer1.Enabled = False
'    a$ = Str$(ConnectionID)
'    For i% = 0 To ConnectionList.ListCount - 1
'      If ConnectionList.List(i%) = a$ Then
'        ConnectionList.RemoveItem i%
'        Exit For
'      End If
'    Next
'    DoEvents
'    DoEvents
'    If ConnectionList.ListCount < 1 Then
'      Debug.Print "Disconnect from ipdaemon"
'      x = Disconnect
'    End If
'    Timer1.Enabled = True

End Sub

Private Sub IPPort1_Connected(StatusCode As Integer, Description As String)
```

```
 If StatusCode = 0 Then
   'made a connection
   'testlogin and completed login Flags are set in ReadyToSend event
  Else
   'connection failed
    testLogin = False
    completedLogin = False
    a$ = "Login from ipport1 failed"
    'Call NTService1.LogEvent(svcEventInformation, svcMessageInfo, a$)
  End If
End Sub

Private Sub IPPort1_DataIn(Text As String, EOL As Integer)
    Static in_proc$, c$, flag%
'This piece of code filters the header for the lines sent and
'sends it as a seperate variable to a sub calles parse_sock
'it also sets lags for the line number that it is reading.

    If EOL = False Then
       c$ = c$ + Text
       flag% = True
       Exit Sub
    End If

    If EOL And flag% Then
        a$ = c$ + Text
        c$ = ""
        flag% = False
    Else
        a$ = Text
    End If
'Debug.Print a$
    ' for nt
    If a$ <> "" Then
     If Asc(Right$(a$, 1)) = 13 Then
      a$ = Mid$(a$, 1, Len(a$) - 1)
     End If
    End If

    'check this out for traps
    'If Left$(a$, 2) = "**" Then
    ' If Mid$(a$, 3, 2) <> "SS" And Mid$(a$, 3, 2) <> "tw" And in_sock_read Then
    '  in_sock_read = 0
    '  read_level = 0
    ' End If
    'End If
```

```
'    Debug.Print A$
    safe$ = in_proc$

    If in_sock_read Then
        Call parse_sock(safe$, a$)
    Else
        in_proc$ = LTrim$(RTrim$(a$))
        in_sock_read = True
        read_level = 0
    End If

End Sub

Private Sub IPPort1_Disconnected(StatusCode As Integer, Description As String)
 'socket disconnected by choice or error
 ' Text1.Text = "Not Connected to Server but" & vbCrLf & _
 '            "Status will still work!"
  not_first_time = False
  HostConnected = False
'  Set cSysTray1.TrayIcon = Image1(2).Picture

End Sub

Private Sub IPPort1_ReadyToSend()
  'first time connect and process a request
  If Not testLogin Then
    Exit Sub
  Else
    testLogin = False 'set flag to break out of loop
    completedLogin = True 'set flag that says that the connect was successfull
  End If
End Sub


Private Sub NTService1_Continue(Success As Boolean)
On Error GoTo Err_Continue

    Timer.Enabled = True
    Success = True
    Call NTService1.LogEvent(svcEventInformation, svcMessageInfo, "Service continued")
    Exit Sub

Err_Continue:
```

```
   Call NTService1.LogEvent(svcMessageError, svcEventError, "[" & Err.Number & "] " &
Err.Description)

End Sub

Private Sub NTService1_Pause(Success As Boolean)
On Error GoTo Err_Pause

   Timer.Enabled = False
   Call NTService1.LogEvent(svcEventError, svcMessageError, "Service paused")
   Success = True
   Exit Sub

Err_Pause:
   Call NTService1.LogEvent(svcMessageError, svcEventError, "[" & Err.Number & "] " &
Err.Description)

End Sub

Private Sub NTService1_Start(Success As Boolean)
On Error GoTo Err_Start
   Call getReady
   Timer.Enabled = True
   Success = True
   res_timer.Enabled = True
   Exit Sub

Err_Start:
   Call NTService1.LogEvent(svcMessageError, svcEventError, "[" & Err.Number & "] " &
Err.Description)

End Sub

Private Sub NTService1_Stop()
On Error GoTo Err_Stop
   Close #1
   Unload Me

Err_Stop:
   Call NTService1.LogEvent(svcMessageError, svcEventError, "[" & Err.Number & "] " &
Err.Description)

End Sub

Private Sub res_timer_Timer()
   For i = 0 To Total_Resource
```

```
      If Res(i).Operational And Res(i).Status And (Trim(Res(i).start_time) <> "") Then
         If (DateDiff("s", Trim(Res(i).start_time), Now) > 600) Then
            'Debug.Print DateDiff("s", Res(i).start_time, Now)
            Res(i).Client = ""
            Res(i).Status = False
            Res(i).Operational = False
            Res(i).start_time = ""
            Res(i).Tries = Res(i).Tries + 1
            'ActionQ(ActionPointer) = "verify!" & Trim(Str(i))
            'IncActionPointer
         End If
      End If
   Next i
End Sub

Private Sub rescan_timer_Timer()
Dim i As Integer
   For i = 0 To Total_Resource
      If Not Res(i).Operational Then
         ActionQ(ActionPointer) = "verify!" & Trim(Str(i))
         IncActionPointer
      End If
   Next i

End Sub

Private Sub SMTP1_EndTransfer()
'SMTP1.Action = 2
End Sub


Private Sub Timer_Timer()

Dim MyArray() As String
Dim i As Integer
Dim err_msg As String
   Timer.Enabled = False
   On Error GoTo timer_error
   If CurrentPointer <> ActionPointer Then
      i = ParseString(ActionQ(CurrentPointer), "!", MyArray)
      Select Case MyArray(1)
         Case "status"
            'Call NTService1.LogEvent(svcEventInformation, svcMessageInfo, "status")
            err_msg = "status"
            Call ProcessStatus(Val(MyArray(2)))
         Case "rescan"
```

```
            'Call NTService1.LogEvent(svcEventInformation, svcMessageInfo, "rescan")
            err_msg = "rescan"
            Call setup
            Call ProcessStatus(Val(MyArray(2)))
         Case "emergency_reset"
            For i = 0 To Total_Resource
               Res(i).Status = False
               Res(i).Client = ""
            Next
            err_msg = "emergency_reset"
            'Call NTService1.LogEvent(svcEventInformation, svcMessageInfo,
"emergency_reset")
            Call setup
            Call ProcessStatus(Val(MyArray(2)))
         Case "password"
            err_msg = "password"
            'Call NTService1.LogEvent(svcEventInformation, svcMessageInfo, "password")
            Call get_password(Val(MyArray(2)))
         Case "old"
            err_msg = "set type"
            Call set_type("old", Val(MyArray(2)))
         Case "new"
            err_msg = "set type"
            Call set_type("new", Val(MyArray(2)))
         Case "release"
            err_msg = "release"
            'Call NTService1.LogEvent(svcEventInformation, svcMessageInfo, "release")
            Call ReleaseRes(MyArray(3))
         Case "verify"
            err_msg = "verify"
            'Call NTService1.LogEvent(svcEventInformation, svcMessageInfo, "verify")
            Call Verify(Val(MyArray(2)))
         'Case "releaseall"
         '   Call ReleaseAll(Val)
      End Select
      'Debug.Print MyArray(1)
      IncCurrentPointer
   End If
Timer.Enabled = True
Exit Sub

timer_error:
   err_msg = err_msg & ": Timer Error " & Err.Number & "  " & Err.Description
   'Call NTService1.LogEvent(svcMessageError, svcEventError, err_msg)
   logMsg (err_msg)
   'Debug.Print "General Timer Error"
```

```
   Timer.Enabled = True
End Sub

Private Sub Timer1_Timer()
   Timer1.Enabled = False
   completedLogin = False
   testLogin = False
End Sub



" File Worker.bas
Attribute VB_Name = "Worker"

Public Sub getReady()

   'load the port number to listen to from the monitor.ini file
   in_sock_read = False
   read_level = 0
   Busy = False
   MustStop = False
   ActionPoiner = 0
   CurrentPointer = 0
   a$ = App.Path & "\monitor.ini"
   priviniregister "machine", a$
   B$ = "not_found_ini"
   LPort$ = PrivGetString("port", B$)  'Get port number to listen on
   'see if ini file exists
   If LPort$ = B$ Then
     If Main_frm.NTService1.Interactive Then
       MsgBox "The file monitor.ini was not found.", vbOKOnly + vbCritical + vbDefaultButton1
+ _
         vbApplicationModal, "No monitor.ini"
       Main_frm.NTService1.StopService
       Exit Sub
     Else
       Call NTService1.LogEvent(svcMessageError, svcEventError, "Monitor.ini not found")
       Main_frm.NTService1.StopService
       Exit Sub
     End If
   End If
   priviniregister "general", a$
   numClients = PrivGetString("numclients", B$) 'Get # clients
   numServers = PrivGetString("numservers", B$) 'Get # servers
   numPorts = PrivGetString("numports", B$) 'Get # ports
   numUsers = PrivGetString("numusers", B$) 'Get # users
```

```
    priviniregister "mail", a$
    MailServer = PrivGetString("server", B$) 'Get # mail Server
    MailTo = PrivGetString("mailto", B$) 'Get # mail to address

Main_frm.SMTP1.WinsockLoaded = True
Main_frm.SMTP1.MailServer = MailServer
Main_frm.SMTP1.To = MailTo
Main_frm.SMTP1.From = MailTo
Main_frm.SMTP1.Subject = "Server Failure notice"

    priviniregister "clients", a$
    For i = 1 To numClients
        Clients(i) = PrivGetString(Trim(Str(i)), B$) 'Get valid clients
    Next i

    Main_frm.Timer1.Enabled = False
    l = -1
    'Loop and fill resource array
    For i = 1 To numUsers
       priviniregister "usernames", a$
       uname = PrivGetString(Trim(Str(i)), B$)
       For j = 1 To numPorts
          priviniregister "ports", a$
          Port = PrivGetString(Trim(Str(j)), B$)
          For k = 1 To numServers
             priviniregister "servers", a$
             server = PrivGetString(Trim(Str(k)), B$)
             priviniregister "server_type", a$
             server_typ = PrivGetString(Trim(Str(k)), B$)
             l = l + 1
             Res(l).Machine = server
             Res(l).Status = False
             Res(l).Operational = False
             Res(l).Port = Port
             Res(l).User = uname
             Res(l).Client = ""
             Res(l).Tries = 0
             Res(l).Type = server_typ
          Next k
       Next j
    Next i

    Total_Resource = l
    HostConnected = False
    'Call setup subroutine to validate all servers
    Call setup
```

```
    Main_frm.rescan_timer.Interval = 30000 "'RESET TIMER INTEREVAL TO 60000
    Main_frm.rescan_timer.Enabled = True
    B$ = ""


On Error GoTo no_connect
    'start the monitor listening to the selected port
    Main_frm.IPDaemon1.WinsockLoaded = True
    Main_frm.IPDaemon1.KeepAlive = True
    Main_frm.IPDaemon1.Linger = True
    Main_frm.IPDaemon1.LocalPort = Val(LPort$)
    Main_frm.IPDaemon1.Listening = True
    'Timer1.Interval = 2000
    'Timer1.Enabled = True
    Exit Sub


no_connect:
    If Main_frm.NTService1.Interactive Then
       MsgBox "Error establishing a port to listen." + Chr$(10) + _
      "Cannot connect to port " & LPort$, vbOKOnly + vbCritical + _
      vbDefaultButton1 + vbApplicationModal, "Fatal Connection Error"
       Main_frm.NTService1.StopService
       Exit Sub
      Else
       Call NTService1.LogEvent(svcMessageError, svcEventError, "Fatal Connection Error")
       Main_frm.NTService1.StopService
       Exit Sub
      End If
End Sub

Public Sub Verify(i As Integer)
    'Subroutine to validate all resources
    testLogin = True 'Connect wait loop breakup flag
    completedLogin = False 'Success Flag
    'wait until the connection is achieved
    '(timeout in 10 seconds)
    'If Res(i).Tries > 3 Then
    '    Exit Sub
    'End If
'Main_frm.rescan_timer.Enabled = False
'Debug.Print "Doing Verification"
    Main_frm.Timer1.Interval = 2500 'Time out interval
    testLogin = True
    Main_frm.IPPort1.WinsockLoaded = True
```

```
    Main_frm.IPPort1.EOL = Chr(10)
    Main_frm.IPPort1.RemotePort = Res(i).Port
    Main_frm.IPPort1.RemoteHost = Res(i).Machine
    ' IPPort1.RemoteHost = "127.0.0.1"
    Main_frm.IPPort1.Linger = True
    Main_frm.IPPort1.KeepAlive = True
    Main_frm.IPPort1.Connected = True
    Main_frm.Timer1.Enabled = True
    Do While testLogin 'Wait for connect or time out
       DoEvents
    Loop

    If completedLogin Then 'If successfull connect
      completedLogin = False
      Res(i).Operational = Login(i) 'Check for login success
    Else
      Res(i).Operational = False

    End If
    Main_frm.IPPort1.Connected = False 'disconnect
    Do While Main_frm.IPPort1.Connected
      DoEvents
    Loop

    If Res(i).Operational Then
       Res(i).Operational = True
       Res(i).Tries = 0
    Else
       Res(i).Tries = Res(i).Tries + 1
       If Res(i).Tries = 4 Then
          MailMe i
       End If
       If Res(i).Tries > 32000 Then
          Res(i).Tries = 1
       End If
    End If
'Main_frm.rescan_timer.Enabled = True
End Sub
Public Sub IncActionPointer()
'Increment Action pointer ( Filling pointer) Taken care to reset back to 0 if it reaches
'255
ActionPointer = ActionPointer + 1
If ActionPointer = 256 Then
   ActionPointer = 0
End If
End Sub
```

```
Public Sub IncCurrentPointer()
'Increment current pointer ( processor reading pointer) Taken care to reset back to
'0 if it reaches 255
CurrentPointer = CurrentPointer + 1
If CurrentPointer = 256 Then
   CurrentPointer = 0
End If
End Sub

'ParseString
'ParseString will take a string and divide it into sections using a
'delimiter you specify.  The sections will be returned in an array you
'provide.  ParseString will also return the number of sections within the
'string.
'Pass it the string you want to parse, the delimiter separating the
'sections, and a string array in which the string sections will be
'returned.

Public Function ParseString(strSource As String, strDelim As String, strArray() As String)
Dim intElementCnt As Integer
Dim intCurPos As Integer
Dim intStrLen As Integer

intElementCnt = 1
intCurPos = 1
intStrLen = Len(strSource)

Do
   ReDim Preserve strArray(1 To intElementCnt)
   intStrLen = (InStr(intCurPos, strSource, strDelim) - intCurPos)
   If intStrLen < 0 Then
      strArray(intElementCnt) = Right$(strSource, (Len(strSource) - (intCurPos - 1)))
   Else
      strArray(intElementCnt) = Mid$(strSource, intCurPos, intStrLen)
      intCurPos = intCurPos + (Len(strArray(intElementCnt)) + Len(strDelim))
      intElementCnt = intElementCnt + 1
   End If
Loop Until intStrLen < 0
ParseString = UBound(strArray)
End Function
Public Sub ProcessStatus(ConnectionID As Integer)
'Sends back the status of resources to the client specified by the connection ID
Dim a$
send2Client "**status", ConnectionID
'a$ = "ID" & vbTab & "Machine        Status  Operational Port User Client Tries"
'send2Client a$, ConnectionId
```

```
For i = 0 To Total_Resource
   a$ = Trim(Str(i + 1)) & "," & Trim(Res(i).Machine) & "," & Trim(Res(i).Status) & "," & _
   Trim(Res(i).Operational) & "," & Trim(Res(i).Port) & "," & Trim(Res(i).User) & "," & _
   Trim(Res(i).Client) & "," & Trim(Res(i).Tries)
   send2Client a$, ConnectionID
Next i
send2Client "@@done", ConnectionID
End Sub

Public Sub logMsg(Msg As String)
   Print #1, Msg & vbCrLf
End Sub
Public Sub set_type(typ As String, ConnectionID As Integer)
Server_Type(ConnectionID) = Trim(typ)
End Sub
Public Sub get_password(ConnectionID As Integer)
'Look for the next avaliable free resource and send it to the client ID'd by Connection ID
'But got to test for the validity of the resource b4 sending it.
'Then mark the resource ststus as used
'IF no more resource is free it sends "**error"
Dim i As Integer
   For i = 0 To Total_Resource
      If Server_Type(ConnectionID) = "old" Then
         If Res(i).Operational And Not Res(i).Status And Res(i).Type = "old" Then
            Main_frm.Timer1.Interval = 10000
            testLogin = True
            Main_frm.IPPort1.WinsockLoaded = True
            Main_frm.IPPort1.EOL = Chr(10)
            Main_frm.IPPort1.RemotePort = Res(i).Port
            Main_frm.IPPort1.RemoteHost = Res(i).Machine
           ' IPPort1.RemoteHost = "127.0.0.1"
            Main_frm.IPPort1.Linger = True
            Main_frm.IPPort1.KeepAlive = True
            Main_frm.IPPort1.Connected = True
            Main_frm.Timer1.Enabled = True
            Do While testLogin
              DoEvents
            Loop
            If completedLogin Then
              completedLogin = False
              If Login(i) Then
                 Main_frm.IPPort1.Connected = False
                 Do While Main_frm.IPPort1.Connected
                    DoEvents
                 Loop
```

```
                a$ = Trim(Str$(i)) & "," & Trim(Res(i).Machine) & "," & Trim(Str$(Res(i).Port))
& "," & _
                Trim(Res(i).User)
                send2Client "**password", ConnectionID
                send2Client a$, ConnectionID
                Res(i).Status = True
                Res(i).Client = Main_frm.IPDaemon1.RemoteHost(ConnectionID)
                Res(i).start_time = Now
                Exit Sub
              Else
                Res(i).Operational = False
                Res(i).Tries = 1
              End If
            Else
              Res(i).Operational = False
              Res(i).Tries = 1
            End If
            Main_frm.IPPort1.Connected = False
            Do While Main_frm.IPPort1.Connected
              DoEvents
            Loop
          End If
        Else
          If Res(i).Operational And Not Res(i).Status And Res(i).Type = "new" Then
            Main_frm.Timer1.Interval = 10000
            testLogin = True
            Main_frm.IPPort1.WinsockLoaded = True
            Main_frm.IPPort1.EOL = Chr(10)
            Main_frm.IPPort1.RemotePort = Res(i).Port
            Main_frm.IPPort1.RemoteHost = Res(i).Machine
           ' IPPort1.RemoteHost = "127.0.0.1"
            Main_frm.IPPort1.Linger = True
            Main_frm.IPPort1.KeepAlive = True
            Main_frm.IPPort1.Connected = True
            Main_frm.Timer1.Enabled = True
            Do While testLogin
              DoEvents
            Loop
            If completedLogin Then
              completedLogin = False
              If Login(i) Then
                Main_frm.IPPort1.Connected = False
                Do While Main_frm.IPPort1.Connected
                  DoEvents
                Loop
```

```
                a$ = Trim(Str$(i)) & "," & Trim(Res(i).Machine) & "," & Trim(Str$(Res(i).Port))
& "," & _
                Trim(Res(i).User)
                send2Client "**password", ConnectionID
                send2Client a$, ConnectionID
                Res(i).Status = True
                Res(i).Client = Main_frm.IPDaemon1.RemoteHost(ConnectionID)
                Res(i).start_time = Now
                Exit Sub
             Else
                Res(i).Operational = False
                Res(i).Tries = 1
             End If
          Else
             Res(i).Operational = False
             Res(i).Tries = 1
          End If
          Main_frm.IPPort1.Connected = False
          Do While Main_frm.IPPort1.Connected
             DoEvents
          Loop
       End If
    End If
  Next i
a$ = "**error"
send2Client "**password", ConnectionID
send2Client a$, ConnectionID
End Sub

Public Sub ReleaseRes(a$)
'releases resources identified by the comma seperated string varaible
Dim MyArray() As String
i = ParseString(a$, ",", MyArray())
For j = 1 To i
  If Val(MyArray(j)) > -1 Then
    Res(Val(MyArray(j))).Status = False
    Res(Val(MyArray(j))).Client = ""
    Res(Val(MyArray(j))).start_time = ""
  End If
Next j
End Sub
Public Sub setup()
'Subroutine to validate all resources
testLogin = True 'Connect wait loop breakup flag
completedLogin = False 'Success Flag
Dim i As Integer
```

```
For i = 0 To Total_Resource
  'wait until the connection is achieved
  '(timeout in 10 seconds)

  Main_frm.Timer1.Interval = 10000 'Time out interval

  testLogin = True
  Main_frm.IPPort1.WinsockLoaded = True
  Main_frm.IPPort1.EOL = Chr(10)
  Main_frm.IPPort1.RemotePort = Res(i).Port
  Main_frm.IPPort1.RemoteHost = Res(i).Machine
 ' IPPort1.RemoteHost = "127.0.0.1"
  Main_frm.IPPort1.Linger = True
  Main_frm.IPPort1.KeepAlive = True
  Main_frm.IPPort1.Connected = True
  Main_frm.Timer1.Enabled = True 'Start Connection time out timer
  Do While testLogin 'Wait for connect or time out
    DoEvents
  Loop

  If completedLogin Then 'If successfull connect
    completedLogin = False
    Res(i).Operational = Login(i) 'Check for login success
  Else
    Res(i).Operational = False
  End If
  Main_frm.IPPort1.Connected = False 'disconnect
  Do While Main_frm.IPPort1.Connected
    DoEvents
  Loop
Next i
End Sub

Public Function Login(i As Integer) As Boolean
'tests resource i (res(i)) with a valid user id. And returns a boolean for success / Failure
    completedLogin = False
    testLogin = True
 'request to log on to server
    Call send2wsk("logon.")
    a$ = Trim(Res(i).User) + "."
    Call send2wsk(a$)
    Main_frm.Timer1.Interval = 10000
    Main_frm.Timer1.Enabled = True
    Do While testLogin
       DoEvents
    Loop
```

```
   Login = completedLogin
End Function
Public Sub send2wsk(a$)
   'send a line to socket
   On Error GoTo snd_error
   a$ = a$ + Main_frm.IPPort1.EOL
   Main_frm.IPPort1.DataToSend = a$
   Exit Sub

snd_error:
      If Err = 25036 Then              'would block
      BytesSent = Main_frm.IPPort1.BytesSent
      If BytesSent > 0 Then          'was any sent?
         a$ = Mid$(a$, BytesSent + 1)  'send rest
      End If
      DoEvents       'wait a little while - important!
      Resume         'go back and try again!
  Else               'handle other errors here (this has not been tested)
      Main_frm.IPPort1.Connected = False
      On Error GoTo 0
      Exit Sub
  End If
End Sub
Public Sub send2Client(a$, ConnectionID As Integer)
   'send a line to socket
   On Error GoTo snd_error
   a$ = a$ + Main_frm.IPDaemon1.EOL(ConnectionID)
   Main_frm.IPDaemon1.DataToSend(ConnectionID) = a$
   Exit Sub

snd_error:
      If Err = 25036 Then              'would block
      BytesSent = Main_frm.IPDaemon1.BytesSent(ConnectionID)
      If BytesSent > 0 Then          'was any sent?
         a$ = Mid$(a$, BytesSent + 1)  'send rest
      End If
      DoEvents       'wait a little while - important!
      Resume         'go back and try again!
  Else               'handle other errors here (this has not been tested)
      Main_frm.IPDaemon1.Connected(ConnectionID) = False
      On Error GoTo 0
      Exit Sub
  End If
End Sub
Public Sub parse_sock(a$, Msg$)
   'here we just started to receive a
```

```
    'message from the host The first line
    'is always **FUNCTION

    Static B$, nowlen%
    'Debug.Print a$ & "    " & Msg$
    Select Case a$



      'logon to host request
      Case "**logon"
       'get hosts response
'       in_sock_read = False
       'check hosts response
       If Msg$ = "true" Then
          completedLogin = True
          testLogin = False
       Else
          completedLogin = False
          testLogin = False
       End If
       read_level = 0
       in_sock_read = False

       Case "**getoff"
       'get hosts response
'       in_sock_read = False
       'check hosts response
       completedLogin = False
       testLogin = False
       read_level = 0
       in_sock_read = False
    End Select

End Sub

Sub MailMe(i As Integer)

    a$ = Res(i).Machine & " ," & Res(i).Port & " ," & Res(i).User & vbCrLf
    Main_frm.SMTP1.MessageText = a$
    Main_frm.SMTP1.Action = 3
End Sub

" File iniset.bas
Attribute VB_Name = "INISET"
Global Total_Resource As Integer 'Total number of resources
```

```
Public Type resources  'type structure used for storing resource information
      Machine As String * 16
      Status As Boolean
      Operational As Boolean
      Port As Integer
      User As String * 16
      Client As String * 16
      Tries As Integer
      Type As String
      start_time As String
End Type
Global Clients(10) As String ' Holding valid client names read from ini file
Global Res(25) As resources   ' Resource array
Global numClients As Integer ' Total # clients
Global numServers As Integer ' Total # servers
Global numPorts As Integer   ' Total # ports
Global numUsers As Integer   ' Total # users
Global testLogin As Boolean  ' Flag for breaking out of waiting loop for connecting
Global completedLogin As Boolean ' Flag indicating that a succesfull login was done
Global in_sock_read As Boolean  ' Falg for indicating that a sock read in in progress
Global read_level As Integer    ' Variable for holding the line # for the lines read from sock read
Global Busy As Boolean ' Resource array locking variable
Global ActionQ(256) As String
Global ActionPointer As Integer
Global CurrentPointer As Integer
Global MailServer As String
Global MailTo As String
Global MustStop As Boolean
Global Server_Type(256) As String

'** Windows API calls
'(NOTE: Profile calls *altered* from those found in WIN30API.TXT!)
  Declare Function kpGetPrivateProfileString Lib "kernel32" Alias "GetPrivateProfileStringA"
(ByVal lpApplicationName As String, ByVal lpKeyName As Any, ByVal lpDefault As String,
ByVal lpReturnedString As String, ByVal nSize As Long, ByVal lpFileName As String) As
Long
  Declare Function kpWritePrivateProfileString Lib "kernel32" Alias
"WritePrivateProfileStringA" (ByVal lpApplicationName As String, ByVal lpKeyName As Any,
ByVal lpString As Any, ByVal lpFileName As String) As Long
  Public Declare Sub Sleep Lib "kernel32" (ByVal dwMilliseconds As Long)
'** Module-level variables for [Section] and Ini file names
Dim smSectionName As String    'Current section in private Ini file
Dim smIniFileName As String     'Fully qualified path/name of current private Ini file
Dim nmPrivInit As Integer          'Flag to indicate that Private.Ini is initialized

'** Constants used to size buffers
```

```
Const Max_SectionBuffer = 4096
Const Max_EntryBuffer = 255


Public Sub privdeleteentry(sEntryName As String)

  'Bail if not initialized
    If Not nmPrivInit Then
      PrivIniNotReg
      Exit Sub
    End If

  'Deletes a specific entry in Private.Ini
    Dim nRetVal
    nRetVal = kpWritePrivateProfileString(smSectionName, sEntryName, 0&, smIniFileName)

End Sub

Public Sub PrivDeleteSection()

  'Bail if not initialized
    If Not nmPrivInit Then
      PrivIniNotReg
      Exit Sub
    End If

  'Deletes an *entire* [Section] and all its Entries in Private.Ini
    Dim nRetVal
    nRetVal = kpWritePrivateProfileString(smSectionName, 0&, 0&, smIniFileName)

  'Now Private.Ini needs to be reinitialized
    smSectionName = ""
    nmPrivInit = False

End Sub

Public Function PrivGetSectEntries() As String

  'Bail if not initialized
    If Not nmPrivInit Then
      PrivIniNotReg
      Exit Function
    End If

  'Retrieves all Entries in a [Section] of Private.Ini
  'Entries nul terminated; last entry double-terminated
```

```
    Dim sTemp As String * Max_SectionBuffer
    Dim nRetVal
    nRetVal = kpGetPrivateProfileString(smSectionName, 0&, "", sTemp, Len(sTemp),
smIniFileName)
    PrivGetSectEntries$ = Left$(sTemp, nRetVal + 1)

End Function

Public Function PrivGetString(sEntryName As String, ByVal sDefaultValue As String) As
String

  'Bail if not initialized
    If Not nmPrivInit Then
      PrivIniNotReg
      Exit Function
    End If

  'Retrieves Specific Entry from Private.Ini
    Dim sTemp As String * Max_EntryBuffer
    Dim nRetVal
    nRetVal = kpGetPrivateProfileString(smSectionName, sEntryName, sDefaultValue, sTemp,
Len(sTemp), smIniFileName)
    If nRetVal Then
      PrivGetString = Left$(sTemp, nRetVal)
    End If

End Function

Private Sub PrivIniNotReg()

  'Warn that there's a logic error!
    MsgBox "[Section] and FileName Not Registered in Private.Ini!", 16, "IniFile Logic Error"

End Sub

Public Sub priviniregister(sSectionName As String, sIniFileName As String)

  'Store module-level values for future reference
    smSectionName = Trim$(sSectionName)
    smIniFileName = Trim$(sIniFileName)
    nmPrivInit = True

End Sub

Public Function privputstring(sEntryName As String, ByVal sValue As String) As Integer
```

```
  'Bail if not initialized
    If Not nmPrivInit Then
      PrivIniNotReg
      Exit Function
    End If
  Dim temp
  'Write a string to Private.Ini
    temp = kpWritePrivateProfileString(smSectionName, sEntryName, sValue, smIniFileName)
    privputstring = temp
End Function

"Client code
" File Client.vbp
Type=Exe
Form=Form1.frm
Reference=*\G{00020430-0000-0000-C000-
000000000046}#2.0#0#C:\WINNT\System32\stdole2.tlb#OLE Automation
Object={33335113-F789-11CE-86F8-0020AFD8C6DB}#2.0#0; IPPORT35.Ocx
IconForm="Form1"
Startup="Form1"
ExeName32="Client.exe"
Command32=""
Name="Client"
HelpContextID="0"
CompatibleMode="0"
MajorVer=1
MinorVer=0
RevisionVer=0
AutoIncrementVer=0
ServerSupportFiles=0
VersionCompanyName="UGA"
CompilationType=0
OptimizationType=0
FavorPentiumPro(tm)=0
CodeViewDebugInfo=0
NoAliasing=0
BoundsCheck=0
OverflowCheck=0
FlPointCheck=0
FDIVCheck=0
UnroundedFP=0
StartMode=0
Unattended=0
Retained=0
ThreadPerObject=0
MaxNumberOfThreads=1
```

```
[MS Transaction Server]
AutoRefresh=1

" File Form1.frm
VERSION 5.00
Object = "{33335113-F789-11CE-86F8-0020AFD8C6DB}#2.0#0"; "IPPORT35.Ocx"
Begin VB.Form Form1
  Caption        =   "Client for Server Manager"
  ClientHeight   =   6600
  ClientLeft     =   60
  ClientTop      =   345
  ClientWidth    =   7575
  Icon           =   "Form1.frx":0000
  LinkTopic      =   "Form1"
  ScaleHeight    =   6600
  ScaleWidth     =   7575
  StartUpPosition =   3  'Windows Default
  Begin VB.TextBox Text3
    Height         =   375
    Left           =   4680
    TabIndex       =   13
    Text           =   "Text3"
    Top            =   5160
    Width          =   2295
  End
  Begin VB.TextBox txtMinTime
    Height         =   495
    Left           =   4680
    TabIndex       =   12
    Text           =   "100"
    Top            =   5760
    Width          =   2295
  End
  Begin VB.Timer BangTimer
    Enabled        =   0   'False
    Interval       =   1500
    Left           =   6000
    Top            =   4200
  End
  Begin VB.CommandButton cmdBang
    Caption        =   "Ping Server Manager"
    Height         =   855
    Left           =   5400
    TabIndex       =   11
    Top            =   3240
```

```
   Width          =   2055
End
Begin VB.TextBox Text2
   Height         =   285
   Left           =   1440
   TabIndex       =   9
   Text           =   "ibmlc5"
   Top            =   6120
   Width          =   1935
End
Begin VB.CheckBox Check1
   Caption        =   "OLD"
   Height         =   375
   Left           =   2040
   TabIndex       =   8
   Top            =   5520
   Width          =   1815
End
Begin VB.Timer Timer1
   Left           =   5040
   Top            =   4200
End
Begin VB.CommandButton cmdRelease
   Caption        =   "Release"
   Height         =   495
   Left           =   1920
   TabIndex       =   6
   Top            =   4800
   Width          =   2415
End
Begin VB.CommandButton cmdLogOff
   Caption        =   "LogOff"
   Height         =   855
   Left           =   360
   TabIndex       =   5
   Top            =   5040
   Width          =   1095
End
Begin VB.CommandButton cmdPassword
   Caption        =   "Password"
   Height         =   855
   Left           =   360
   TabIndex       =   4
   Top            =   4080
   Width          =   1095
End
```

```
Begin VB.CommandButton Command3
   Caption         =   "Rescan"
   Height          =   495
   Left            =   1920
   TabIndex        =   3
   Top             =   4080
   Width           =   2415
End
Begin VB.CommandButton Command2
   Caption         =   "Login"
   Height          =   855
   Left            =   360
   TabIndex        =   2
   Top             =   3120
   Width           =   1095
End
Begin VB.TextBox Text1
   BeginProperty Font
      Name            =   "Courier New"
      Size            =   9
      Charset         =   0
      Weight          =   400
      Underline       =   0   'False
      Italic          =   0   'False
      Strikethrough   =   0   'False
   EndProperty
   Height          =   2295
   Left            =   120
   MultiLine       =   -1  'True
   TabIndex        =   1
   Top             =   600
   Width           =   7335
End
Begin VB.CommandButton Command1
   Caption         =   "Status"
   Height          =   495
   Left            =   1920
   TabIndex        =   0
   Top             =   3240
   Width           =   2415
End
Begin VB.Label Label2
   AutoSize        =   -1  'True
   Caption         =   "Use Server:"
   Height          =   195
   Left            =   480
```

```
    TabIndex        =   10
    Top             =   6120
    Width           =   840
  End
  Begin VB.Label Label1
    Caption         =   "Just started. Not logged on."
    Height          =   495
    Left            =   360
    TabIndex        =   7
    Top             =   0
    Width           =   5175
  End
  Begin IPPortLib.IPPort IPPort1
    Left            =   4680
    Top             =   3360
    _ExtentX        =   741
    _ExtentY        =   741
    RemoteHost      =   ""
    Linger          =   -1  'True
    EOL             =   ""
  End
End
Attribute VB_Name = "Form1"
Attribute VB_GlobalNameSpace = False
Attribute VB_Creatable = False
Attribute VB_PredeclaredId = True
Attribute VB_Exposed = False
Dim in_sock_read As Boolean
Dim read_level As Integer
Dim timer_wait As Boolean
Dim connections(25) As Integer
Dim ConnectionNum As Integer
'Const RemoteMachine = "ibmlc5.chem.uga.edu"
Dim im_logged_in As Boolean
Dim gotPassword As Boolean
Dim minTime As Integer



Private Sub BangTimer_Timer()

   If Not gotPassword Then
      BangTimer.Enabled = False
      Call cmdPassword_Click
      gotPassword = True
      BangTimer.Enabled = True
```

```
    Else
        BangTimer.Enabled = False
        Call resettimer
        Call cmdRelease_Click
        gotPassword = False
        BangTimer.Enabled = True
    End If
End Sub
Sub resettimer()
BangTimer.Interval = Int((3500 - minTime + 1) * Rnd + minTime)
Text3.Text = Str(BangTimer.Interval)
End Sub
Private Sub cmdBang_Click()
BangTimer.Enabled = Not BangTimer.Enabled
End Sub

Private Sub cmdLogOff_Click()
Text1.Text = ""
If im_logged_in Then
    Call send2wsk("logoff")
End If
IPPort1.Connected = False
im_logged_in = False
Beep
End Sub

Private Sub cmdPassword_Click()
Text1.Text = ""
If im_logged_in Then
    If Check1.Value Then
        Call send2wsk("old")
    Else
        Call send2wsk("new")
    End If
    Call send2wsk("password")
Else
    Beep
    MsgBox "You are not logged in."
End If
End Sub

Private Sub cmdRelease_Click()
Dim i As Integer
Text1.Text = ""
If im_logged_in Then
    a$ = "release,"
```

```
    If ConnectionNum < 1 Then
       Text1.Text = "No resources to release"
       Exit Sub
    End If

    For i = 1 To ConnectionNum
       a$ = a$ & Str$(connections(i))
       If i <> ConnectionNum Then
          a$ = a$ & ","
       End If
    Next i
    ConnectionNum = 0
    Call send2wsk(a$)
    Text1.Text = "Resources released"
Else
    Beep
    MsgBox "You are not logged in."
End If


End Sub

Private Sub Command1_Click()
Text1.Text = ""
    If im_logged_in Then
       Call send2wsk("status")
    Else
       Beep
       MsgBox "You are not logged in."
    End If

End Sub

Private Sub Command2_Click()
    Text1.Text = ""
'Login call
    If im_logged_in Then
       MsgBox "You are already logged in."
       Exit Sub
    End If
    RemoteMachine = Trim(Text2.Text)
    Timer1.Interval = 5000
    Timer1.Enabled = True
    timer_wait = True
    IPPort1.WinsockLoaded = True
    IPPort1.EOL = Chr(10)
```

```
    IPPort1.RemoteHost = RemoteMachine
    IPPort1.RemotePort = 1250
    IPPort1.Linger = True
    IPPort1.KeepAlive = True
    IPPort1.Connected = True
    Do While timer_wait
       DoEvents
    Loop
    If IPPort1.Connected Then
       im_logged_in = True
       Label1.Caption = "Logged into Server Manager at " & RemoteMachine
    Else
       im_logged_in = False
       Label1.Caption = "Failed to log into Server Manager at " & RemoteMachine
    End If
    Beep
End Sub

Private Sub Command3_Click()
    Text1.Text = ""
    If im_logged_in Then
       Call send2wsk("rescan")
    Else
       Beep
       MsgBox "You are not logged in."
    End If

End Sub

Private Sub Form_Load()
    im_logged_in = False
    gotPassword = False
    minTime = Val(txtMinTime.Text)
End Sub

Private Sub IPPort1_Connected(StatusCode As Integer, Description As String)
If StatusCode = 0 Then
    timer_wait = False
End If

End Sub

Private Sub IPPort1_DataIn(Text As String, EOL As Integer)
   Static in_proc$, c$, flag%
    If EOL = False Then
      c$ = c$ + Text
```

```
      flag% = True
      Exit Sub
    End If

    If EOL And flag% Then
       a$ = c$ + Text
       c$ = ""
       flag% = False
    Else
       a$ = Text
    End If
    If a$ <> "" Then
     If Asc(Right$(a$, 1)) = 13 Then
      a$ = Mid$(a$, 1, Len(a$) - 1)
     End If
    End If

    safe$ = in_proc$

    If in_sock_read Then
       Call parse_sock(safe$, a$)
    Else
       in_proc$ = LTrim$(RTrim$(a$))
       in_sock_read = True
       read_level = 0
    End If

End Sub

Sub send2wsk(a$)
   'send a line to socket
   On Error GoTo snd_error
   a$ = a$ + IPPort1.EOL
   IPPort1.DataToSend = a$
   Exit Sub

snd_error:
      If Err = 25036 Then               'would block
      BytesSent = IPPort1.BytesSent
      If BytesSent > 0 Then          'was any sent?
         a$ = Mid$(a$, BytesSent + 1)  'send rest
      End If
      DoEvents       'wait a little while - important!
      Resume         'go back and try again!
   Else                'handle other errors here (this has not been tested)
      IPPort1.Connected = False
```

```
      On Error GoTo 0
      Exit Sub
   End If
End Sub
Sub parse_sock(a$, msg$)
   'here we just started to receive a
   'message from the host The first line
   'is always **FUNCTION
Dim MyArray() As String
   Static B$, nowlen%

   Select Case a$
     'logon to host request
      Case "**status"
       'get hosts response
'       in_sock_read = False
       'check hosts response
'       Debug.Print Msg$
          If InStr(msg$, "@@done") Then
             read_level = 0
             in_sock_read = False
             Text1.Text = B$
             Exit Sub
          End If
          If read_level = 0 Then
             B$ = "ID " & "CalcServer        " & "Used " & " Avail" & " Port " & "User      " &
"SrvManager  " & "    Try"

'1 128.192.5.71 False True 1203 browser1 ResServer Try
             B$ = B$ & process_msg(msg$)
             read_level = 1
          Else
             B$ = B$ & process_msg(msg$)

          End If
       'request for password
       Case "**password"
          If UCase$(msg$) = "**ERROR" Then
             Text1.Text = "Error. Resources exhausted"
             read_level = 0
             in_sock_read = False
          Else
             numL = ParseString(msg$, ",", MyArray())
             a$ = "ConnectionID = " & MyArray(1) & vbCrLf
             a$ = a$ & "Resource = " & MyArray(2) & vbCrLf
             a$ = a$ & "Port = " & MyArray(3) & vbCrLf
```

```
            a$ = a$ & "User = " & MyArray(4)
            Text1.Text = a$
            ConnectionNum = ConnectionNum + 1
            connections(ConnectionNum) = Val(MyArray(1))
            read_level = 0
            in_sock_read = False
         End If
   End Select

End Sub

Private Sub IPPort1_Disconnected(StatusCode As Integer, Description As String)
Label1.Caption = "Logged off from Server Manager " & RemoteMachine
End Sub

Private Sub Timer1_Timer()
   timer_wait = False
End Sub


'ParseString
'ParseString will take a string and divide it into sections using a
'delimiter you specify.  The sections will be returned in an array you
'provide.  ParseString will also return the number of sections within the
'string.
'Pass it the string you want to parse, the delimiter separating the
'sections, and a string array in which the string sections will be
'returned.

Function ParseString(strSource As String, strDelim As String, strArray() As String)
Dim intElementCnt As Integer
Dim intCurPos As Integer
Dim intStrLen As Integer

intElementCnt = 1
intCurPos = 1
intStrLen = Len(strSource)

Do
   ReDim Preserve strArray(1 To intElementCnt)
   intStrLen = (InStr(intCurPos, strSource, strDelim) - intCurPos)
   If intStrLen < 0 Then
      strArray(intElementCnt) = Right$(strSource, (Len(strSource) - (intCurPos - 1)))
   Else
      strArray(intElementCnt) = Mid$(strSource, intCurPos, intStrLen)
      intCurPos = intCurPos + (Len(strArray(intElementCnt)) + Len(strDelim))
```

```
        intElementCnt = intElementCnt + 1
    End If
Loop Until intStrLen < 0
ParseString = UBound(strArray)
End Function


Public Function process_msg(msg As String) As String
    Dim myra() As String
    num = ParseString(msg, ",", myra())
    a$ = vbCrLf & Trim(myra(1)) & "  " 'id
    B$ = Trim(myra(2))
    a$ = a$ & B$ 'calcserver
    l = 16 - Len(B$)
    If l > 0 Then
        a$ = a$ & String(l, " ")
    End If
    B$ = Trim(myra(3)) 'inuse
    a$ = a$ & B$ & " "
    l = 5 - Len(B$)
    If l > 0 Then
        a$ = a$ & String(l, " ")
    End If
    B$ = Trim(myra(4)) 'avail
    a$ = a$ & B$ & " "
    l = 5 - Len(B$)
    If l > 0 Then
        a$ = a$ & String(l, " ")
    End If
    B$ = Trim(myra(5)) 'port
    a$ = a$ & B$ & " "
    l = 4 - Len(B$)
    If l > 0 Then
        a$ = a$ & String(l, " ")
    End If
    B$ = Trim(myra(6)) 'user
    a$ = a$ & B$ & " "
    l = 8 - Len(B$)
    If l > 0 Then
        a$ = a$ & String(l, " ")
    End If
    B$ = Trim(myra(7)) 'srvmanager
    a$ = a$ & B$
    l = 16 - Len(B$)
    If l > 0 Then
        a$ = a$ & String(l, " ")
```

```
    End If
    B$ = Trim(myra(8)) 'try
    a$ = a$ & B$

    process_msg = a$
End Function

Private Sub txtMinTime_Change()
minTime = Val(txtMinTime.Text)
End Sub
```

Appendix B

Code Listing for SPARC Remote Training Program

```
" File RemoteTraining.vbp
Type=Exe
Module=GENERIC; GENERIC.BAS
Reference=*\G{00020430-0000-0000-C000-
000000000046}#2.0#0#C:\WINDOWS\System32\stdole2.tlb#Standard OLE Types
Object={2037E3AD-18D6-101C-8158-221E4B551F8E}#5.0#0; vsocx32.ocx
Form=multipletrain.frm
Form=multiplesetup.frm
Form=PasswdForm.frm
Object={33335113-F789-11CE-86F8-0020AFD8C6DB}#2.0#0; IPPORT35.Ocx
Object={33335173-F789-11CE-86F8-0020AFD8C6DB}#2.0#0; FTP35.Ocx
Form=newparam.frm
Form=ftpfrm.frm
Object={F9043C88-F6F2-101A-A3C9-08002B2F49FB}#1.2#0; comdlg32.ocx
Form=Multipliers.frm
Startup="MultipleSetup"
HelpFile=""
ExeName32="remote_training.exe"
Command32=""
Name="RemoteTrainig"
HelpContextID="0"
CompatibleMode="0"
MajorVer=1
MinorVer=0
RevisionVer=0
AutoIncrementVer=0
ServerSupportFiles=0
VersionCompanyName="UGA"
CompilationType=0
OptimizationType=0
FavorPentiumPro(tm)=0
CodeViewDebugInfo=0
NoAliasing=0
BoundsCheck=0
OverflowCheck=0
FlPointCheck=0
FDIVCheck=0
UnroundedFP=0
StartMode=0
Unattended=0
```

```
Retained=0
ThreadPerObject=0
MaxNumberOfThreads=1

[MS Transaction Server]
AutoRefresh=1

" File ftpfrm.frm
VERSION 5.00
Object = "{33335173-F789-11CE-86F8-0020AFD8C6DB}#2.0#0"; "FTP35.Ocx"
Begin VB.Form ftpfrm
   Caption        =   "FTPfrm"
   ClientHeight   =   3135
   ClientLeft     =   60
   ClientTop      =   405
   ClientWidth    =   4680
   LinkTopic      =   "Form1"
   ScaleHeight    =   3135
   ScaleWidth     =   4680
   StartUpPosition =   3  'Windows Default
   Begin FTPLib.FTP FTP1
      Left           =   600
      Top            =   2160
      _ExtentX       =   741
      _ExtentY       =   741
      RemoteHost     =   ""
      User           =   ""
      Password       =   ""
      RemotePath     =   ""
      LocalFile      =   ""
      RemoteFile     =   ""
      AltFile        =   ""
   End
End
Attribute VB_Name = "ftpfrm"
Attribute VB_GlobalNameSpace = False
Attribute VB_Creatable = False
Attribute VB_PredeclaredId = True
Attribute VB_Exposed = False
'This is a general form for performing FTP file transfers
Dim transfer_done As Integer
Dim temp$
Dim the_ftp_string As String
Dim finished_ftp As Integer

'Main method
```

```
'  UpLoad true for upload false for download
'  localfile is full path to local file
'  remotefile is the remote file (can be but usually is not a path)
'  port to connect to on machine
'  user on remote machine
'  password for user
'  machine is the name of the remote machine
'  was_error set if there is a problem
'  errmsg is a description of what caused the error
Public Sub ftpFile(UpLoad As Boolean, Localfile As String, _
            Remotefile As String, Port As String, User As String, _
            Password As String, Machine As String, _
            was_error As Boolean, ErrMsg As String)

   was_error = False

   'Set up ocx params
   FTP1.WinsockLoaded = True
   FTP1.Remotefile = Remotefile
   FTP1.Localfile = Localfile
   FTP1.RemoteHost = Machine
   FTP1.RemotePort = Val(Port)
   FTP1.User = User
   FTP1.Password = Password
   FTP1.Passive = True
   FTP1.TransferMode = 1 'ASCII transfer used to handle line termination

   'Try to logon to remote machine
   On Error GoTo logfail
   'Logon
   FTP1.Action = 2

   If UpLoad Then
      'Uploading a file to the remote machine
      'Check to see if local file exists
      If Not FileExists(Localfile) Then
         was_error = True
         ErrMsg = "Local file does not exist"
         Exit Sub
      End If

      On Error GoTo E2
      transfer_done = False
      FTP1.Action = 5 'upload
      'Wait for completion of transfer
      Do While Not transfer_done
```

```
    DoEvents
  Loop
  Beep
  DoEvents
  On Error Resume Next
  FTP1.Action = 3 ' logoff
  DoEvents
  DoEvents
  DoEvents
  DoEvents
  DoEvents
  DoEvents
  DoEvents
  DoEvents
  DoEvents
  DoEvents
  DoEvents
  DoEvents
  DoEvents
Else
  'downloading a file.....................
  'directory list short
  the_ftp_string = ""
  FTP1.Action = 8
  Do While the_ftp_string = ""
    DoEvents
  Loop
  a$ = the_ftp_string
  On Error Resume Next
  'See if the file was there to be downloaded
  If InStr(a$, "No such") <> 0 Then
    was_error = True
    ErrMsg = "File not found on remote machine"
    Exit Sub
  End If

  'set local file name and kill it
  On Error Resume Next
  bb$ = Localfile
  Kill bb$

  'start download
  On Error GoTo E2
  transfer_done = False
  FTP1.Action = 4 'download
  'wait for download to complete
  Do While Not transfer_done
```

```
        DoEvents
      Loop
      Beep
      DoEvents
      DoEvents
      On Error Resume Next
      FTP1.Action = 3 ' logoff
      DoEvents
      DoEvents
      DoEvents
      DoEvents
      DoEvents
      DoEvents
      DoEvents
      DoEvents
      DoEvents
      DoEvents
      DoEvents
      DoEvents
   End If

Exit Sub

logfail:
   FTP1.Action = 3
   was_error = True
   ErrMsg = "Failed to logon with the supplied host/user/password"
   Exit Sub

E2:
   FTP1.Action = 3
   ErrMsg = "The file could not be transferred"
   was_error = True
   Exit Sub

End Sub


Private Sub FTP1_DirList(DirEntry As String)
   temp$ = DirEntry
End Sub

Private Sub FTP1_EndTransfer()
   transfer_done = True
   the_ftp_string = temp$
```

```
End Sub

Private Sub FTP1_Error(ErrorCode As Integer, Description As String)
MsgBox "The file retrieve function experienced an error." & Chr$(13) & Chr$(10) & "The error number is " & Trim$(str$(ErrorCode)) & Chr$(13) & Chr$(10) & "The error message is:" & Chr$(13) & Chr$(10) & Description & Chr$(13) & Chr$(10) & "" & Chr$(13) & Chr$(10) & "Please report this error to your SPARC manager.", 16, "Retrieve File Error"

End Sub



'FileExists
'The FileExists function returns a value of TRUE if the specified file exists, or FALSE if it doesn't.
'Pass it a string containing the file name to check.

Private Function FileExists(strFileName As String) As Boolean
Dim strReturn As String

On Error Resume Next
strReturn = Dir$(strFileName)
If LTrim(strFileName) = "" Or strReturn = "" Then
   FileExists = False
Else
   FileExists = True
End If
End Function



" File multiplesetup.frm
VERSION 5.00
Object = "{2037E3AD-18D6-101C-8158-221E4B551F8E}#5.0#0"; "Vsocx32.ocx"
Begin VB.Form MultipleSetup
  Caption        =   "Multiple Setup"
  ClientHeight   =   8610
  ClientLeft     =   2325
  ClientTop      =   2010
  ClientWidth    =   3255
  Icon           =   "multiplesetup.frx":0000
  LinkTopic      =   "Form1"
  PaletteMode    =   1  'UseZOrder
  ScaleHeight    =   8610
  ScaleWidth     =   3255
  Begin VsOcxLib.VideoSoftElastic VideoSoftElastic1
    Height         =   8610
```

```
Left            =   0
TabIndex        =   0
Top             =   0
Width           =   3255
_Version        =   327680
_ExtentX        =   5741
_ExtentY        =   15187
_StockProps     =   70
ConvInfo        =   1418783674
Align           =   5
BevelOuter      =   0
CaptionPos      =   0
Picture         =   "multiplesetup.frx":030A
MouseIcon       =   "multiplesetup.frx":0326
Begin VB.CommandButton Command5
   Caption         =   "Set Passwords"
   Height          =   375
   Left            =   120
   TabIndex        =   17
   Top             =   240
   Width           =   3015
End
Begin VB.CommandButton Command2
   Caption         =   "Quit"
   Height          =   375
   Left            =   120
   TabIndex        =   16
   Top             =   7920
   Width           =   3015
End
Begin VB.TextBox trafile
   BeginProperty Font
      Name            =   "MS Sans Serif"
      Size            =   9.75
      Charset         =   0
      Weight          =   400
      Underline       =   0   'False
      Italic          =   0   'False
      Strikethrough   =   0   'False
   EndProperty
   Height          =   375
   Left            =   120
   TabIndex        =   14
   Top             =   5640
   Width           =   3015
End
```

```
Begin VB.CommandButton Command4
   Caption        =   "Send Small Training Files"
   Height         =   375
   Left           =   120
   TabIndex       =   9
   Top            =   6840
   Width          =   3015
End
Begin VB.CommandButton Command3
   Caption        =   "Train on Multiple Machines"
   Height         =   375
   Left           =   120
   TabIndex       =   8
   Top            =   7440
   Width          =   3015
End
Begin VB.CommandButton Command1
   Caption        =   "Retrieve Above Training File"
   Height         =   375
   Left           =   120
   TabIndex       =   7
   Top            =   6240
   Width          =   3015
End
Begin VB.Frame Frame1
   Caption        =   " Training Machines "
   Height         =   4215
   Left           =   120
   TabIndex       =   1
   Top            =   720
   Width          =   3015
   Begin VB.ComboBox Combo1
      Height         =   315
      Index          =   7
      Left           =   360
      Style          =   2  'Dropdown List
      TabIndex       =   13
      Top            =   3720
      Width          =   2295
   End
   Begin VB.ComboBox Combo1
      Height         =   315
      Index          =   6
      Left           =   360
      Style          =   2  'Dropdown List
      TabIndex       =   12
```

```
   Top             =   3240
   Width           =   2295
End
Begin VB.ComboBox Combo1
   Height          =   315
   Index           =   5
   Left            =   360
   Style           =   2  'Dropdown List
   TabIndex        =   11
   Top             =   2760
   Width           =   2295
End
Begin VB.ComboBox Combo1
   Height          =   315
   Index           =   4
   Left            =   360
   Style           =   2  'Dropdown List
   TabIndex        =   6
   Top             =   2280
   Width           =   2295
End
Begin VB.ComboBox Combo1
   Height          =   315
   Index           =   3
   Left            =   360
   Style           =   2  'Dropdown List
   TabIndex        =   5
   Top             =   1800
   Width           =   2295
End
Begin VB.ComboBox Combo1
   Height          =   315
   Index           =   2
   Left            =   360
   Style           =   2  'Dropdown List
   TabIndex        =   4
   Top             =   1320
   Width           =   2295
End
Begin VB.ComboBox Combo1
   Height          =   315
   Index           =   1
   Left            =   360
   Style           =   2  'Dropdown List
   TabIndex        =   3
   Top             =   840
```

```
      Width          =   2295
    End
    Begin VB.ComboBox Combo1
      Height         =   315
      Index          =   0
      ItemData       =   "multiplesetup.frx":0342
      Left           =   360
      List           =   "multiplesetup.frx":0344
      Style          =   2  'Dropdown List
      TabIndex       =   2
      Top            =   360
      Width          =   2295
    End
  End
  Begin VB.Label Label2
    AutoSize       =   -1  'True
    Caption        =   "MSG"
    Height         =   195
    Left           =   120
    TabIndex       =   15
    Top            =   5160
    Width          =   360
  End
  Begin VB.Label Label1
    Alignment      =   2  'Center
    BackColor      =   &H00FFFFFF&
    BorderStyle    =   1  'Fixed Single
    BeginProperty Font
      Name           =   "MS Sans Serif"
      Size           =   12
      Charset        =   0
      Weight         =   400
      Underline      =   0   'False
      Italic         =   0   'False
      Strikethrough  =   0   'False
    EndProperty
    Height         =   375
    Left           =   600
    TabIndex       =   10
    Top            =   5040
    Width          =   2535
  End
 End
End
Attribute VB_Name = "MultipleSetup"
Attribute VB_GlobalNameSpace = False
```

```
Attribute VB_Creatable = False
Attribute VB_PredeclaredId = True
Attribute VB_Exposed = False
'Breaks train files to several machines using ssh port forwarding
Dim transfer_done As Integer
Dim temp$
Dim lastone As Integer
Dim combo_loaded As Boolean



Sub get_coded_name(mach$, a$, code$, usr$, passwd$, Port$)
'decodes coded name to retrieve parts
For i% = 1 To num_machines
   'search for this machines name (e.g. wolf1(2)) and bring back info
   If mach$ = dataRA(i%, 0) Then
      'OK found the entry now dehex and unencrypt the password
      passwd$ = Encrypt(dehex(dataRA(i, 1)), salt)
      'get the localhost port to connect to for port forwarding of ftp
      Port$ = dataRA(i, 3)
      Exit For
   End If
Next
i% = InStr(Trim$(mach$), " ")
a$ = Mid$(mach$, 1, i% - 1)
b$ = Mid$(mach$, i% + 2, 1)
'this is the coded name for file storage prefix e.g. wolf12, b$ will be 1 or 2
code$ = a$ & b$
'now get the username on the remote machine
If b$ = "1" Then
   usr$ = userRA(1)
Else
   usr$ = userRA(2)
End If

End Sub

'given the combo control it gets the trainin1.sam template as master_file.txt
Private Function getfiles(cmb As Control) As Boolean
Dim was_error As Boolean, ErrMsg As String

'get the machine name
mach$ = cmb.Text

'leave if none
If mach$ = "None" Then
```

```
   Exit Function
End If


Localfile$ = App.Path & "\master_file.txt"

'get all the info for this machine
Call get_coded_name(mach$, Mac$, code$, usr$, passwd$, Port$)

'execute the ftp method
Call ftpfrm.ftpFile(False, Localfile$, _
            trafile.Text, Port$, usr$, _
            passwd$, "localhost", _
            was_error, ErrMsg)

'check for errors
If was_error Then
   MsgBox ErrMsg
End If

getfiles = Not was_error


End Function


'this makes sure the user does not set duplicate calculator machines
Private Sub Combo1_Click(index As Integer)
If combo_loaded Then
   Call check_duplicates(index)
End If

End Sub

Private Sub Command1_Click()
Dim Denom

'this retreives the training file template and breaks it up (locally)
'into machine speed weighted pieces to distribute the computations
'across several machines

If Combo1(0).Text = "None" Then
  MsgBox "You must choose at least one machine", 48, "No Machine Chosen"
  Exit Sub
End If
```

```
Screen.MousePointer = 11
Label1.Caption = "Retrieving file"

'getfiles just gets the template file from the top box
If Not getfiles(Combo1(0)) Then
   Label1 = "Retrieve Failed"
   Screen.MousePointer = 0
   Exit Sub
End If

Screen.MousePointer = 0
Label1 = "File Retrieved"

Denom = 0#
'clean out comments etc...
clean_master
'save the bottom piece
pre_process
'get the relative weights (sum(1/wt))
For i = 0 To 7
   If Combo1(i).ListIndex = 0 Then
   Else
      Denom = Denom + 1 / Val(dataRA(Combo1(i).ListIndex, 2))
   End If
Next
'Construct the trainin1.sam files for each machine as tra0,tra1 etc.
k = 0
starter = 1
numdone = 0

'find lastone (the number of machines for this calc)
lastone = 0
For i = 0 To 7
   If Combo1(i).ListIndex <> 0 Then
      lastone = lastone + 1
   End If
Next


For i = 0 To 7
   If Combo1(i).ListIndex = 0 Then
   Else
      'construct a training file locally as tr0.txt,tr1.txt.....
      a$ = App.Path & "\tra" & Chr$(48 + k) & ".txt"
      k = k + 1
```

```
      Open a$ For Output As #1
      'num molecules
      a$ = Trim$(str$(numpts)) & "."
      Print #1, a$
      Print #1, " "
      'begin the parameter list
      Print #1, "["
      'get the weighted number of params for this machine
      thisnum = Int(((numparam / Val(dataRA(Combo1(i).ListIndex, 2))) / Denom) + 0.5)
      'find as big as it can get
      maxpos = numparam - numdone
      'set it back if it is too big
      If thisnum > maxpos Then
         thisnum = maxpos
      End If
      'if we are on the last machine then it must absorb any mismatch
      If lastone = k Then
         thisnum = maxpos
      End If
      'set how many we have done
      numdone = numdone + thisnum
      'spit out the parameters for this training set
      For ii = starter To starter + thisnum - 1
         If ii = starter + thisnum - 1 Then
            'last one gets no comma
            a$ = RTrim$(paramRA(ii))
            Print #1, a$
         Else
            a$ = RTrim$(paramRA(ii)) & ","
            Print #1, a$
         End If
      Next
      'where to start next training set
      starter = starter + thisnum
      'close the list
      Print #1, "]."
      'all the bottoms (the observed data) are the same for each training set
      a$ = App.Path & "\bottom_file.txt"
      Open a$ For Input As #2
      Do While Not EOF(2)
         Line Input #2, a$
         Print #1, a$
      Loop
      Close
   End If
Next
```

```
Label1.Caption = "File retrieved"
Screen.MousePointer = 0

Beep
End Sub

Private Sub Command2_Click()
   'quit
   On Error Resume Next
   End
End Sub

Private Sub Command3_Click()
   'show the training window multipletrain.frm
   If Combo1(0).Text = "None" Then
      MsgBox "You must choose at least one machine", 48, "No Machine Chosen"
      Exit Sub
   End If
   MultipleTrain.Show
   Me.Hide

End Sub

Private Sub Command4_Click()

'send all the broken up training files (tra0, tra1 etc) back to the training machines
Dim was_error As Boolean, ErrMsg As String

If Combo1(0).Text = "None" Then
   MsgBox "You must choose at least one machine", 48, "No Machine Chosen"
   Exit Sub
End If

Label1.Caption = "Sending files"
DoEvents
Screen.MousePointer = 11


For i = 0 To 7
   If Combo1(i).ListIndex = 0 Then
   Else
      'generate local file name
      localname$ = App.Path & "\tra" & Trim$(Chr$(48 + i)) & ".txt"
      mach$ = Combo1(i).Text
```

```
        'get all the particulars for the machine to send to
        Call get_coded_name(mach$, Mac$, code$, usr$, passwd$, Port$)
        'remote name always trainin1.sam
        remotename$ = "trainin1.sam"
        L$ = "Sending " & Mac$ & " " & localname$
        Label1.Caption = L$

        'send the file
        Call ftpfrm.ftpFile(True, localname$, _
              remotename$, Port$, usr$, _
              passwd$, "localhost", _
              was_error, ErrMsg)
        If was_error Then
           MsgBox ErrMsg
           Label1 = "Transfer Error"
           Exit Sub
        End If
        Beep
        DoEvents
        DoEvents
        DoEvents
        DoEvents
        DoEvents
        DoEvents
        On Error Resume Next
        FTP1.Action = 3 ' logoff
    End If
Next

Label1.Caption = "Files sent"
Screen.MousePointer = 0

'save the machine state for the next time we come back in
a$ = App.Path & "\last_machine_state.txt"
Open a$ For Output As #10
For i = 0 To 7
    Print #10, Combo1(i).ListIndex
Next
Print #10, Trim(trafile.Text)
Close #10

Exit Sub

E1:
FTP1.Action = 3
MsgBox "Unable to find the remote host", 16, "Remote Host Error"
```

```
End
E2:
FTP1.Action = 3
MsgBox "The file could not be transferred" & Chr$(13) & Chr$(10) & "", 16, "File Transfer Error"
End

End Sub


Private Sub Command5_Click()
   'show password form to enable user to change the encrypted passwords
   PasswdForm.Show
End Sub

Private Sub Form_Load()

   'to keep the combo change event quiet during loading
   combo_loaded = False

   center Me
   Me.Show

   'this retrieves all the allowed machine info
   get_machines

   'curently allows 8 training machines
   'fill the drop down box
   For i% = 0 To 7
      With Combo1(i%)
         .AddItem "None"
         For j% = 1 To num_machines
            .AddItem dataRA(j%, 0)
         Next
         .ListIndex = 0
       End With
   Next

   'see if old machine state exists and prefill
   a$ = App.Path & "\last_machine_state.txt"
   If FileExists(a$) Then
      Open a$ For Input As #1
      For i% = 0 To 7
         Line Input #1, a$
         Combo1(i%).ListIndex = Val(a$)
      Next
```

```
        On Error Resume Next
        Line Input #1, a$
        trafile.Text = Trim(a$)
        Close #1
    End If

    'now turn on the checking for duplicates
    combo_loaded = True

End Sub

Private Sub Form_Unload(Cancel As Integer)
    On Error Resume Next
    Unload ftpfrm
    Unload MultipleTrain
    Unload newparam
    Unload PasswdForm
    Unload Me
End Sub

Public Sub check_duplicates(index As Integer)
    'called from the combo change event

    'if the box was changed to none then has to be OK so just leave
    If Combo1(index).ListIndex = 0 Then
        Exit Sub
    End If

    'loop through all the machines that are not this machine
    For i% = 0 To 7
        'only go in if it is another machine
        If i% <> index Then
            'is it the same as another machine?
            If Combo1(i%).ListIndex = Combo1(index).ListIndex Then
                'it was so display error
                If combo_loaded Then
                    MsgBox "This is a duplicate"
                End If
                'now set it back to none
                combo_loaded = False
                Combo1(index).ListIndex = 0
                combo_loaded = True
                Exit Sub
            End If
        End If
    Next
```

```
End Sub

" File multipletrain.frm
VERSION 5.00
Object = "{2037E3AD-18D6-101C-8158-221E4B551F8E}#5.0#0"; "vsocx32.ocx"
Object = "{33335113-F789-11CE-86F8-0020AFD8C6DB}#2.0#0"; "IPPORT35.Ocx"
Begin VB.Form MultipleTrain
  Caption        =   "Multiple Train"
  ClientHeight   =   7470
  ClientLeft     =   2325
  ClientTop      =   2010
  ClientWidth    =   6675
  ControlBox     =   0   'False
  Icon           =   "multipletrain.frx":0000
  LinkTopic      =   "Form1"
  PaletteMode    =   1  'UseZOrder
  ScaleHeight    =   7470
  ScaleWidth     =   6675
  Begin VsOcxLib.VideoSoftElastic VideoSoftElastic1
    Height         =   7470
    Left           =   0
    TabIndex       =   0
    Top            =   0
    Width          =   6675
    _Version       =   327680
    _ExtentX       =   11774
    _ExtentY       =   13176
    _StockProps    =   70
    ConvInfo       =   1418783674
    Align          =   5
    BevelOuter     =   0
    CaptionPos     =   0
    Picture        =   "multipletrain.frx":030A
    MouseIcon      =   "multipletrain.frx":0326
    Begin VB.Frame Frame3
      Height         =   975
      Left           =   240
      TabIndex       =   48
      Top            =   6360
      Width          =   2775
      Begin VB.TextBox Damp
        Alignment      =   2  'Center
        Height         =   285
        Left           =   1320
        TabIndex       =   50
        Text           =   "1"
```

```
    Top             =   240
    Width           =   495
  End
  Begin VB.OptionButton DoBounding
    Caption         =   " Do Bounding"
    Height          =   255
    Left            =   240
    TabIndex        =   49
    Top             =   600
    Width           =   1575
  End
  Begin VB.Label Label8
    AutoSize        =   -1  'True
    Caption         =   "Damping"
    BeginProperty Font
      Name            =   "MS Sans Serif"
      Size            =   9.75
      Charset         =   0
      Weight          =   400
      Underline       =   0   'False
      Italic          =   0   'False
      Strikethrough   =   0   'False
    EndProperty
    Height          =   360
    Left            =   240
    TabIndex        =   51
    Top             =   240
    Width           =   825
  End
End
Begin VB.CommandButton Command9
  Caption         =   "Send RPFTD"
  Height          =   375
  Left            =   5040
  TabIndex        =   47
  Top             =   6720
  Width           =   1335
End
Begin VB.TextBox NumPass
  Alignment       =   2  'Center
  BeginProperty Font
    Name            =   "MS Sans Serif"
    Size            =   9.75
    Charset         =   0
    Weight          =   400
    Underline       =   0   'False
```

```
    Italic          =   0   'False
    Strikethrough   =   0   'False
  EndProperty
  Height          =   375
  Left            =   5520
  TabIndex        =   46
  Text            =   "1"
  Top             =   120
  Width           =   615
End
Begin VB.TextBox tlevel
  Alignment       =   2  'Center
  BeginProperty Font
    Name            =   "MS Sans Serif"
    Size            =   9.75
    Charset         =   0
    Weight          =   400
    Underline       =   0   'False
    Italic          =   0   'False
    Strikethrough   =   0   'False
  EndProperty
  Height          =   375
  Left            =   5520
  TabIndex        =   45
  Text            =   "0"
  Top             =   600
  Width           =   615
End
Begin VB.CommandButton Command8
  Caption         =   "Local Newparam"
  Height          =   375
  Left            =   4920
  TabIndex        =   43
  Top             =   2760
  Width           =   1455
End
Begin VB.CommandButton Command3
  Caption         =   "Retrieve Results"
  Height          =   375
  Left            =   3360
  TabIndex        =   42
  Top             =   2760
  Width           =   1455
End
Begin VB.CommandButton Command7
  Caption         =   "Set Multipliers"
```

```
  Height        =   375
  Left          =   3360
  TabIndex      =   41
  Top           =   6720
  Width         =   1455
End
Begin VB.OptionButton TrainType
  Caption       =   "Hydro"
  Height        =   375
  Index         =   2
  Left          =   4920
  TabIndex      =   40
  Top           =   6000
  Width         =   1095
End
Begin VB.Frame Frame2
  Caption       =   "Batch_init.pro Settings"
  Height        =   1335
  Left          =   240
  TabIndex      =   33
  Top           =   5040
  Width         =   2775
  Begin VB.CheckBox check1
    Caption       =   "methyl_on"
    Height        =   255
    Index         =   5
    Left          =   1320
    TabIndex      =   39
    Top           =   960
    Width         =   1095
  End
  Begin VB.CheckBox check1
    Caption       =   "calc_g"
    Height        =   255
    Index         =   4
    Left          =   240
    TabIndex      =   38
    Top           =   960
    Width         =   1095
  End
  Begin VB.CheckBox check1
    Caption       =   "calc_vol"
    Height        =   255
    Index         =   3
    Left          =   1320
    TabIndex      =   37
```

```
    Top             =   600
    Width           =   1095
  End
  Begin VB.CheckBox check1
    Caption         =   "calc_pole"
    Height          =   255
    Index           =   2
    Left            =   240
    TabIndex        =   36
    Top             =   600
    Width           =   1095
  End
  Begin VB.CheckBox check1
    Caption         =   "calc_dipole"
    Height          =   255
    Index           =   1
    Left            =   1320
    TabIndex        =   35
    Top             =   240
    Width           =   1335
  End
  Begin VB.CheckBox check1
    Caption         =   "calc_ab"
    Height          =   255
    Index           =   0
    Left            =   240
    TabIndex        =   34
    Top             =   240
    Width           =   1095
  End
End
Begin VB.Timer Timer2
  Enabled         =   0   'False
  Left            =   4920
  Top             =   7320
End
Begin VB.OptionButton TrainType
  Caption         =   "pKa"
  Height          =   375
  Index           =   1
  Left            =   4920
  TabIndex        =   32
  Top             =   5640
  Width           =   735
End
Begin VB.OptionButton TrainType
```

```
  Caption        =   "Properties"
  Height         =   375
  Index          =   0
  Left           =   4920
  TabIndex       =   31
  Top            =   5280
  Value          =   -1  'True
  Width          =   1095
End
Begin VB.Timer Timer1
  Enabled        =   0   'False
  Left           =   5400
  Top            =   7320
End
Begin VB.CommandButton Command6
  Caption        =   "Resume Status Query"
  Height         =   375
  Left           =   3360
  TabIndex       =   29
  Top            =   4200
  Width          =   3015
End
Begin VB.CommandButton Command5
  Caption        =   "Previous Page"
  Height         =   375
  Left           =   3360
  TabIndex       =   26
  Top            =   4680
  Width          =   3015
End
Begin VB.CommandButton Command2
  Caption        =   "Store Results at Level 0"
  Height         =   375
  Left           =   3360
  TabIndex       =   25
  Top            =   3240
  Width          =   3015
End
Begin VB.TextBox Text9
  Alignment      =   2  'Center
  BeginProperty Font
    Name           =   "MS Sans Serif"
    Size           =   9.75
    Charset        =   0
    Weight         =   400
    Underline      =   0   'False
```

```
      Italic          =   0   'False
      Strikethrough   =   0   'False
    EndProperty
    Height          =   405
    Left            =   5520
    TabIndex        =   24
    Text            =   "5"
    Top             =   1080
    Width           =   615
  End
  Begin VB.CommandButton Command4
    Caption         =   "Abort Training"
    Height          =   375
    Left            =   3360
    TabIndex        =   11
    Top             =   3720
    Width           =   3015
  End
  Begin VB.CommandButton Command1
    Caption         =   "Start Training at Level Shown Above"
    Height          =   375
    Left            =   3360
    TabIndex        =   10
    Top             =   2280
    Width           =   3015
  End
  Begin VB.Frame Frame1
    Caption         =   " Machine/Molecule Number "
    Height          =   4695
    Left            =   120
    TabIndex        =   1
    Top             =   240
    Width           =   3015
    Begin VB.TextBox Text10
      Alignment       =   2  'Center
      Height          =   375
      Left            =   1440
      TabIndex        =   27
      Text            =   "?"
      Top             =   4200
      Width           =   735
    End
    Begin VB.TextBox Text1
      Alignment       =   2  'Center
      Height          =   375
      Index           =   3
```

```
  Left          =   1440
  TabIndex      =   9
  Text          =   "0"
  Top           =   1800
  Width         =   735
End
Begin VB.TextBox Text1
  Alignment     =   2  'Center
  Height        =   375
  Index         =   2
  Left          =   1440
  TabIndex      =   8
  Text          =   "0"
  Top           =   1320
  Width         =   735
End
Begin VB.TextBox Text1
  Alignment     =   2  'Center
  Height        =   375
  Index         =   1
  Left          =   1440
  TabIndex      =   7
  Text          =   "0"
  Top           =   840
  Width         =   735
End
Begin VB.TextBox Text1
  Alignment     =   2  'Center
  Height        =   375
  Index         =   7
  Left          =   1440
  TabIndex      =   6
  Text          =   "0"
  Top           =   3720
  Width         =   735
End
Begin VB.TextBox Text1
  Alignment     =   2  'Center
  Height        =   375
  Index         =   6
  Left          =   1440
  TabIndex      =   5
  Text          =   "0"
  Top           =   3240
  Width         =   735
End
```

```
Begin VB.TextBox Text1
   Alignment      =   2  'Center
   Height         =   375
   Index          =   5
   Left           =   1440
   TabIndex       =   4
   Text           =   "0"
   Top            =   2760
   Width          =   735
End
Begin VB.TextBox Text1
   Alignment      =   2  'Center
   Height         =   375
   Index          =   4
   Left           =   1440
   TabIndex       =   3
   Text           =   "0"
   Top            =   2280
   Width          =   735
End
Begin VB.TextBox Text1
   Alignment      =   2  'Center
   Height         =   375
   Index          =   0
   Left           =   1440
   TabIndex       =   2
   Text           =   "0"
   Top            =   360
   Width          =   735
End
Begin VB.Shape Shape1
   FillColor      =   &H000000C0&
   FillStyle      =   0  'Solid
   Height         =   135
   Index          =   7
   Left           =   2400
   Top            =   3840
   Width          =   255
End
Begin VB.Shape Shape1
   FillColor      =   &H000000C0&
   FillStyle      =   0  'Solid
   Height         =   135
   Index          =   6
   Left           =   2400
   Top            =   3360
```

```
   Width          =   255
End
Begin VB.Shape Shape1
   FillColor      =   &H000000C0&
   FillStyle      =   0  'Solid
   Height         =   135
   Index          =   5
   Left           =   2400
   Top            =   2880
   Width          =   255
End
Begin VB.Shape Shape1
   FillColor      =   &H000000C0&
   FillStyle      =   0  'Solid
   Height         =   135
   Index          =   4
   Left           =   2400
   Top            =   2400
   Width          =   255
End
Begin VB.Shape Shape1
   FillColor      =   &H000000C0&
   FillStyle      =   0  'Solid
   Height         =   135
   Index          =   3
   Left           =   2400
   Top            =   1920
   Width          =   255
End
Begin VB.Shape Shape1
   FillColor      =   &H000000C0&
   FillStyle      =   0  'Solid
   Height         =   135
   Index          =   2
   Left           =   2400
   Top            =   1440
   Width          =   255
End
Begin VB.Shape Shape1
   FillColor      =   &H000000C0&
   FillStyle      =   0  'Solid
   Height         =   135
   Index          =   1
   Left           =   2400
   Top            =   960
   Width          =   255
```

```
End
Begin VB.Shape Shape1
  FillColor      =   &H000000C0&
  FillStyle      =   0  'Solid
  Height         =   135
  Index          =   0
  Left           =   2400
  Top            =   480
  Width          =   255
End
Begin VB.Label Label6
  Alignment      =   1  'Right Justify
  AutoSize       =   -1  'True
  Caption        =   "Total Molecules"
  Height         =   195
  Left           =   210
  TabIndex       =   28
  Top            =   4320
  Width          =   1125
End
Begin VB.Label Label3
  Alignment      =   1  'Right Justify
  AutoSize       =   -1  'True
  Caption        =   "Label4"
  Height         =   195
  Index          =   7
  Left           =   840
  TabIndex       =   21
  Top            =   3840
  Width          =   480
End
Begin VB.Label Label3
  Alignment      =   1  'Right Justify
  AutoSize       =   -1  'True
  Caption        =   "Label5"
  Height         =   195
  Index          =   6
  Left           =   840
  TabIndex       =   20
  Top            =   3360
  Width          =   480
End
Begin VB.Label Label3
  Alignment      =   1  'Right Justify
  AutoSize       =   -1  'True
  Caption        =   "Label4"
```

```
   Height        =   195
   Index         =   5
   Left          =   840
   TabIndex      =   19
   Top           =   2880
   Width         =   480
End
Begin VB.Label Label3
   Alignment     =   1  'Right Justify
   AutoSize      =   -1  'True
   Caption       =   "Label4"
   Height        =   195
   Index         =   4
   Left          =   840
   TabIndex      =   18
   Top           =   2400
   Width         =   480
End
Begin VB.Label Label3
   Alignment     =   1  'Right Justify
   AutoSize      =   -1  'True
   Caption       =   "Label3"
   Height        =   195
   Index         =   3
   Left          =   840
   TabIndex      =   17
   Top           =   1920
   Width         =   480
End
Begin VB.Label Label3
   Alignment     =   1  'Right Justify
   AutoSize      =   -1  'True
   Caption       =   "Label4"
   Height        =   195
   Index         =   2
   Left          =   840
   TabIndex      =   16
   Top           =   1440
   Width         =   480
End
Begin VB.Label Label3
   Alignment     =   1  'Right Justify
   AutoSize      =   -1  'True
   Caption       =   "Label4"
   Height        =   195
   Index         =   1
```

```
    Left           =   840
    TabIndex       =   15
    Top            =   960
    Width          =   480
  End
  Begin VB.Label Label3
    Alignment      =   1  'Right Justify
    AutoSize       =   -1  'True
    Caption        =   "Label3"
    Height         =   195
    Index          =   0
    Left           =   840
    TabIndex       =   14
    Top            =   480
    Width          =   480
  End
End
Begin VB.Label Label9
  Alignment      =   1  'Right Justify
  AutoSize       =   -1  'True
  Caption        =   "Number of Passes"
  Height         =   195
  Left           =   4110
  TabIndex       =   44
  Top            =   240
  Width          =   1290
End
Begin VB.Label Label7
  AutoSize       =   -1  'True
  Caption        =   "I am training..."
  BeginProperty Font
    Name           =   "MS Sans Serif"
    Size           =   9.75
    Charset        =   0
    Weight         =   400
    Underline      =   0   'False
    Italic         =   0   'False
    Strikethrough  =   0   'False
  EndProperty
  Height         =   240
  Left           =   3480
  TabIndex       =   30
  Top            =   5715
  Width          =   1200
End
Begin IPPortLib.IPPort IPPort1
```

```
  Index          =   3
  Left           =   4320
  Top            =   7320
  _ExtentX       =   741
  _ExtentY       =   741
  RemoteHost     =   ""
  Linger         =   -1  'True
  EOL            =   ""
End
Begin IPPortLib.IPPort IPPort1
  Index          =   2
  Left           =   3840
  Top            =   7320
  _ExtentX       =   741
  _ExtentY       =   741
  RemoteHost     =   ""
  Linger         =   -1  'True
  EOL            =   ""
End
Begin IPPortLib.IPPort IPPort1
  Index          =   1
  Left           =   3360
  Top            =   7320
  _ExtentX       =   741
  _ExtentY       =   741
  RemoteHost     =   ""
  Linger         =   -1  'True
  EOL            =   ""
End
Begin IPPortLib.IPPort IPPort1
  Index          =   0
  Left           =   2880
  Top            =   7320
  _ExtentX       =   741
  _ExtentY       =   741
  RemoteHost     =   ""
  Linger         =   -1  'True
  EOL            =   ""
End
Begin VB.Label Label5
  Alignment      =   1  'Right Justify
  AutoSize       =   -1  'True
  Caption        =   "Query Status Interval (sec)"
  Height         =   195
  Left           =   3540
  TabIndex       =   23
```

```
      Top             =   1200
      Width           =   1875
    End
    Begin VB.Label Label4
      Alignment       =   1  'Right Justify
      AutoSize        =   -1  'True
      Caption         =   "Start at train level"
      Height          =   195
      Left            =   4080
      TabIndex        =   22
      Top             =   720
      Width           =   1350
    End
    Begin VB.Label Label2
      AutoSize        =   -1  'True
      Caption         =   "MSG"
      Height          =   195
      Left            =   3360
      TabIndex        =   13
      Top             =   1800
      Width           =   360
    End
    Begin VB.Label Label1
      Alignment       =   2  'Center
      BackColor       =   &H00FFFFFF&
      BorderStyle     =   1  'Fixed Single
      BeginProperty Font
        Name            =   "MS Sans Serif"
        Size            =   12
        Charset         =   0
        Weight          =   400
        Underline       =   0   'False
        Italic          =   0   'False
        Strikethrough   =   0   'False
      EndProperty
      Height          =   375
      Left            =   3840
      TabIndex        =   12
      Top             =   1680
      Width           =   2535
    End
  End
End
Attribute VB_Name = "MultipleTrain"
Attribute VB_GlobalNameSpace = False
Attribute VB_Creatable = False
```

```
Attribute VB_PredeclaredId = True
Attribute VB_Exposed = False
'This module carries out the remote training by communicating
'through an ssh port forward with a listener on port 3000 on the
'training machines
Dim transfer_done As Integer
Dim temp$
Dim host_connected(4) As Boolean
Dim machine_list(8) As String, user_list(8) As String
Dim pass_list(8) As String
Dim connecting(4) As Boolean
Dim in_sock_read(4) As Integer, read_level(4) As Integer
Dim waiting_to_finish As Boolean, color_wait As Boolean
Dim FilesRetrieved As Boolean
Dim inMultiplePass As Boolean
Dim onPass As Integer


'same as in the setup module (see comments there)
Sub get_coded_name(mach$, a$, code$, usr$, passwd$, Port$)
For i% = 1 To num_machines
   If mach$ = dataRA(i%, 0) Then
      passwd$ = Encrypt(dehex(dataRA(i, 1)), salt)
      Port$ = dataRA(i, 3)
      Exit For
   End If
Next
i% = InStr(Trim$(mach$), " ")
a$ = Mid$(mach$, 1, i% - 1)
b$ = Mid$(mach$, i% + 2, 1)
code$ = a$ & b$
If b$ = "1" Then
   usr$ = userRA(1)
Else
   usr$ = userRA(2)
End If
End Sub

'this routine gets train.wk1 and train.dat from the machine identified
'through the combo control and stores it as a coded_name file
Sub getfiles(cmb As Control)

Dim was_error As Boolean, ErrMsg As String

mach$ = cmb.Text
If mach$ = "None" Then
```

```
   Exit Sub
End If

'get all the info for the specified machine
Call get_coded_name(mach$, Mac$, code$, usr$, passwd$, Port$)

'set local file name
On Error Resume Next
Localfile$ = App.Path & "\" & code$ & "train.wk1"

'remove any existing local copy
Kill Localfile$

'get the file
Call ftpfrm.ftpFile(False, Localfile$, _
      "train.wk1", Port$, usr$, _
      passwd$, "localhost", _
      was_error, ErrMsg)
If was_error Then
   MsgBox ErrMsg
   Label1 = "Error train.wk1"
   Exit Sub
End If

Beep
DoEvents
DoEvents
DoEvents

'repeat procedure for the train.dat file

'set local file name
On Error Resume Next
Localfile$ = App.Path & "\" & code$ & "train.dat"
Kill Localfile$

Call ftpfrm.ftpFile(False, Localfile$, _
      "train.dat", Port$, usr$, _
      passwd$, "localhost", _
      was_error, ErrMsg)
If was_error Then
   MsgBox ErrMsg
   Label1 = "Error train.dat"
   Exit Sub
End If
```

```
Beep
DoEvents
DoEvents
DoEvents


End Sub



'this communicates with the remote listener and starts the trainsparc session going
Private Sub Command1_Click()
If Not inMultiplePass Then
   onPass = 0
   If Val(NumPass.Text) > 1 Then
      inMultiplePass = True
   End If
Else
   If onPass >= Val(NumPass.Text) Then
      inMultiplePass = False
      MsgBox "Multiple pass (" & onPass & ") is complete."
      Command8.value = True
      Exit Sub
   End If
End If
   Label1 = "Starting sessions..."

   'check that everybody is still connected
   For i% = 0 To 7
      If MultipleSetup.Combo1(i%).ListIndex <> 0 Then
         j% = (MultipleSetup.Combo1(i%).ListIndex - 1) \ 2
         If Not host_connected(j%) Then
            MsgBox MultipleSetup.Combo1(i%).Text & "not connected. Aborting!"
            Exit Sub
         End If
      End If
   Next

   'create and send batch_init.pro
   a$ = App.Path & "\bi.pro"
   Open a$ For Output As #1
   Print #1, " "
   'create the batch init entries
   Print #1, "dont_save_this_level."
   For i% = 0 To 5
      If check1(i%).value = 1 Then
```

```
        Select Case i%
           Case 0
              Print #1, "calc_ab."
           Case 1
              Print #1, "calc_dipole."
           Case 2
              Print #1, "calc_pole."
           Case 3
              Print #1, "calc_vol."
           Case 4
              Print #1, "calc_g."
           Case 5
              Print #1, "methyl_on."
        End Select
     End If
  Next
  Close #1

  'send the batch_init file to each of the training machines
  Call send_bi
  'send the multipliers.dat file to each of the training machines
  Call send_mult

  For i% = 0 To 7
     If MultipleSetup.Combo1(i%).ListIndex <> 0 Then
        j% = (MultipleSetup.Combo1(i%).ListIndex - 1) \ 2
        g! = Val(Damp.Text)
        'fix the damping factor to be prolog readable
        Call fixit(g!, aaa$)
        If DoBounding.value Then
           bbb$ = "1"
        Else
           bbb$ = "0"
        End If
        'set up string to send to remote machine for starting the job
        If TrainType(0) Then
           a$ = "[startprop," & Trim(user_list(i%)) & "," & Trim(str$(Val(tlevel))) & "," &
Trim(str(i%)) & "," & aaa$ & "," & bbb$ & "]."
        ElseIf TrainType(1) Then
           a$ = "[startpka," & Trim(user_list(i%)) & "," & Trim(str$(Val(tlevel))) & "," &
Trim(str(i%)) & "," & aaa$ & "," & bbb$ & "]."
        Else
           a$ = "[starthydro," & Trim(user_list(i%)) & "," & Trim(str$(Val(tlevel))) & "," &
Trim(str(i%)) & "," & aaa$ & "," & bbb$ & "]."
        End If
        waiting_to_finish = True
```

```
            Call send2wsk(j%, a$)
            Do While waiting_to_finish
               DoEvents
            Loop
            'the remote machine has now started the job so start the next one
         End If
      Next

      'start query clock to monitor the job progress on each machine
      Timer1.Enabled = False
      Timer1.Interval = 1000 * Val(Text9.Text)
      Timer1.Enabled = True

      'save state of run so that it can prefill on next entry
      a$ = App.Path & "\last_machine_state2.txt"
      Open a$ For Output As #10
      If TrainType(1).value Then
         Print #10, "pka"
      ElseIf TrainType(0).value Then
         Print #10, "prop"
      Else
         Print #10, "hydro"
      End If
      For i% = 0 To 5
         Print #10, check1(i%).value
      Next
      Print #10, Trim(Damp.Text)
      Close #10

End Sub

Private Sub Command2_Click()

      'store results of last pass by using a remote transfer to /lastX
      Label1 = "Tranfer files..."

      'make sure we are connected everywhere
      For i% = 0 To 7
         If MultipleSetup.Combo1(i%).ListIndex <> 0 Then
            j% = (MultipleSetup.Combo1(i%).ListIndex - 1) \ 2
            If Not host_connected(j%) Then
               MsgBox MultipleSetup.Combo1(i%).Text & "not connected. Aborting!"
               Exit Sub
            End If
         End If
      Next
```

```
   'now give each machine the tranfer command
   For i% = 0 To 7
      If MultipleSetup.Combo1(i%).ListIndex <> 0 Then
         j% = (MultipleSetup.Combo1(i%).ListIndex - 1) \ 2
         a$ = "[transfer," & Trim(user_list(i%)) & "," & Trim(str$(Val(tlevel) + 1)) & "]."
         waiting_to_finish = True
         Call send2wsk(j%, a$)
         Do While waiting_to_finish
            DoEvents
         Loop
      End If
   Next

   Label1 = "Done Transfer"
   Beep
   'change the train level to next higher
   tlevel.Text = str(Val(tlevel.Text) + 1)

End Sub

Private Sub Command3_Click()

Dim was_error As Boolean, ErrMsg As String
If FilesRetrieved Then
   MsgBox "Can't do this again until another pass is made"
   Exit Sub
End If

If MultipleSetup.Combo1(0).Text = "None" Then
   MsgBox "You must choose at least on machine", 48, "No Machine Chosen"
   Exit Sub
End If

Screen.MousePointer = 11
Label1.Caption = "Retrieving files"

'get train.wk1 and train.dat from each machine and save as a coded_name file
For i% = 0 To 7
   Call getfiles(MultipleSetup.Combo1(i%))
Next

Label1.Caption = "Files retrieved"

Dim npar(8) As Integer
```

```
Label1.Caption = "Processing files"
DoEvents

'now generate the combined final file for reparam input
a$ = App.Path & "\train.wk1"
Open a$ For Output As #2
For i% = 0 To 7
   npar(i%) = 0
   mach$ = MultipleSetup.Combo1(i%).Text
   If mach$ <> "None" Then
      'for each machine open the coded_name file
      Call get_coded_name(mach$, Mac$, code$, usr$, passwd$, Port$)
      a$ = App.Path & "\" & code$ & "train.wk1"
      'this has all the param names trained on this particular machine
      Open a$ For Input As #1
      Do While Not EOF(1)
       Line Input #1, a$
       npar(i%) = npar(i%) + 1
       Print #2, a$
      Loop
      Close #1
   End If
Next
Close #2

'now open (and leave open) a file handle for each coded_name train.dat file
ntotal% = 0
For i% = 0 To 7
   ntotal% = ntotal% + npar(i%)
   mach$ = MultipleSetup.Combo1(i%).Text
   If mach$ <> "None" Then
      Call get_coded_name(mach$, Mac$, code$, usr$, passwd$, Port$)
      a$ = App.Path & "\" & code$ & "train.dat"
      j% = i% + 1
      'strip top 2 lines
      Open a$ For Input As #j%
      Line Input #j%, a$
      Line Input #j%, b$
   End If
Next

'now we have all the appropriate files open with the top 2 lines stripped
'we now combine into train.dat
aa$ = App.Path & "\train.dat"
Open aa$ For Output As #10
'from above same for all files so last is OK
```

```
Print #10, a$
'print new total number of params
Print #10, ntotal%

For i% = 0 To 7
   mach$ = MultipleSetup.Combo1(i%).Text
   'see if it is used
   If mach$ <> "None" Then
      'file handle for this machine
      j% = i% + 1
      'read from little file and add to full train.dat
      For k% = 1 To npar(i%)
         Line Input #j%, a$
         Print #10, a$
      Next
      'the next in line is the number of molecules
      Input #j%, nobs%
   End If
Next
'write this out
Print #10, nobs%
only_one% = False

For i% = 0 To 7
   mach$ = MultipleSetup.Combo1(i%).Text
   If mach$ <> "None" Then
      'file handle for the machine
      j% = i% + 1
      'read from small and append to big
      For k% = 1 To 2 * nobs%
         Line Input #j%, a$
         If Not only_one% Then
            Print #10, a$
         End If
      Next
      only_one% = True
   End If
Next

'repeat for next block
For i% = 0 To 7
   mach$ = MultipleSetup.Combo1(i%).Text
   If mach$ <> "None" Then
      'file handle for the machine
      j% = i% + 1
      Do While Not EOF(j%)
```

```
            Line Input #j%, a$
            Print #10, a$
        Loop
        Close #j%
    End If
Next
Close #10

Label1.Caption = "Files Processed"
DoEvents
DoEvents
DoEvents
DoEvents
DoEvents
DoEvents
DoEvents



Label1.Caption = "Sending files"
DoEvents
DoEvents
DoEvents
DoEvents
DoEvents
DoEvents

'at this point the combined train.wk1 and train.dat files are ready

'only the top machine will get them as this is the machine that will
'do the reparam to construct all the files needed for the next pass
mach$ = MultipleSetup.Combo1(0).Text
Call get_coded_name(mach$, Mac$, code$, usr$, passwd$, Port$)

Remotefile$ = "train.wk1"
Localfile$ = App.Path & "\train.wk1"

L$ = "Sending " & Mac$ & " " & Remotefile$
Label1.Caption = L$

'put the file train.wk1 to the top machine
Call ftpfrm.ftpFile(True, Localfile$, _
        Remotefile$, Port$, usr$, _
        passwd$, "localhost", _
        was_error, ErrMsg)
If was_error Then
```

```
   Label1 = "SndErr train.wk1"
   MsgBox ErrMsg
   Exit Sub
End If


Remotefile$ = "train.dat"
Localfile = App.Path & "\train.dat"

L$ = "Sending " & Mac$ & " " & Remotefile$
Label1.Caption = L$

'put the file train.dat to the top machine
Call ftpfrm.ftpFile(True, Localfile$, _
      Remotefile$, Port$, usr$, _
      passwd$, "localhost", _
      was_error, ErrMsg)
If was_error Then
   Label1 = "SndErr train.dat"
   MsgBox ErrMsg
   Exit Sub
End If


DoEvents
DoEvents
DoEvents
DoEvents
DoEvents
DoEvents

On Error Resume Next
ftpfrm.FTP1.Action = 3 ' logoff

Label1.Caption = "Files sent"

DoEvents
DoEvents
DoEvents
DoEvents
DoEvents
DoEvents

'now we ask the remote top machine to perform a reparam
j% = (MultipleSetup.Combo1(0).ListIndex - 1) \ 2
a$ = "[reparam," & Trim(user_list(0)) & "]."
```

```
waiting_to_finish = True
Call send2wsk(j%, a$)
Do While waiting_to_finish
   DoEvents
Loop

If InStr(Label1.Caption, "OK") = 0 Then
   Screen.MousePointer = 0
   MsgBox "Failed to create newparam.sam" & vbCrLf & Label1.Caption & vbCrLf & "Go fix
it"
   Exit Sub
End If

'get the four base files from the top machine needed to do an r_t on the other side
get_base_files

'put these same files into the root area of each training machine
put_base_files

'we are done! Whew!

FilesRetrieved = True

Screen.MousePointer = 0
If Not inMultiplePass Then
   newparam.Show
End If
Exit Sub

E1:
ftpfrm.FTP1.Action = 3
MsgBox "Unable to find the remote host", 16, "Remote Host Error"
End
E2:
ftpfrm.FTP1.Action = 3
MsgBox "The file could not be transferred" & Chr$(13) & Chr$(10) & "", 16, "File Transfer
Error"
End

End Sub

Private Sub Command4_Click()

   'this aborts a remote job
   Timer1.Enabled = False
   Label1 = "Aborting sessions..."
```

```
   'check that we are still connected to all the machines
   For i% = 0 To 7
      If MultipleSetup.Combo1(i%).ListIndex <> 0 Then
         j% = (MultipleSetup.Combo1(i%).ListIndex - 1) \ 2
         If Not host_connected(j%) Then
            MsgBox MultipleSetup.Combo1(i%).Text & "not connected!"
            Exit Sub
         End If
      End If
   Next

   'tell each reote machine to abort its job
   For i% = 0 To 7
      If MultipleSetup.Combo1(i%).ListIndex <> 0 Then
         j% = (MultipleSetup.Combo1(i%).ListIndex - 1) \ 2
         a$ = "[abort," & Trim(user_list(i%)) & "," & Trim(str$(i%)) & "]."
         waiting_to_finish = True
         Call send2wsk(j%, a$)
         Do While waiting_to_finish
            DoEvents
         Loop

      End If
   Next



End Sub

Private Sub Command5_Click()
   'go back to the setup form
   If inMultiplePass Then
      i = MsgBox("You are running a multiple pass. If you exit now you will not be able to finish
all the passes." & vbCrLf & "Do you really want to leave?", vbYesNo, "Do you really want to do
it?")
      If i = vbYes Then
         MultipleSetup.Show
         inMultiplePass = False
         Unload Me
      End If
   Else
      MultipleSetup.Show
      Unload Me
   End If
End Sub
```

```
Private Sub Command6_Click()
   'when coming back in after remotely submitting a job this will restart the
   'timer that looks for the molecules completed on the remote machines

   'start query clock
   Timer1.Enabled = False
   Timer1.Interval = 1000 * Val(Text9.Text)
   Timer1.Enabled = True

End Sub


Private Sub Command7_Click()
   Multipliers.Show
End Sub

Private Sub Command8_Click()
   newparam.Show
End Sub

Private Sub Command9_Click()
get_base_files

put_base_files

Command2.value = True
End Sub

Private Sub Form_Activate()
   'after form load or upon re-entry
   FilesRetrieved = True

   Command2.Caption = "Store Results at Level " & Trim(str(Val(tlevel.Text) + 1))
   For i% = 0 To 7
      If MultipleSetup.Combo1(i%).ListIndex <> 0 Then

         'get the machine number (8 processors but only 4 machines)
         j% = (MultipleSetup.Combo1(i%).ListIndex - 1) \ 2

         'fill the host_connected array
         If IPPort1(j%).Connected Then
            host_connected(j%) = True
         Else
            host_connected(j%) = False
         End If
```

```
        'connect if not currently connected
        If Not host_connected(j%) Then
           IPPort1(j%).WinsockLoaded = True
           IPPort1(j%).EOL = Chr$(10)
           Port$ = str(Val(dataRA(MultipleSetup.Combo1(i%).ListIndex, 3)) + 1000)
           IPPort1(j%).RemotePort = Val(Port$)
           'for port forwarding
           IPPort1(j%).RemoteHost = "127.0.0.1"
           connecting(j%) = True
           IPPort1(j%).Connected = True
           Do While connecting(j%)
              DoEvents
           Loop
        End If

        'show green(connected) red(disconnected) box
        If host_connected(j%) Then
           Shape1(i%).FillColor = &HC000&
        Else
           Shape1(i%).FillColor = &HC0&
        End If

        'retrieve the number of molecules being trained from the top machine
        If i% = 0 And host_connected(j%) Then
           a$ = "[get_num_molecules," & user_list(0) & "]."
           Call send2wsk(j%, a$)
        End If
     End If
  Next

  'see if old machine state exists and prefill
  a$ = App.Path & "\last_machine_state2.txt"
  If FileExists(a$) Then
     Open a$ For Input As #1
     Line Input #1, a$
     If InStr(a$, "pka") > 0 Then
        TrainType(1).value = True
     ElseIf InStr(a$, "prop") > 0 Then
        TrainType(0).value = True
     Else
        TrainType(2).value = True
     End If
     For i% = 0 To 5
        Input #1, jj%
        check1(i%).value = jj%
```

```
        Next
        Input #1, d
        Damp.Text = Trim(str(d))
        Close #1
    End If

End Sub

Private Sub Form_Load()

    center Me
    Me.Show

    'curently allows 4 training machines and 2 processors each
    For i% = 0 To 7
        j% = i% \ 2
        'show the machine name
        Label3(i).Caption = MultipleSetup.Combo1(i).Text

        'fill the machine, password and user lists
        Call fill_lists(i%)

        'first time in they are all disconnected
        host_connected(j%) = False
        IPPort1(j%).Connected = False
        IPPort1(j%).WinsockLoaded = False
    Next
End Sub


Private Sub Form_Unload(Cancel As Integer)
    'show the setup form when this is hidden/unloaded
    MultipleSetup.Show
End Sub




Private Sub IPPort1_Connected(index As Integer, StatusCode As Integer, Description As String)
    If Description <> "OK" Then
        'connection error if here
        a$ = dataRA(2 * index + 1, 0)
        If StatusCode = 10061 Then
         MsgBox Description & CrLf & "Check your port number " & a$
```

```
        Else
         MsgBox Description & CrLf & "Check your port number?" & a$
        End If
        'MsgBox "Connection error: " & Description
        IPPort1(index).Connected = False
        host_connected(index) = False
        connecting(index) = False
        Beep
        Exit Sub
    End If

    'host connected to us
    host_connected(index) = True
    connecting(index) = False
    Beep

End Sub

Private Sub IPPort1_DataIn(index As Integer, Text As String, EOL As Integer)
    Static in_proc$(4), c$(4), flag%(4)

    If EOL = False Then
      c$(index) = c$(index) + Text
      flag%(index) = True
      Exit Sub
    End If

    If EOL And flag%(index) Then
       a$ = c$(index) + Text
       c$(index) = ""
       flag%(index) = False
    Else
       a$ = Text
    End If

    ' for nt
    If a$ <> "" Then
     If Asc(Right$(a$, 1)) = 13 Then
      a$ = Mid$(a$, 1, Len(a$) - 1)
     End If
    End If

    'check this out for traps
    If Left$(a$, 2) = "**" Then
     If in_sock_read(index) Then
      in_sock_read(index) = 0
```

```
      read_level(index) = 0
      MsgBox "ipport error"
     End If
    End If

'    Debug.Print A$
    safe$ = in_proc$(index)

    If in_sock_read(index) Then
       'process the data returned
       Call parse_sock(index, safe$, a$)
    Else
       'just got back echo of procedure so save proc name and get
       'ready for the next set of returned data
       in_proc$(index) = Trim$(a$)
       in_sock_read(index) = True
       read_level(index) = 0
    End If

End Sub

Private Sub IPPort1_Disconnected(index As Integer, StatusCode As Integer, Description As
String)
       'clean up on disconnect
       IPPort1(index).Connected = False
       Shape1(2 * index + 1).FillColor = &HC0&
       host_connected(index) = False

End Sub

Private Sub Timer1_Timer()
    Timer1.Enabled = False

    'each tick of the clock requests each machine to send the number
    'of molecules done so far
    For i% = 0 To 7
       If MultipleSetup.Combo1(i%).ListIndex <> 0 Then
          j% = (MultipleSetup.Combo1(i%).ListIndex - 1) \ 2
          a$ = "[num_done," & Trim(user_list(i%)) & "," & Trim(str$(i%)) & "]."
          waiting_to_finish = True
          Call send2wsk(j%, a$)
          Do While waiting_to_finish
             DoEvents
          Loop
          'the parse routine fills the text box with the number done so far
       End If
```

```
    Next


    total = 0
    total2 = 0
    For i% = 0 To 7
        If MultipleSetup.Combo1(i%).ListIndex <> 0 Then
            total = total + 1
            If Val(Text1(i%).Text) = Val(Text10.Text) Then
                'number done is total # molecules so paint box green
                Shape1(i%).FillColor = &HC000&
                total2 = total2 + 1
            End If
        End If
    Next


    If total = total2 Then
        'all boxes are at final number so the run is done
        Timer1.Enabled = False
        Beep
        FilesRetrieved = False
        If inMultiplePass Then
            onPass = onPass + 1
            Command3.value = True
            Command2.value = True
            Command1.value = True
        Else
            MsgBox "Your run is finished"
        End If
    Else
        'not all done so restart time to keep asking
        Timer1.Enabled = True
    End If
End Sub

Private Sub Timer2_Timer()
    'had to put this in so yellow was on long enough to see
    Timer2.Enabled = False
    color_wait = False
End Sub

Private Sub tlevel_Change()
    'when tlevel changes this changes caption on the button
    Command2.Caption = "Store Results at Level " & Trim(str(Val(tlevel.Text) + 1))
End Sub
```

```
Public Sub fill_lists(num As Integer)
   'fill the machine, password and user lists
   salt = "asaltstringforencryptingpasswords"
   If MultipleSetup.Combo1(num).ListIndex <> 0 Then
      machine_list(num) = MultipleSetup.Combo1(num).ListIndex \ 2
      If (MultipleSetup.Combo1(num).ListIndex Mod 2) = 1 Then
         user_list(num) = userRA(1)
      Else
         user_list(num) = userRA(2)
      End If
      a$ = Encrypt(dehex(dataRA(MultipleSetup.Combo1(num).ListIndex, 1)), salt)
      pass_list(num) = Trim(a$)
   Else
      machine_list(num) = "none"
      user_list(num) = "none"
      pass_list(num) = "none"
   End If

End Sub
Sub send2wsk(index As Integer, a$)
   'send a line to tcp/ip socket
   On Error GoTo snd_error
   a$ = a$ + Chr$(10)
   IPPort1(index).DataToSend = a$
   Exit Sub

snd_error:
      If Err = 25036 Then              'would block
      BytesSent = IPPort1(index).BytesSent
      If BytesSent > 0 Then          'was any sent?
         a$ = Mid$(a$, BytesSent + 1)  'send rest
      End If
      DoEvents       'wait a little while - important!
      Resume         'go back and try again!
  Else               'handle other errors here (this has not been tested)
      MsgBox "An error has occurred in" & Chr$(13) & Chr$(10) & "sending data to the host.",
48, "TCP/IP Send Error"
      IPPort1(index).Connected = False
      On Error GoTo 0
      Exit Sub
  End If
End Sub


Public Sub get_base_files()
```

```
Dim was_error As Boolean, ErrMsg As String

'this gets the 4 base files from the top machine after a reparam

Call get_coded_name(MultipleSetup.Combo1(0).Text, Mac$, code$, usr$, passwd$, Port$)

L$ = "Getting " & Mac$ & " newparam.sam"
Label1.Caption = L$

'set  remotefile$
Remotefile$ = "newparam.sam"
'set local file name
On Error Resume Next
Localfile$ = App.Path & "\" & "newparam.sam"
Kill Localfile$


Call ftpfrm.ftpFile(False, Localfile$, _
            Remotefile$, Port$, usr$, _
            passwd$, "localhost", _
            was_error, ErrMsg)

If was_error Then
   MsgBox "Error getting newparam.sam"
   Exit Sub
End If

On Error Resume Next
ftpfrm.FTP1.Action = 3 ' logoff
DoEvents
DoEvents
DoEvents
DoEvents
DoEvents
DoEvents
DoEvents
DoEvents
DoEvents
DoEvents
DoEvents
DoEvents

L$ = "Getting " & Mac$ & " train.dat"
Label1.Caption = L$
```

```
'set  remotefile$
Remotefile$ = "train.dat"
'set local file name
On Error Resume Next
Localfile$ = App.Path & "\" & "train.dat"
Kill Localfile$


Call ftpfrm.ftpFile(False, Localfile$, _
            Remotefile$, Port$, usr$, _
            passwd$, "localhost", _
            was_error, ErrMsg)

If was_error Then
   MsgBox "Error getting train.dat"
   Exit Sub
End If

DoEvents
DoEvents
On Error Resume Next
ftpfrm.FTP1.Action = 3 ' logoff
DoEvents
DoEvents
DoEvents
DoEvents
DoEvents
DoEvents
DoEvents
DoEvents
DoEvents
DoEvents
DoEvents
DoEvents

L$ = "Getting " & Mac$ & " train.out"
Label1.Caption = L$

'set  remotefile$
Remotefile$ = "train.out"
'set local file name
On Error Resume Next
Localfile$ = App.Path & "\" & "train.out"
Kill Localfile$
```

```
Call ftpfrm.ftpFile(False, Localfile$, _
            Remotefile$, Port$, usr$, _
            passwd$, "localhost", _
            was_error, ErrMsg)

If was_error Then
   MsgBox "Error getting train.out"
   Exit Sub
End If

DoEvents
DoEvents
On Error Resume Next
ftpfrm.FTP1.Action = 3 ' logoff
DoEvents
DoEvents
DoEvents
DoEvents
DoEvents
DoEvents
DoEvents
DoEvents
DoEvents
DoEvents
DoEvents


L$ = "Getting " & Mac$ & " train.wk1"
Label1.Caption = L$

'set  remotefile$
Remotefile$ = "train.wk1"
'set local file name
On Error Resume Next
Localfile$ = App.Path & "\" & "train.wk1"
Kill Localfile$


Call ftpfrm.ftpFile(False, Localfile$, _
            Remotefile$, Port$, usr$, _
            passwd$, "localhost", _
            was_error, ErrMsg)

If was_error Then
   MsgBox "Error getting train.wk1"
```

```
    Exit Sub
End If

DoEvents
DoEvents
On Error Resume Next
ftpfrm.FTP1.Action = 3 ' logoff
DoEvents
DoEvents
DoEvents
DoEvents
DoEvents
DoEvents
DoEvents
DoEvents
DoEvents
DoEvents
DoEvents
DoEvents


End Sub

Public Sub put_base_files()

Dim was_error As Boolean, ErrMsg As String

'this puts the 4 files from a reparam to each training machine

    'check they are all connected
    For i% = 0 To 7
        If MultipleSetup.Combo1(i%).ListIndex <> 0 Then
            j% = (MultipleSetup.Combo1(i%).ListIndex - 1) \ 2
            If Not host_connected(j%) Then
                MsgBox MultipleSetup.Combo1(i%).Text & "not connected. Aborting!"
                Exit Sub
            End If
        End If
    Next

    Label1.Caption = "Sending files"
    DoEvents

    For i% = 0 To 7
        If MultipleSetup.Combo1(i%).ListIndex <> 0 Then
            mach$ = MultipleSetup.Combo1(i%).Text
```

```
Call get_coded_name(mach$, Mac$, code$, usr$, passwd$, Port$)

Remotefile$ = "train.wk1"
Localfile$ = App.Path & "\" & Remotefile$
L$ = "Sending " & Mac$ & " " & Remotefile$
Label1.Caption = L$

Call ftpfrm.ftpFile(True, Localfile$, _
            Remotefile$, Port$, usr$, _
            passwd$, "localhost", _
            was_error, ErrMsg)

If was_error Then
   MsgBox "Error putting " & Remotefile$
   Exit Sub
End If


Remotefile$ = "train.dat"
Localfile$ = App.Path & "\" & Remotefile$
L$ = "Sending " & Mac$ & " " & Remotefile$
Label1.Caption = L$

Call ftpfrm.ftpFile(True, Localfile$, _
            Remotefile$, Port$, usr$, _
            passwd$, "localhost", _
            was_error, ErrMsg)

If was_error Then
   MsgBox "Error putting " & Remotefile$
   Exit Sub
End If


Remotefile$ = "train.out"
Localfile$ = App.Path & "\" & Remotefile$
L$ = "Sending " & Mac$ & " " & Remotefile$
Label1.Caption = L$

Call ftpfrm.ftpFile(True, Localfile$, _
            Remotefile$, Port$, usr$, _
            passwd$, "localhost", _
            was_error, ErrMsg)

If was_error Then
   MsgBox "Error putting " & Remotefile$
```

```
            Exit Sub
        End If


        Remotefile$ = "newparam.sam"
        Localfile$ = App.Path & "\" & Remotefile$
        L$ = "Sending " & Mac$ & " " & Remotefile$
        Label1.Caption = L$

        Call ftpfrm.ftpFile(True, Localfile$, _
                    Remotefile$, Port$, usr$, _
                    passwd$, "localhost", _
                    was_error, ErrMsg)

        If was_error Then
            MsgBox "Error putting " & Remotefile$
            Exit Sub
        End If


        On Error Resume Next
        ftpfrm.FTP1.Action = 3 ' logoff
        DoEvents
        DoEvents
        DoEvents
        DoEvents
        DoEvents
        DoEvents
        DoEvents
      End If
   Next

Label1.Caption = "Files sent"

End Sub

Private Sub fixit(g!, a$)
      'makes sure the number is in a prolog readable format
      a$ = LTrim$(RTrim$(str$(g!)))
      If Left$(a$, 1) = "." Then
       a$ = "0" + a$
      End If
      If Left$(a$, 1) = "-" Then
       If Mid$(a$, 2, 1) = "." Then
        a$ = "-0" + Mid$(a$, 2)
       End If
```

```
    End If
    End Sub

Private Sub parse_sock(index As Integer, safe$, a$)

   'this processes all the return data from the remote machines

   Select Case safe$

      Case "get_num_molecules"
         Label1 = "# Molecules Received"
         Text10.Text = Trim(a$)
         in_sock_read(index) = False
         read_level(index) = 0
      Case "start"
         Label1 = a$
         j = InStr(a$, ",")
         k = Val(Mid$(a$, j + 1))
         in_sock_read(index) = False
         read_level(index) = 0
         Shape1(k).FillColor = &HC00000
         Beep
         waiting_to_finish = False
      Case "abort"
         Label1 = a$
         j = InStr(a$, ",")
         k = Val(Mid$(a$, j + 1))
         in_sock_read(index) = False
         read_level(index) = 0
         Shape1(k).FillColor = &HC000&
         Text1(k).Text = "0"
         Beep
         waiting_to_finish = False
      Case "num_done"
         Label1 = "getting num..."
         j = InStr(a$, ",")
         Click$ = Mid$(a$, 1, j - 1)
         k = Val(Mid$(a$, j + 1))
         Shape1(k).FillColor = &HFFFF&
         Timer2.Interval = 500
         Timer2.Enabled = True
         color_wait = True
         in_sock_read(index) = False
         read_level(index) = 0
         Text1(k).Text = Click$
         Do While color_wait
```

```
            DoEvents
         Loop
         waiting_to_finish = False
         Shape1(k).FillColor = &HF00000
      Case "reparam"
         in_sock_read(index) = False
         read_level(index) = 0
         Label1 = a$
         waiting_to_finish = False
      Case "transfer"
         in_sock_read(index) = False
         read_level(index) = 0
         waiting_to_finish = False
   End Select
End Sub


Public Sub send_bi()

'this just sends the local bi.pro file to each of the training
'machines as batch_init.pro to set calc props

   Dim was_error As Boolean, ErrMsg As String

   Label1.Caption = "Send batch_init"
   DoEvents

   For i% = 0 To 7
      If MultipleSetup.Combo1(i%).ListIndex <> 0 Then
         mach$ = MultipleSetup.Combo1(i%).Text
         Call get_coded_name(mach$, Mac$, code$, usr$, passwd$, Port$)

         Remotefile$ = "batch_init.pro"
         Localfile$ = App.Path & "\bi.pro"

         Call ftpfrm.ftpFile(True, Localfile$, _
                     Remotefile$, Port$, usr$, _
                     passwd$, "localhost", _
                     was_error, ErrMsg)

         If was_error Then
            MsgBox "Error putting " & Remotefile$
            Exit Sub
         End If

         On Error Resume Next
```

```
            ftpfrm.FTP1.Action = 3 ' logoff
            DoEvents
            DoEvents
            DoEvents
            DoEvents
            DoEvents
            DoEvents
            DoEvents
        End If
    Next

Label1.Caption = "Files sent"

End Sub
Public Sub send_mult()

'this just sends the local multipliers.dat file to each of the training
'machines as multipliers.dat to set multipliers

    Dim was_error As Boolean, ErrMsg As String

    Label1.Caption = "Send Multipliers"
    DoEvents

    For i% = 0 To 7
        If MultipleSetup.Combo1(i%).ListIndex <> 0 Then
            mach$ = MultipleSetup.Combo1(i%).Text
            Call get_coded_name(mach$, Mac$, code$, usr$, passwd$, Port$)

            Remotefile$ = "multipliers.dat"
            Localfile$ = App.Path & "\multipliers.dat"

            Call ftpfrm.ftpFile(True, Localfile$, _
                        Remotefile$, Port$, usr$, _
                        passwd$, "localhost", _
                        was_error, ErrMsg)

            If was_error Then
                MsgBox "Error putting " & Remotefile$
                Exit Sub
            End If

            On Error Resume Next
            ftpfrm.FTP1.Action = 3 ' logoff
            DoEvents
            DoEvents
```

```
            DoEvents
            DoEvents
            DoEvents
            DoEvents
            DoEvents
        End If
    Next

Label1.Caption = "Files sent"

End Sub


" File multipliers.frm
VERSION 5.00
Begin VB.Form Multipliers
   Caption        =   "Set Multipliers"
   ClientHeight   =   4605
   ClientLeft     =   60
   ClientTop      =   345
   ClientWidth    =   5895
   LinkTopic      =   "Form1"
   ScaleHeight    =   4605
   ScaleWidth     =   5895
   StartUpPosition =   3  'Windows Default
   Begin VB.CommandButton Command3
      Caption        =   "Cancel"
      Height         =   375
      Left           =   4080
      TabIndex       =   42
      Top            =   4080
      Width          =   1575
   End
   Begin VB.CommandButton Command2
      Caption        =   "Set It"
      Height         =   375
      Left           =   2160
      TabIndex       =   41
      Top            =   4080
      Width          =   1575
   End
   Begin VB.CommandButton Command1
      Caption        =   "Reset to defaults"
      Height         =   375
      Left           =   240
      TabIndex       =   40
```

```
  Top           =   4080
  Width         =   1575
End
Begin VB.TextBox Text1
  Height        =   285
  Index         =   19
  Left          =   4080
  TabIndex      =   39
  Text          =   "Text1"
  Top           =   3480
  Width         =   735
End
Begin VB.TextBox Text1
  Height        =   285
  Index         =   18
  Left          =   4080
  TabIndex      =   38
  Text          =   "Text1"
  Top           =   3120
  Width         =   735
End
Begin VB.TextBox Text1
  Height        =   285
  Index         =   17
  Left          =   4080
  TabIndex      =   37
  Text          =   "Text1"
  Top           =   2760
  Width         =   735
End
Begin VB.TextBox Text1
  Height        =   285
  Index         =   16
  Left          =   4080
  TabIndex      =   36
  Text          =   "Text1"
  Top           =   2400
  Width         =   735
End
Begin VB.TextBox Text1
  Height        =   285
  Index         =   15
  Left          =   4080
  TabIndex      =   35
  Text          =   "Text1"
  Top           =   2040
```

```
   Width          =   735
End
Begin VB.TextBox Text1
   Height         =   285
   Index          =   14
   Left           =   4080
   TabIndex       =   34
   Text           =   "Text1"
   Top            =   1680
   Width          =   735
End
Begin VB.TextBox Text1
   Height         =   285
   Index          =   13
   Left           =   4080
   TabIndex       =   33
   Text           =   "Text1"
   Top            =   1320
   Width          =   735
End
Begin VB.TextBox Text1
   Height         =   285
   Index          =   12
   Left           =   4080
   TabIndex       =   32
   Text           =   "Text1"
   Top            =   960
   Width          =   735
End
Begin VB.TextBox Text1
   Height         =   285
   Index          =   11
   Left           =   4080
   TabIndex       =   31
   Text           =   "Text1"
   Top            =   600
   Width          =   735
End
Begin VB.TextBox Text1
   Height         =   285
   Index          =   10
   Left           =   4080
   TabIndex       =   30
   Text           =   "Text1"
   Top            =   240
   Width          =   735
```

```
End
Begin VB.TextBox Text1
  Height        =   285
  Index         =   9
  Left          =   1320
  TabIndex      =   29
  Text          =   "Text1"
  Top           =   3480
  Width         =   735
End
Begin VB.TextBox Text1
  Height        =   285
  Index         =   8
  Left          =   1320
  TabIndex      =   28
  Text          =   "Text1"
  Top           =   3120
  Width         =   735
End
Begin VB.TextBox Text1
  Height        =   285
  Index         =   7
  Left          =   1320
  TabIndex      =   27
  Text          =   "Text1"
  Top           =   2760
  Width         =   735
End
Begin VB.TextBox Text1
  Height        =   285
  Index         =   6
  Left          =   1320
  TabIndex      =   26
  Text          =   "Text1"
  Top           =   2400
  Width         =   735
End
Begin VB.TextBox Text1
  Height        =   285
  Index         =   5
  Left          =   1320
  TabIndex      =   25
  Text          =   "Text1"
  Top           =   2040
  Width         =   735
End
```

```
Begin VB.TextBox Text1
  Height        =  285
  Index         =  4
  Left          =  1320
  TabIndex      =  24
  Text          =  "Text1"
  Top           =  1680
  Width         =  735
End
Begin VB.TextBox Text1
  Height        =  285
  Index         =  3
  Left          =  1320
  TabIndex      =  23
  Text          =  "Text1"
  Top           =  1320
  Width         =  735
End
Begin VB.TextBox Text1
  Height        =  285
  Index         =  2
  Left          =  1320
  TabIndex      =  22
  Text          =  "Text1"
  Top           =  960
  Width         =  735
End
Begin VB.TextBox Text1
  Height        =  285
  Index         =  1
  Left          =  1320
  TabIndex      =  21
  Text          =  "Text1"
  Top           =  600
  Width         =  735
End
Begin VB.TextBox Text1
  Height        =  285
  Index         =  0
  Left          =  1320
  TabIndex      =  20
  Text          =  "Text1"
  Top           =  240
  Width         =  735
End
Begin VB.Label Label1
```

```
  AutoSize       =   -1  'True
  Caption        =   "otherdrugsol"
  Height         =   195
  Index          =   19
  Left           =   2880
  TabIndex       =   19
  Top            =   3480
  Width          =   870
End
Begin VB.Label Label1
  AutoSize       =   -1  'True
  Caption        =   "otheract"
  Height         =   195
  Index          =   18
  Left           =   2880
  TabIndex       =   18
  Top            =   3120
  Width          =   585
End
Begin VB.Label Label1
  AutoSize       =   -1  'True
  Caption        =   "othersol"
  Height         =   195
  Index          =   17
  Left           =   2880
  TabIndex       =   17
  Top            =   2760
  Width          =   555
End
Begin VB.Label Label1
  AutoSize       =   -1  'True
  Caption        =   "othervp"
  Height         =   195
  Index          =   16
  Left           =   2880
  TabIndex       =   16
  Top            =   2400
  Width          =   540
End
Begin VB.Label Label1
  AutoSize       =   -1  'True
  Caption        =   "othersvp"
  Height         =   195
  Index          =   15
  Left           =   2880
  TabIndex       =   15
```

```
   Top             =   2040
   Width           =   615
End
Begin VB.Label Label1
   AutoSize        =   -1  'True
   Caption         =   "vol"
   Height          =   195
   Index           =   14
   Left            =   2880
   TabIndex        =   14
   Top             =   1680
   Width           =   210
End
Begin VB.Label Label1
   AutoSize        =   -1  'True
   Caption         =   "beta"
   Height          =   195
   Index           =   13
   Left            =   2880
   TabIndex        =   13
   Top             =   1320
   Width           =   315
End
Begin VB.Label Label1
   AutoSize        =   -1  'True
   Caption         =   "alpha"
   Height          =   195
   Index           =   12
   Left            =   2880
   TabIndex        =   12
   Top             =   960
   Width           =   390
End
Begin VB.Label Label1
   AutoSize        =   -1  'True
   Caption         =   "diff"
   Height          =   195
   Index           =   11
   Left            =   2880
   TabIndex        =   11
   Top             =   600
   Width           =   210
End
Begin VB.Label Label1
   AutoSize        =   -1  'True
   Caption         =   "hv"
```

```
   Height        =   195
   Index         =   10
   Left          =   2880
   TabIndex      =   10
   Top           =   240
   Width         =   180
End
Begin VB.Label Label1
   AutoSize      =   -1  'True
   Caption       =   "ret"
   Height        =   195
   Index         =   9
   Left          =   120
   TabIndex      =   9
   Top           =   3480
   Width         =   180
End
Begin VB.Label Label1
   AutoSize      =   -1  'True
   Caption       =   "bp"
   Height        =   195
   Index         =   8
   Left          =   120
   TabIndex      =   8
   Top           =   3120
   Width         =   180
End
Begin VB.Label Label1
   AutoSize      =   -1  'True
   Caption       =   "hen1"
   Height        =   195
   Index         =   7
   Left          =   120
   TabIndex      =   7
   Top           =   2760
   Width         =   360
End
Begin VB.Label Label1
   AutoSize      =   -1  'True
   Caption       =   "hen"
   Height        =   195
   Index         =   6
   Left          =   120
   TabIndex      =   6
   Top           =   2400
   Width         =   270
```

```
End
Begin VB.Label Label1
   AutoSize        =   -1  'True
   Caption         =   "svp"
   Height          =   195
   Index           =   5
   Left            =   120
   TabIndex        =   5
   Top             =   2040
   Width           =   255
End
Begin VB.Label Label1
   AutoSize        =   -1  'True
   Caption         =   "vp"
   Height          =   195
   Index           =   4
   Left            =   120
   TabIndex        =   4
   Top             =   1680
   Width           =   180
End
Begin VB.Label Label1
   AutoSize        =   -1  'True
   Caption         =   "dist"
   Height          =   195
   Index           =   3
   Left            =   120
   TabIndex        =   3
   Top             =   1320
   Width           =   240
End
Begin VB.Label Label1
   AutoSize        =   -1  'True
   Caption         =   "newdrugsol"
   Height          =   195
   Index           =   2
   Left            =   120
   TabIndex        =   2
   Top             =   960
   Width           =   810
End
Begin VB.Label Label1
   AutoSize        =   -1  'True
   Caption         =   "act"
   Height          =   195
   Index           =   1
```

```
    Left           =   120
    TabIndex       =   1
    Top            =   600
    Width          =   225
  End
  Begin VB.Label Label1
    AutoSize       =   -1  'True
    Caption        =   "sol"
    Height         =   195
    Index          =   0
    Left           =   120
    TabIndex       =   0
    Top            =   240
    Width          =   195
  End
End
Attribute VB_Name = "Multipliers"
Attribute VB_GlobalNameSpace = False
Attribute VB_Creatable = False
Attribute VB_PredeclaredId = True
Attribute VB_Exposed = False
Private Sub Command1_Click()
'Default Button
a$ = App.Path & "\default_mult.txt"
Open a$ For Input As #1
For i = 1 To 20
   Line Input #1, a$
   Text1(i - 1) = Trim(a$)
Next
Close #1

End Sub

Private Sub Command2_Click()
a$ = App.Path & "\multipliers.dat"
b$ = "multiply_property("
aa$ = App.Path & "\last_multipliers.txt"
Open a$ For Output As #1
Open aa$ For Output As #2
For i = 1 To 20
   If Trim(Text1(i - 1).Text) = "" Then
      Close
      MsgBox "Multipliers cannot be empty."
      Exit Sub
   End If
   c$ = b$ & Trim(Label1(i - 1).Caption) & "," & Trim(Text1(i - 1).Text) & ")."
```

```
    Print #1, c$
    Print #2, Trim(Text1(i - 1).Text)
Next
Close #1
Close #2
Me.Hide
End Sub

Private Sub Command3_Click()
Me.Hide
End Sub

Private Sub Form_Load()
a$ = App.Path & "\last_multipliers.txt"
Open a$ For Input As #1
For i = 1 To 20
    Line Input #1, a$
    Text1(i - 1) = Trim(a$)
Next
Close #1
End Sub

" File newparam.frm
VERSION 5.00
Object = "{F9043C88-F6F2-101A-A3C9-08002B2F49FB}#1.2#0"; "COMDLG32.OCX"
Begin VB.Form newparam
   Caption         =   "View Newparam"
   ClientHeight    =   7680
   ClientLeft      =   60
   ClientTop       =   405
   ClientWidth     =   10035
   LinkTopic       =   "Form1"
   ScaleHeight     =   7680
   ScaleWidth      =   10035
   StartUpPosition =   3  'Windows Default
   Begin VB.CommandButton Command3
      Caption         =   "View in Notepad"
      Height          =   375
      Left            =   5760
      TabIndex        =   3
      Top             =   7080
      Width           =   1695
   End
   Begin MSComDlg.CommonDialog CommonDialog1
      Left            =   7800
      Top             =   7200
```

```
    _ExtentX       =   847
    _ExtentY       =   847
    _Version       =   393216
  End
  Begin VB.CommandButton Command2
    Caption        =   "Print"
    Height         =   375
    Left           =   3240
    TabIndex       =   2
    Top            =   7080
    Width          =   1935
  End
  Begin VB.CommandButton Command1
    Caption        =   "Close"
    Height         =   375
    Left           =   1080
    TabIndex       =   1
    Top            =   7080
    Width          =   1455
  End
  Begin VB.TextBox Text1
    BeginProperty Font
      Name           =   "MS Sans Serif"
      Size           =   12
      Charset        =   0
      Weight         =   400
      Underline      =   0   'False
      Italic         =   0   'False
      Strikethrough  =   0   'False
    EndProperty
    Height         =   6855
    Left           =   240
    MultiLine      =   -1  'True
    ScrollBars     =   3  'Both
    TabIndex       =   0
    Text           =   "newparam.frx":0000
    Top            =   120
    Width          =   9495
  End
End
Attribute VB_Name = "newparam"
Attribute VB_GlobalNameSpace = False
Attribute VB_Creatable = False
Attribute VB_PredeclaredId = True
Attribute VB_Exposed = False
'This form just displays the newparam.sam file after reparam
```

```
Private Sub Command1_Click()
   Unload Me
End Sub

Private Sub Command2_Click()
  Dim BeginPage, EndPage, NumCopies, Orientation, i
  ' Set Cancel to True.
  CommonDialog1.CancelError = True
  On Error GoTo ErrHandler
  ' Display the Print dialog box.
  CommonDialog1.ShowPrinter
  ' Get user-selected values from the dialog box.
  BeginPage = CommonDialog1.FromPage
  EndPage = CommonDialog1.ToPage
  NumCopies = CommonDialog1.Copies
  Orientation = CommonDialog1.Orientation
  For i = 1 To NumCopies
   Printer.Print " "
   Printer.Print " "
   Printer.FontSize = 14
   Printer.Print "Newparam Output " & "  " & Date & "  " & Time
   Printer.Print " "
   Printer.FontSize = 10
   Printer.Print Text1.Text
  Next
  Printer.EndDoc
  Exit Sub
ErrHandler:
  ' User pressed Cancel button.
  Exit Sub

End Sub

Private Sub Command3_Click()
   a$ = "notepad.exe " & App.Path & "\newparam.sam"
   Shell a$, vbNormalFocus


End Sub

Private Sub Form_Load()
   center Me
   'open the file and read it in one big bite
   a$ = App.Path & "\newparam.sam"
   Open a$ For Input As #11
   Text1.Text = Input(LOF(11), 11)
```

```
    Close (11)


End Sub

" File PasswordForm.frm
VERSION 5.00
Begin VB.Form PasswdForm
   Caption        =   "Set Passwords"
   ClientHeight   =   2985
   ClientLeft     =   60
   ClientTop      =   345
   ClientWidth    =   4110
   LinkTopic      =   "Form1"
   ScaleHeight    =   2985
   ScaleWidth     =   4110
   StartUpPosition =   3  'Windows Default
   Begin VB.CommandButton Command1
      Caption        =   "Set Password"
      Height         =   375
      Left           =   1320
      TabIndex       =   7
      Top            =   2400
      Width          =   2295
   End
   Begin VB.TextBox Text2
      Height         =   285
      IMEMode        =   3  'DISABLE
      Left           =   1320
      PasswordChar   =   "*"
      TabIndex       =   6
      Top            =   1920
      Width          =   2295
   End
   Begin VB.TextBox Text1
      Height         =   285
      IMEMode        =   3  'DISABLE
      Left           =   1320
      PasswordChar   =   "*"
      TabIndex       =   5
      Top            =   1560
      Width          =   2295
   End
   Begin VB.ComboBox Combo1
      Height         =   315
      ItemData       =   "PasswdForm.frx":0000
```

```
  Left          =   1320
  List          =   "PasswdForm.frx":0002
  TabIndex      =   1
  Text          =   "Combo1"
  Top           =   1080
  Width         =   2295
End
Begin VB.Label Label4
  Alignment     =   1  'Right Justify
  Caption       =   "Verify Pass"
  Height        =   255
  Left          =   240
  TabIndex      =   4
  Top           =   1934
  Width         =   855
End
Begin VB.Label Label3
  Alignment     =   1  'Right Justify
  Caption       =   "New Pass"
  Height        =   255
  Left          =   240
  TabIndex      =   3
  Top           =   1574
  Width         =   855
End
Begin VB.Label Label2
  Alignment     =   1  'Right Justify
  Caption       =   "Machine"
  Height        =   255
  Left          =   240
  TabIndex      =   2
  Top           =   1110
  Width         =   855
End
Begin VB.Label Label1
  Alignment     =   2  'Center
  Caption       =   "Change Password"
  BeginProperty Font
    Name          =   "MS Sans Serif"
    Size          =   13.5
    Charset       =   0
    Weight        =   400
    Underline     =   0   'False
    Italic        =   0   'False
    Strikethrough =   0   'False
  EndProperty
```

```
    Height          =   375
    Left            =   488
    TabIndex        =   0
    Top             =   240
    Width           =   3135
  End
End
Attribute VB_Name = "PasswdForm"
Attribute VB_GlobalNameSpace = False
Attribute VB_Creatable = False
Attribute VB_PredeclaredId = True
Attribute VB_Exposed = False

'This form is used to set encrypted passwords for the various machines
Private Sub Command1_Click()
   Dim i, a$
   'ensure the newpasswd and its verify are the same
   If Text1.Text = Text2.Text Then
      a$ = App.Path & "\" & "machines.txt"
      Open a$ For Output As #1
      Print #1, "*** DO NOT EDIT THIS FILE ****"
      Print #1, userRA(1)
      Print #1, userRA(2)
      Print #1, Mid$(domain, 2)
      For i = 1 To num_machines
         If Combo1.Text = dataRA(i, 0) Then
            'first encrypt the password
            a$ = Encrypt(Text1.Text, salt)
            'write the data
            Print #1, dataRA(i, 0)
            'this is the hexified encrypted password
            printhex (a$)
            Print #1, dataRA(i, 2)
            Print #1, dataRA(i, 3)
            a$ = Encrypt(a$, salt) 'for debug purposes
         Else
            Print #1, dataRA(i, 0)
            Print #1, dataRA(i, 1)
            Print #1, dataRA(i, 2)
            Print #1, dataRA(i, 3)
         End If
      Next
      Close
      'now refill everything for immediate use
      get_machines
      Beep
```

```
      Resp% = MsgBox("Password Set", vbExclamation, "Password Set")
   Else
      Beep
      Resp% = MsgBox("Password entry error", vbExclamation, "Password Mismatch")
   End If

End Sub


Private Sub Form_Load()
   Dim i
   center Me
   Me.Show
   'fill all the data in case it is not yet done
   get_machines
   Combo1.Clear
   'fill the drop down box
   For i = 1 To num_machines
      Combo1.AddItem dataRA(i, 0)
   Next
End Sub
```