

IMPROVE
STANDARD OPERATING PROCEDURE

SOP 351 Version 2 (SOP 351-2)
Data Processing and Validation

Crocker Nuclear Laboratory
One Shields Avenue
University of California
Davis, CA 95616-8569


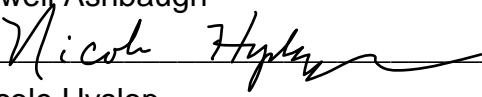
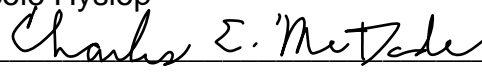
Submitted by
Charles E. McDade

Date of submission
July 16, 2008

APPROVAL SHEET

This Standard Operating Procedures Manual was developed by Crocker Nuclear Laboratory (CNL) to document the activities of the IMPROVE project. The following signatures of key participants indicate agreement with the procedures specified within this SOP manual.

Crocker Nuclear Laboratory

| | |
|---|---------|
|  | 7/16/08 |
| Lowell Ashbaugh | Date |
|  | 7/16/08 |
| Nicole Hyslop | Date |
|  | 7/16/08 |
| Charles McDade | Date |
| | |
| | Date |
| | |
| | Date |

DOCUMENT HISTORY

| Date Modified | Initials | Section/s Modified | Brief Description of Modifications |
|---------------|-----------------|--------------------|---|
| July 2008 | LLA/NPH/ CEM | Entire document | Revised and reorganized SOP. Added appendices with computer code. |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |

SOP 351-2: DATA PROCESSING AND VALIDATION

TABLE OF CONTENTS

| | | |
|-------|--|----|
| 1. | Purpose and Applicability..... | 1 |
| 2. | Responsibilities..... | 1 |
| 2.1 | Project Manager..... | 1 |
| 2.2 | Quality Assurance Officer..... | 1 |
| 2.3 | Data Validation Manager..... | 1 |
| 2.4 | Laboratory and Field Operations Manager..... | 2 |
| 2.5 | Field Operations Technicians..... | 2 |
| 2.6 | Laboratory Technicians..... | 2 |
| 2.7 | Laboratory Students..... | 2 |
| 3. | Required equipment and materials..... | 3 |
| 4. | Data processing and validation overview..... | 5 |
| 5. | Data Processing..... | 7 |
| 5.1 | Artifacts..... | 8 |
| 5.2 | Volume..... | 8 |
| 5.3 | Concentration, Uncertainty, and Minimum Detectable Limit (mdl)..... | 10 |
| 5.3.1 | PM _{2.5} and PM ₁₀ Mass (Module A)..... | 11 |
| 5.3.2 | Ions (Module B)..... | 11 |
| 5.3.3 | Carbon (Module C)..... | 12 |
| 5.3.4 | SO ₂ | 13 |
| 5.3.5 | Elements (Module A)..... | 13 |
| 5.3.6 | Laser Absorption (Module A)..... | 14 |
| 5.4 | Equations of Composite Variables..... | 14 |
| 5.4.1 | Sulfate by XRF (S3) and Ammonium Sulfate (NH ₄ SO ₄)..... | 15 |
| 5.4.2 | Ammonium Nitrate (NH ₄ NO ₃)..... | 15 |
| 5.4.3 | Soil..... | 15 |
| 5.4.4 | Non-soil Potassium (KNON)..... | 16 |
| 5.4.5 | Light-Absorbing Carbon (LAC)..... | 16 |
| 5.4.6 | Organic Mass by Carbon (OMC)..... | 17 |
| 5.4.7 | Organic Mass by Hydrogen (OMH)..... | 17 |
| 5.4.8 | Reconstructed Mass from the Teflon® filter (RCMA)..... | 18 |
| 5.4.9 | Reconstructed Mass using Carbon Measurements (RCMC)..... | 18 |
| 6. | Data validation..... | 19 |
| 6.1 | Definition of status flags..... | 19 |
| 6.2 | Level I Validation Tests..... | 20 |
| 6.2.1 | Validation of laboratory analyses..... | 20 |
| 6.3 | Flow rate validation..... | 20 |
| 6.3.1 | Automated flow rate validation..... | 21 |
| 6.4 | Level II validation procedures..... | 22 |
| 6.4.1 | Module A versus Module B data..... | 22 |
| 6.4.2 | Module A versus Module C data..... | 24 |

| | | |
|-------|---|-----|
| 6.4.3 | Module A versus Module D data | 26 |
| 6.5 | Reconstructed mass versus measured mass (MF)..... | 26 |
| 6.6 | Troubleshooting and final review | 27 |
| 7. | Submission to CIRA | 28 |
| 8. | References..... | 28 |
| | Appendix A. IMPROVE Data Dictionary | A-1 |
| | Appendix B. BuildDBF Code Diagrams | B-1 |
| | Appendix C. BuildDBF Code..... | C-1 |
| | Appendix D. SWAP Code | D-1 |
| | Appendix E. NETWORK_METRIC_CHECKS Code..... | E-1 |
| | Appendix F. Measurement of IMPROVE Flow Rate..... | F-1 |

LIST OF FIGURES

| | |
|---|----|
| Figure 1. Menu screen for SWAP..... | 4 |
| Figure 2. Data processing flow chart..... | 5 |
| Figure 3. Flow rate versus the pressure transducer reading for the IMPROVE-calculated flow rate and the actual flow rate..... | 22 |
| Figure 4. Time series and scatter plot of sulfur (*3) and sulfate for a single site..... | 24 |
| Figure 5. Time series and scatter plot of laser absorption (LRNC) and light absorbing carbon (LAC) for a single site..... | 25 |
| Figure 6. Time series and scatter plot of organic mass by hydrogen (OMH) and organic mass by carbon (OMC) for a single site..... | 25 |
| Figure 7. Time series and scatter plot of PM ₁₀ mass and PM _{2.5} mass for a single site. The difference between the two [coarse mass (CM)] is also shown with the gray line..... | 26 |
| Figure 8. Time series and scatter plot of reconstructed mass (RCMC) and measured fine mass (MF). In this case there is excellent agreement..... | 27 |
| Figure 9. Time series and scatter plot of reconstructed mass (RCMC), reconstructed mass with nitrate (RCMN), and measured fine mass (MF) for a single site. In this case nitrate is a significant contributor to the fine mass..... | 27 |

LIST OF TABLES

| | |
|---|----|
| Table 1. Fractional analytical uncertainty for the ions..... | 12 |
| Table 2. Fractional analytical uncertainty for the carbon species..... | 13 |
| Table 3. Status flags and their definitions..... | 19 |
| Table 4. Flow rate related validation flag definitions and application criteria (John and Reischl, 1980)..... | 21 |

1. PURPOSE AND APPLICABILITY

This standard operating procedure (SOP) document provides an overview of the procedures for processing and validating IMPROVE data. Data processing and data validation are performed in parallel.

2. RESPONSIBILITIES

This section describes the responsibilities of the individuals involved in data processing and validation.

2.1 PROJECT MANAGER

The project manager oversees all aspects of the program.

2.2 QUALITY ASSURANCE OFFICER

The quality assurance officer

- devises techniques that improve the efficiency, traceability, and accuracy of the data management
- characterizes measurement deficiencies
- quantifies measurement uncertainty
- analyzes IMPROVE data to assess the influence of specific components of the measurement process and to assess the quantitative ramifications of procedural changes
- conducts ongoing evaluation of the program's progress in data quality
- develops and applies approaches for assessing data quality
- proposes program improvements
- develops automated data quality checks to increase the objectivity and consistency of the data validation process

2.3 DATA VALIDATION MANAGER

The data validation manager

- receives analyzed data
- reviews ion and carbon data for errors
- reviews and evaluates flow data
- examines flow data for disparities and errors
- builds the concentration database
- examines concentration data site by site for consistency
- notes and resolves inconsistencies in the data
- submits data to project sponsors (CIRA)

2.4 LABORATORY AND FIELD OPERATIONS MANAGER

The laboratory and field operations manager

- oversees all field operations including site installation, maintenance, and routine operations
- coordinates annual routine maintenance
- maintains documentation on the flow rate calibration for each module
- verifies that the flow rates are within acceptable tolerances
- oversees all weighing-room operations from filter acceptance to gravimetric mass data quality checks
- reviews independent audits of the IMPROVE network
- oversees maintenance shop operations

2.5 FIELD OPERATIONS TECHNICIANS

The field operations technicians

- perform annual routine maintenance
- perform annual audits and calibrations of sites
- maintain site documentation, such as pictures, global positioning system coordinates, and site details
- implement any required equipment and/or procedure changes
- train site operators on new equipment and/or procedural changes
- train shop technicians on in-house equipment repairs and field techniques
- install new site locations

2.6 LABORATORY TECHNICIANS

The laboratory technicians

- review log sheets for completeness and check the validity of the samples
- upload flashcard data to the Crocker computer system
- communicate with operators about problems and questions
- maintain the records on site and sampler operations
- maintain timely filter preparation and recovery
- maintain balance calibrations and quality control of measurements
- review final shipments for independent analysis
- perform acceptance tests on new filter lots for use in the network

2.7 LABORATORY STUDENTS

The undergraduate student members of the laboratory staff

- review log sheets for completeness and check the validity of the samples
- perform filter preparation and recovery
- prepare filters for analysis
- prepare filters for archive

3. REQUIRED EQUIPMENT AND MATERIALS

The data processing and validation requires the logs files, weights files, carbon files generated by DRI, the ions files generated by RTI, the PIXEXRF (elements) files, and the laser absorption files. The logs file is used to control the data processing by BuildDBF, SWAP, and other programs. It contains all the information known about the sample related to processing, including the site, date collected, flow rate measurements, elapsed time, validity status, field comments, and other sample specific information. The flow rate calibration files and the flashcard files are required for calculating flow rates, and the balance calibrations are required for calculating filter masses. All these files are stored in subdirectories in the U:\IMPROVE\ directory on the Crocker Nuclear Laboratory server. Appendix A contains the data dictionaries for all these files.

SWAP: This program (written in FoxPro) allows access to the data files for viewing the data, calculating summary statistics, and correcting errors. We began using SWAP in March 2005. Additional capabilities were added to SWAP over the next 14 months. Prior to March 2005, Auxdata was used to perform most of these functions; Auxdata is described in the first version of this SOP. SWAP has two levels of access for different users. Figure 1 shows the main menu screen for SWAP. SWAP provides access (by site and by quarter) to the following information for all users:

- ions, carbon, XRF, laser, and SO₂ data files
- logs files for status codes and flow readings
- problems file
- flow rate calibration files
- weights files, along with calculated filter mass difference
- flashcard data
- ions field blanks, sortable by any ion in ascending and descending order, along with access to median values
- carbon secondary filters, sortable by any carbon fraction in ascending and descending order, along with access to median values
- number of field blanks and their percentage by month and module with comparison to the target value, which is 1-2% of the sample filters
- balance equation data files
- flow rates, including relative standard deviations (see below)

SWAP allows access to several additional subprograms and data files for selected users; all options listed under “Data Processing” are restricted. The following features are used to calculate concentrations and validate data.

- **ArtHist** provides access to the artifact history file so the data validation manager can review the current artifact data in context with prior data.
- **FlowRSD** calculates the relative standard deviation (RSD) of the 15-minute flow and temperature readings for each 24-hour period and presents a table of the results for each module. All RSD values exceeding 3% (2% on Module D) are highlighted. The number of data readings is also highlighted if it does not equal 96. It can be run for all sites to generate a report of only those values exceeding the threshold. This is valuable for data validation before generating concentration files.

- **FlowRate** calculates the average flow rate measurements from the flashcard data and highlights any values outside the “LF” flag limits (discussed in Section 6.3). It also calculates the mean temperature. The background color of the resulting grid cells indicates flow rate RSD ranging from green (low RSD) to yellow (high RSD).

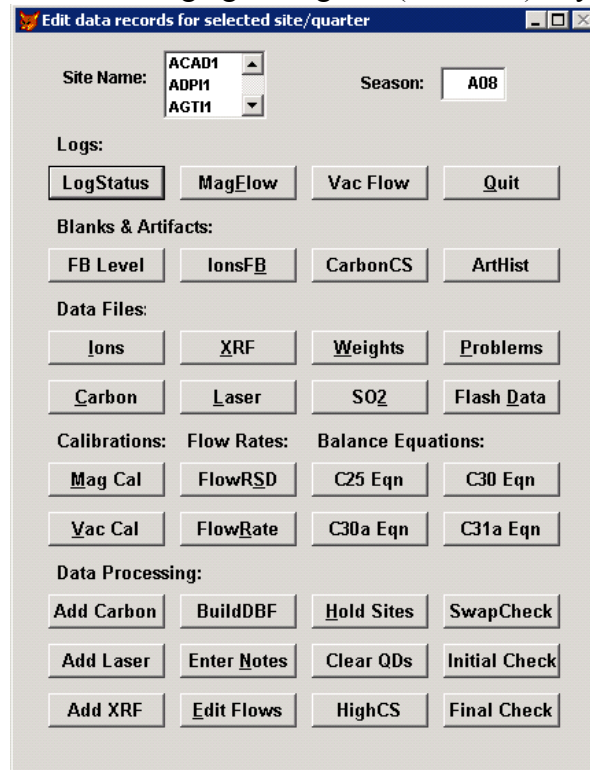


Figure 1. Menu screen for SWAP

- **Add Carbon, Add Laser, and Add XRF** are used to add new data to FoxPro data tables. At this writing the carbon and laser data have been incorporated into SQL tables so these options are not used. Once the XRF data are ingested into a SQL table the Add XRF option will be dropped.
- **BuildDBF** calculates the concentrations for a month at any single site or at all sites. It produces two output files, one for data validation and one for submission to CIRA. They are identical except that the data validation file includes the composite variables (described in Section 5.4). The data validation file is imported into Excel and used to perform the data validation described in Section 6.4. BuildDBF was first used to process June 2002 data but has been modified to deliver data from as far back as January 2000; it was used to redeliver all the data to CIRA from January 2000 through December 2004. Plans are underway to replace it with a SQL Server based data management system.
- **Edit Notes** opens a form to systematically add comments and document changes to the database during validation.
- **Edit Flows** is used to review the flow rate file produced by a SQL query from Flashcard data. This option does not now allow editing.

- **Hold Sites** provides viewing and editing access to the data file that records sites not included in the generated delivery file.
- **Clear QDs** replaces all “QD” flags remaining in the logs file for a given month. It does not clear them if the site is in the HOLDS file.
- **High CS** executes a SQL query that isolates cases where the carbon secondary filter has higher concentrations than the carbon primary filter. This option will find possible swapped primary/secondary filters.
- **SwapCheck** examines the concentration files produced by BuildDBF for potential swaps between the A and B modules, and between the A and C modules.
- **Initial Check** performs some of the checks described below under “Final Check” but does not look for temporary status flags that are intended to be reset during data validation. It’s a useful check on data to start performing data validation.
- **Final Check** opens the output file for a selected month and year. It then runs through a series of checks on the file to be sure it's ready to deliver to CIRA. It shows all records where PM_{2.5} mass is greater than PM₁₀ mass (MF>MT) along with a measure of the significance of the difference, all records where the flow rate has been flagged as “RF”, “CG”, or “CL”, all records with a “QD” flag for any module, all records with an “SA” or “QA” flag, and a list of all sites in the HOLDS file for that month. It also shows all sites that have an unusually large or small number of records for the month.

4. DATA PROCESSING AND VALIDATION OVERVIEW

Data processing for IMPROVE consists of reducing and combining information collected in the field and laboratories to calculate concentrations, uncertainty estimates, and minimum detectable limits (mdl). Several custom software packages are used to perform these manipulations. Figure 2 shows the data involved; the process moves from left to right in the figure, and the concentration data are the final products.

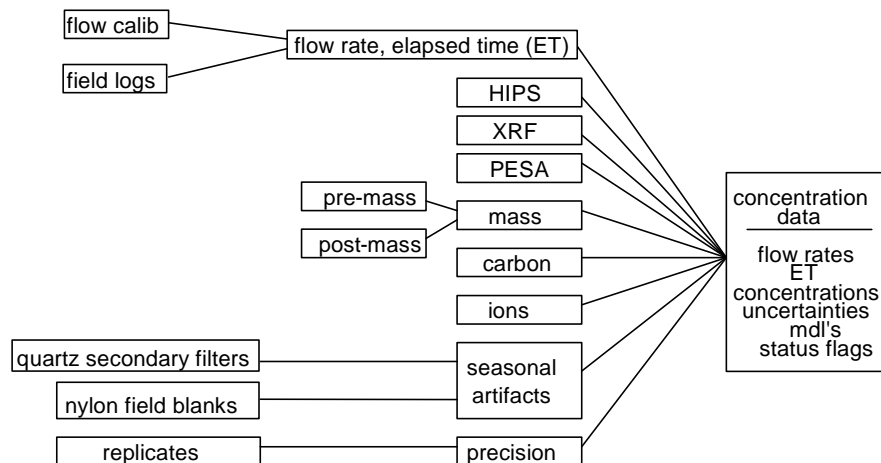


Figure 2. Data processing flow chart

Data validation is not a linear process, and a significant amount of data validation is performed by the analytical laboratories before the data are delivered to the data validation manager. The SOPs for the analytical laboratories describe their data validation procedures in detail.

Watson et al. (1995) define a three-level data validation process for environmental measurement studies. The levels are only intended as general guidelines. The IMPROVE data delivered to CIRA are considered to be a mixture of Level 1B and Level 2 validated data. The levels are applied to IMPROVE as follows:

Level 0: Data at this level are, in essence, raw data, obtained directly from the data-acquiring instruments. These data can be reduced or reformatted, but are unedited and unreviewed, without any adjustments for known biases or problems that might have been identified during preventative maintenance checks or audits. These data may monitor instrument operations on a frequent basis. Averaging times represent the minimum intervals recorded, and these data may need to be aggregated to obtain averages for the sampling periods. Level 0 data have not been edited for instrument downtime, nor have procedural adjustments for baseline shifts, span changes, or known problems been applied. Examples of data at Level 0 validity are

- 15-minute averaged pressure transducer and temperature data downloaded from the sampler flashcards before automated validation tests
- balance measurements before automated validity tests
- XRF raw spectra

Level 1A: Data at this level have passed several qualitative reviews for accuracy and completeness. The focus of Level 1A validation is to obtain as complete a data set as possible. IMPROVE Level 1A data validation consists of

- replacing invalid data values with -999 (the invalid data code) and setting status flags to reflect sampler malfunctions, site or laboratory operator errors, or power outages
- reviewing operator log sheets to verify operation of the sampler
- verifying operator log sheet entries against sampler flashcard data
- setting status flags to use log sheet temperatures if flashcard temperatures are faulty
- replacing flow rate data with data from a backup system (orifice transducer or log sheet readings) in the event of failure of the primary cyclone transducer system
- identifying, investigating, and flagging data that are beyond reasonable bounds or that are unrepresentative of the variable being measured (e.g., flow rate measurements that change significantly over the sampling period)

Level 1B: Data at this level have passed additional automated quantitative and qualitative reviews for accuracy and internal consistency. Objective consistency tests are applied by the data validation manager. Discrepancies that cannot be resolved are reported to the measurement laboratories for investigation. Data that deviate from consistency objectives are individually examined for errors. Obvious outliers (e.g., -85 °C temperature) are invalidated by setting a status flag. Changes to the data (e.g., swapping dates on consecutive samples) are recorded and documented by setting status flags. Level 1B data review is carried out using custom software developed for this purpose. IMPROVE level 1B data validation includes

- immediately verifying balance measurements to ensure that
 - the range is within specified limits
 - the postweight is greater than the preweight

- examining daily flow rates based on a report from the FlowRSD program that identifies flow rates with significant variations over 24 hours
- setting status flags when deviations from nominal operational settings have occurred (e.g., flow rates outside quantitative tolerances)
- examining the ion field blank and the carbon secondary filter analyses for evidence of swaps with sample filters
- examining individual data points identified by SwapCheck as potential sample swaps between two adjacent dates on module A, B, or C
- comparing the analytical data to expectations based on prior years

Level 2: Level 2 data validation takes place after data from various measurement methods have been assembled in the database. Level 2 tests involve comparisons of collocated measurements and internal consistency tests. The Level 2 data review is conducted site by site by the data validation manager on the final data set before it is submitted to CIRA. Data submitted to CIRA are considered to be validated at Level 1B and Level 2, depending on the specific types of checks applied to the data. Additional Level 2 data validation is performed by CIRA.

IMPROVE Level 2 data validation consists of a site-by-site examination of time series and scatter plots of data, including

- comparing sulfur and sulfate concentrations
- comparing light-absorbing carbon and organic mass by carbon to LRNC and organic mass by carbon
- examining PM_{10} mass and $PM_{2.5}$ mass for cases where $PM_{2.5}$ is greater than PM_{10}

Level 3: This level of data review is applied after the submission of concentration files to CIRA, when the results from different data analysis approaches are compared with each other and with measurements. At this level, the data are reconciled with other research findings, such as modeling results or theoretical predictions. The first assumption upon finding a measurement that is inconsistent with physical expectations is that the unusual value is due to a measurement error. If, upon tracing the path of the measurement, nothing unusual is found, the value can be assumed to be a valid result of an environmental or statistical cause. Level 3 validation continues for as long as the database is maintained.

5. DATA PROCESSING

The following section is a discussion of the equations used to calculate aerosol concentrations and the associated uncertainty and mdl. Except for the flow rate calculations, all these calculations are performed by BuildDBF to generate the output data file. Appendix A lists the data dictionaries for all files needed by BuildDBF. Appendix B contains flow charts for the BuildDBF procedures. Appendix C lists the FoxPro source code for BuildDBF. Appendix D lists the FoxPro source code for SWAP, and Appendix E lists the FoxPro source code for Network_Metric_Checks. Appendix F gives more detail about the origins of the flow rate equations.

5.1 ARTIFACTS

An *artifact* is defined as any increase or decrease of material on the filter that positively or negatively biases the measurement of ambient concentration. We make artifact corrections on the ion and carbon filter measurements. The five major types of artifacts are

- contamination of the filter medium
- contamination acquired by contact with the cassettes or in handling
- adsorption of gases during collection that are measured as particles
- volatilization during collection and in handling
- fall-off during handling after collection

The first three are positive artifacts and the last two are negative.

For the ions measurements, the first two types of artifacts are assumed to be the most significant. The sum of the two contamination artifacts is estimated through the analysis of field blanks (FB status flag). The field blanks are handled as normal filters (loaded into cassettes and cartridges, shipped to and from the field, and left in the sampler for a week) except that no air is drawn through them. The field blanks are collected randomly at all sites on a periodic basis. The ions artifacts are calculated as the median loading measured on the nylon field blanks in the month. These artifact values are subtracted from each ambient concentration for the corresponding month. The standard deviation of the field blank loadings is used to calculate the mdl. Prior to June 2002, artifacts were calculated by seasonal quarter instead of by month.

For the carbon measurements, the first three types of artifacts are assumed to be the most significant. These artifacts are estimated using secondary filters (CS status flag). Secondary filters are loaded into the filter cassettes behind the primary filters at six sites: Chiricahua, Grand Canyon, Mount Rainier, Okefenokee, Shenandoah, and Yosemite. The assumption is that the first filter collects all the particles and absorbs some organic gases and the secondary filter absorbs an equivalent amount of organic gases. The carbon artifacts are calculated as the median value measured on the secondary filters each month. These artifact values are subtracted from each ambient concentration for the corresponding month. The standard deviation of the secondary filter loadings is used to calculate the mdl.

We do not correct any measurements for the two negative artifact types, volatilization and fall-off. The measured low-temperature organics may be much less than in the atmosphere because of volatilization of particles during the remainder of the sampling. We assume that any volatilization of nitrate and chlorine from nylon is not significant. Depending on the environmental conditions, some ammonium nitrate collected on Teflon® filters may volatilize. In those cases the fine mass on the Teflon® filter may underestimate the ambient mass concentrations.

5.2 VOLUME

The volume is the product of the flow rate and the sample duration. The sample duration is determined using elapsed time (ET), as recorded by the sampler controller on the memory card. The fractional uncertainty of the volume is the quadratic sum of the fractional uncertainties of flow rate and duration. Since the fractional uncertainty of the duration is much smaller than that of the flow rate, it is neglected.

For the PM_{2.5} modules, the flow rate is measured using two independent methods. The first method measures the pressure drop across the cyclone using a pressure transducer and employs a measuring orifice equation. The second method measures the pressure immediately before the critical orifice and employs the equation for flow through a critical orifice. If the cyclone transducer measurements are determined to be in error, the critical orifice transducer measurements are used. The equations for flow rate are described in more detail in Appendix F.

The uncertainty in the average flow rate has two components. The first component is the accuracy of the measured values. Internal audits indicate that the root mean square difference between an audit and the sampler measurement is 3-4%. Since the precision of most audit devices is 2-3%, the sampler flow rate uncertainty is estimated to be 3%.

The second component of flow rate uncertainty results from the averaging method employed because the equations used to calculate flow rate from the pressure measurements are not linear. Prior to January 2005, we averaged the 15-minute pressure measurements over the sample period (nominally 24 hours) and then calculated the flow rate from the average pressure measurements. The more accurate method is to calculate the flow rate for each 15-minute pressure measurement and then calculate an elapsed time-weighted average flow rate for the sampling period. If the sampler flow rate drops during sampling (due to filter clogging, for example), the two methods will produce different results (1-3% difference); otherwise they are nearly identical. We moved to the second method beginning with the January 2005 data.

The sampler flow rate for modules A, B, and C is calculated by

$$Q = 10^a M^b * F(elev) * \sqrt{\frac{T + 273.15}{293.15}} \quad (351-1)$$

where Q = volumetric flow rate (calculated from site-specific temperature and pressure, not STP)

a, b = calibration coefficients

M = cyclone transducer reading. If the transducer readings are taken off the controller screen, they can be plugged into equation 351-1 directly; if the transducer readings are taken off the flashcard file, they must be divided by 100.

F(elev) = elevation factor to account for pressure difference between sea level and the site (see Appendix F)

T = ambient temperature in degrees Celsius at time of sampling.

For module D, the sampler flow rate is calculated by

$$Q = (c + d * G) * F(elev) * \sqrt{\frac{T + 273.15}{293.15}} \quad (351-2)$$

where Q = volumetric flow rate

c, d = calibration coefficients

G = critical orifice transducer reading. If the transducer readings are taken off the controller screen, they can be plugged into equation 351-2 directly; if the transducer readings are taken off the flashcard file, they must be divided by 100.

$F(elev)$ = elevation factor to account for pressure difference between sea level and the site (see Appendix F)

T = ambient temperature in degrees Celsius at time of sampling.

Equation 351-2 may be used to calculate the flow rate on Modules A, B, or C if the cyclone transducer readings are invalid.

5.3 CONCENTRATION, UNCERTAINTY, AND MINIMUM DETECTABLE LIMIT (MDL)

The calculations described in this section are performed by BuildDBF. The mass of material on the filter is equal to the difference between the mass measured on the sample and the mass on the unused filter. The mass on the unused filter is determined from the preweight of each individual Teflon® filter for the gravimetric analysis, from the median of field blank loadings for the ion measurements, and from the median of the secondary filter loadings for the carbon measurements. The concentration equals this number divided by the volume:

$$C = \frac{A - B}{V} \quad (351-3)$$

where C = ambient concentration (ng/m³)

A = mass measured on sample (ng/filter or ng/cm²)

B = artifact mass (ng/filter) = preweight, monthly median of ion field blanks (FB), or monthly median of carbon secondary filters (CS). Prior to June 2002, artifacts were calculated by seasonal quarter instead of by month.

V = sample air volume (m³) = Q * Elapsed Time

The uncertainty is reported with each concentration. The general model for the uncertainty is a quadratic sum of three components of uncertainty as shown in Equation 351-4.

$$\sigma(c) = \sqrt{[f_v C]^2 + [f_a C]^2 + \left[\frac{\sigma_a}{V}\right]^2} \quad (351-4)$$

where f_v = fractional volume uncertainty. A value of 3% is used, based on audits.

f_a = fractional analytical uncertainty. This term results from uncertainty in the analytical calibration. This term is zero for gravimetric analysis. The values for the other methods were determined from historical replicate analyses.

σ_a = constant analytical uncertainty. This term results from uncertainty in either the preweight, artifact subtraction, or XRF peak counting uncertainty. For large concentrations, this is small compared to the fractional terms.

The equation becomes more complicated when blank subtraction is required, as for the ions and carbon measurements; details on the derivation of the uncertainty equations are contained in the Quality Assurance Project Plan (QAPP, 2002).

The minimum detectable limit (mdl) is also provided with each concentration in the database. For the ion and carbon measurements, the mdl corresponds to twice the standard deviation of the

field blanks and secondary filters, respectively. For mass and absorption, the mdl corresponds to twice the analytical uncertainty determined by controls. For PIXE, XRF, and PESA, the mdl is based on the background under the peaks in the spectrum and is calculated separately for each filter.

5.3.1 PM_{2.5} and PM₁₀ Mass (Module A)

PM_{2.5} mass is measured gravimetrically on the Teflon® filter from Module A. PM₁₀ mass is measured gravimetrically on the Teflon® filter from Module D. The postweight and preweight are stored as milligrams per filter in the weights file. The constant analytical uncertainty, σ_a , in equation 351-4 is equal to 5 μg for all filters, and the proportional analytical uncertainty is assumed to be zero. The mass concentration, uncertainty, and mdl in nanograms per cubic meter are calculated using the following equations:

$$C_{\text{Mass}} = 10^6 \frac{\text{ng}}{\text{mg}} * \left(\frac{\text{Postweight} - \text{preweight}}{V} \right) \quad (351-5)$$

$$\sigma_{\text{Mass}} = 1000 \frac{\text{ng}}{\mu\text{g}} * \sqrt{\left(\frac{5 \mu\text{g}}{V} \right)^2 + \left(\frac{C_{\text{Mass}} * f_v}{1000 \frac{\text{ng}}{\mu\text{g}}} \right)^2} \quad (351-6)$$

$$\text{mdl}_{\text{Mass}} = 1000 \frac{\text{ng}}{\mu\text{g}} * \frac{10 \mu\text{g}}{V} \quad (351-7)$$

5.3.2 Ions (Module B)

Ions are measured by ion chromatography using the nylon filter from Module B. The ions measurements—chloride (Cl^-), nitrite (NO_2^-), nitrate (NO_3^-), sulfate (SO_4^{2-}), and ammonium (NH_4^+)—are stored as micrograms per filter in the ions data files from RTI. The ion concentration, uncertainty, and mdl in nanograms per cubic meter are calculated using the following equations:

$$C_{\text{ion}} = 1000 \frac{\text{ng}}{\mu\text{g}} * \frac{(A_{\text{ion}} - B_{\text{ion}})}{V_{\text{B module}}} \quad (351-8)$$

$$\sigma_{\text{ion}} = 1000 \frac{\text{ng}}{\mu\text{g}} * \frac{\sqrt{\sigma_{\text{dfb}}^2 + 2 * B_{\text{ion}} * f_a^2 * (A_{\text{ion}} - B_{\text{ion}}) + (f_a * (A_{\text{ion}} - B_{\text{ion}}))^2 + (f_v * (A_{\text{ion}} - B_{\text{ion}}))^2}}{V_{\text{B module}}} \quad (351-9)$$

$$\text{mdl} = 1000 \frac{\text{ng}}{\mu\text{g}} * 2 * \frac{\text{Max}(\sigma_{\text{dfb}}, \text{mdl}_{\text{analytical}})}{V_{\text{B module}}} \quad (351-10)$$

where $\text{mdl}_{\text{analytical}} = 0.03$ for chloride, 0.01 for nitrite, 0.05 for nitrate, 0.07 for sulfate, and 0.12 for ammonium. The analytical mdl for each ion is to be used as the overall mdl in the event that the median value of the field blanks is lower than the respective analytical mdl.

σ_{dfb} = standard deviation of the field blank measurements for the appropriate ion.

The fractional analytical uncertainty, f_a , for each ion species is given in Table 1; these values have been used since at least February 28, 1995.

Table 1. Fractional analytical uncertainty for the ions

| Species | f_a |
|----------------|-------|
| Chloride | 0.04 |
| Nitrite | 0.02 |
| Nitrate | 0.03 |
| Sulfate | 0.02 |
| Ammonium | 0.02 |
| Sulfur Dioxide | 0.04 |

5.3.3 Carbon (Module C)

Carbon is measured by thermal optical reflectance (TOR) using the quartz filter from Module C. Carbon measurements are stored in the carbon file as micrograms per filter. For the eight carbon species, the primary source of fractional uncertainty is the separation into temperature ranges. This may be associated with temperature regulation, but it may also be from the inherent variability of the species involved. The concentration, uncertainty, and mdl in nanograms per cubic meter for the carbon fractions – O1, O2, O3, O4, OP, E1, E2, and E3 – are calculated using the following equations:

$$C_{fraction} = 1000 \frac{ng}{\mu g} * \frac{(A_{fraction} - B_{fraction})}{V_{C\ module}} \quad (351-11)$$

$$\sigma_{fraction} = 1000 \frac{ng}{\mu g} * \frac{\sqrt{\sigma_{secondary}^2 + 2 * B_{fraction} * f_a^2 * (A_{fraction} - B_{fraction}) + (f_a * (A_{fraction} - B_{fraction}))^2 + (f_v * (A_{fraction} - B_{fraction}))^2}}{V_{C\ module}} \quad (351-12)$$

$$mdl_{fraction} = 1000 \frac{ng}{\mu g} * 2 * \frac{\sigma_{secondary}}{V_{C\ module}} \quad (351-13)$$

where $\sigma_{secondary}$ = standard deviation of the secondary (backup) filter measurements for the appropriate carbon fraction.

The fractional analytical uncertainty, f_a , for each carbon species is given in Table 2. These values have been used since at least February 28, 1995.

Table 2. Fractional analytical uncertainty for the carbon species

| Species | f _a |
|---------|----------------|
| O1 | 0.27 |
| O2 | 0.16 |
| O3 | 0.11 |
| O4 | 0.13 |
| OP | 0.27 |
| E1 | 0.13 |
| E2 | 0.26 |
| E3 | 0.40 |

5.3.4 SO₂

SO₂ measurements ceased in early 2008. SO₂ was collected at only a few sites in the network using a separate module containing quartz filters impregnated with potassium carbonate. A Teflon® prefilter removes particulate sulfate so that all the sulfate measured on the filter can be attributed to SO₂. The analysis is by ion chromatography. SO₂ measurements are returned from the analysis contractor as micrograms of sulfate per filter. The SO₂ concentration, uncertainty, and mdl are currently not calculated. They would be calculated using the following equations:

$$C_{SO_2} = 1000 \frac{ng}{\mu g} * \frac{2}{3} * \frac{(A_{SO_2} - B_{SO_2})}{V_{D\ module}} \quad (351-14)$$

$$\sigma_{SO_2} = 1000 \frac{ng}{\mu g} * \frac{2}{3} * \frac{\sqrt{\sigma_{dfb}^2 + 2 * B_{SO_2} * f_a^2 * (A_{SO_2} - B_{SO_2}) + (f_a * (A_{SO_2} - B_{SO_2}))^2 + (f_v * (A_{SO_2} - B_{SO_2}))^2}}{V_{D\ module}} \quad (351-15)$$

$$mdl = 1000 \frac{ng}{\mu g} * 2 * \left(\frac{2}{3}\right) * \frac{\sigma_{dfb}}{V_{D\ module}} \quad (351-16)$$

The factor of 2/3 in the above equations converts the analyzed sulfate to SO₂. The variable σ_{dfb} refers to the standard deviation of the field blank measurements for SO₂ (as sulfate).

5.3.5 Elements (Module A)

Elements are measured using X-ray fluorescence (XRF) and proton elastic scattering analysis (PESA) on the Teflon® filters from Module A. Prior to December 2001, particle induced x-ray emission (PIXE) was used to analyze the light elements (sodium through manganese) and XRF was used to analyze the heavier elements (iron through lead). Starting in December 2001 all elements, except hydrogen, are analyzed with XRF systems. Elemental values are stored in the elements file as nanograms per square centimeter of aerosol deposit. The uncertainty and mdl are calculated by the XRF processing software (ACE and RACE) and are stored in the data file along with the areal densities. The artifact values (B in equation 351-3) are assumed to be zero for all elements. The elemental concentration, uncertainty, and mdl are calculated using the following equations:

$$C_{element} = \frac{A_{element} * (Deposit\ area)}{V} \quad (351-17)$$

$$\sigma_{element} = \frac{U_{element} * (Deposit\ area)}{V} \quad (351-18)$$

$$mdl_{element} = \frac{M_{element} * (Deposit\ area)}{V} \quad (351-19)$$

where $A_{element}$ = areal concentration of the element measured by XRF

Deposit area = area of sample deposit on the filter (cm²), determined from the filter holder or mask size

$U_{element}$ = areal uncertainty reported for the element measured by XRF

$M_{element}$ = areal mdl reported for the element measured by XRF.

There are some elemental interferences in the XRF measurements; most of the interferences are addressed by the XRF system software but the sulfur-lead and sulfur-aluminum interferences are addressed by the BuildDBF software because the elements in these pairs are measured on two separate XRF systems. The measurement for sulfur requires subtracting 74% of the value for lead because of an x-ray overlap between a minor x-ray emission of lead and the major x-ray emission of sulfur. A similar situation exists for aluminum, which requires subtracting 62% of the value for bromine from aluminum. The mdl for sulfur and aluminum are not adjusted because the correction is very small.

5.3.6 Laser Absorption (Module A)

Optical absorption is measured by a hybrid integrating plate and sphere using the Teflon® filter from Module A. The laser absorption measurements are stored in the laser file as reflectance (R) and transmittance (T) values. The LRNC values are calculated using the following equations:

$$LRNC = 10000 * \text{Log}\left(\frac{1000 - R}{T}\right) * \frac{(Deposit\ area)}{V} \quad (351-20)$$

$$\sigma_{LRNC} = \sqrt{\left(\frac{(10000 * 0.045)}{2 * V / Deposit\ area}\right)^2 + (f_v * LRNC)^2} \quad (351-21)$$

$$mdl = 10000 * 0.045 * \left(\frac{Deposit\ area}{V}\right) \quad (351-22)$$

5.4 EQUATIONS OF COMPOSITE VARIABLES

The following composite variables are combinations of the measured concentrations of particles collected on the fine filters. These are used in the Level 2 validation procedures, described in Section 6.0. The concentration is determined along with the uncertainty and mdl. In our calculations of the uncertainty of the composite variables, we assume that the component concentrations are independent and the multiplicative factors have no uncertainty. The

independence assumption is not strictly valid for many composites because of common factors, such as volume. However, the effect on the overall uncertainty is too small to warrant the more complicated calculations. The following sections are divided by composite variable.

5.4.1 Sulfate by XRF (S3) and Ammonium Sulfate (NHSO)

Sulfur is predominantly present as sulfate in the atmosphere. To compare the sulfur by XRF and the sulfate by ion chromatography, the XRF concentration is multiplied by the ratio of sulfate to sulfur atomic mass ($96.06/32.06 = 3.0$). This composite is labeled S3 in the data validation file.

The sulfate is generally present as ammonium sulfate, $(\text{NH}_3)_2\text{SO}_4$, although it can be present as ammonium bisulfate, $(\text{NH}_3)\text{HSO}_4$, sulfuric acid, H_2SO_4 , and, in marine areas, as sodium sulfate, NaSO_4 . In many cases, the particle will include associated water, but we omit this from the calculation. In order to simplify the calculation we assume all the sulfur is present as ammonium sulfate. The concentrations, uncertainties, and mdl's are given by

$$NHSO = 4.125 * S \qquad S3 = 3 * S \qquad (351-23)$$

$$\sigma_{NHSO} = 4.125 * \sigma(S) \qquad \sigma_{S3} = 3 * \sigma(S) \qquad (351-24)$$

$$mdl(NHSO) = 4.125 * mdl(S) \qquad mdl(S3) = 3 * mdl(S) \qquad (351-25)$$

If S is below the mdl, both S and $\sigma(S)$ are set equal to $mdl/2$ in these calculations. For ammonium bisulfate, sulfuric acid, and sodium sulfate the factors are 3.59, 3.06, and 4.43, respectively. In the first two cases, the actual dry mass associated with sulfate will be less than NHSO, and in the third case, more.

5.4.2 Ammonium Nitrate (NHNO)

This composite is the total dry concentration associated with nitrate, assuming 100% neutralization by ammonium. For all sites, we define the ammonium nitrate concentration and uncertainty by

$$NHNO = 1.29 * NO_3^- \qquad (351-26)$$

$$\sigma_{NHNO} = 1.29 * (\sigma_{NO_3^-}) \qquad (351-27)$$

$$mdl(NHNO) = 1.29 * mdl(NO_3^-) \qquad (351-28)$$

5.4.3 Soil

The soil component consists of the sum of the predominantly soil elements measured by XRF, multiplied by a coefficient to account for oxygen for the normal oxide forms (Al_2O_3 , SiO_2 , CaO , K_2O , FeO , Fe_2O_3 , TiO_2), and augmented by a factor to account for other compounds not included in the calculation, such as MgO , Na_2O , water, and CO_2 . The following assumptions are made:

- Fe is split equally between FeO (oxide factor of 1.29) and Fe_2O_3 (oxide factor of 1.43), giving an overall Fe oxide factor of 1.36.
- Fine K has a non-soil component from smoke. Based on the K/Fe ratio for average sediment (*Handbook of Chemistry and Physics*) of 0.6, we use $0.6 * \text{Fe}$ as a surrogate for soil K and then add the oxide factor for K $\left(\text{K}_2\text{O}, \frac{39.1 * 2 + 16.0 \text{ g/mol}}{39.1 * 2 \text{ g/mol}} = 1.2 \right)$ to

get a total Fe factor of $0.72 * Fe$ ($0.6 * 1.2$) for the potassium oxide in soil. This increases the factor for Fe from 1.36 to 2.08.

- Because aluminum has not always been detected with as high a frequency as the other major soil elements, we have eliminated Al from the soil parameter. Fortunately, the correlation between Al and Si has a relatively constant ratio of Al/Si of 0.45 at all sites with the exception of the Virgin Islands site, where the Al/Si ratio is 0.60; the soil parameter is then low by 6%. With the oxide factors for Al_2O_3 and SiO_2 the multiplicative factor for Si increases from 2.14 to 2.99.
- The oxide forms of the soil elements account for 86% of average sediment; in order to obtain the total mass associated with soil, the final factors are divided by 0.86 (*Handbook of Chemistry and Physics*). The final equations for fine soil concentrations and uncertainty are:

$$SOIL = 3.48 * Si + 1.63 * Ca + 2.42 * Fe + 1.94 * Ti \quad (351-29)$$

$$\sigma(SOIL) = \sqrt{3.48 * \sigma(Si)^2 + 1.63 * \sigma(Ca)^2 + 2.42 * \sigma(Fe)^2 + 1.94 * \sigma(Ti)^2} \quad (351-30)$$

$$mdl(SOIL) = 0 \quad (351-31)$$

The soil variable is calculated for all valid XRF analyses. If an element is below the mdl, the concentration and uncertainty are both assumed to be equal to mdl/2. The soil variable is always positive. Note that the soil equation (351-29) is used only at CNL for data validation. The Regional Haze Rule uses a different form of the soil equation that includes aluminum.

5.4.4 Non-soil Potassium (KNON)

Non-soil potassium is the measured fine potassium minus the soil potassium estimated from iron. Non-soil potassium is a qualitative tracer of smoke. The problem is that the ratio of potassium/smoke mass may change as the aerosol ages. Particulate smoke potassium is probably produced by the transformation of volatilized potassium, and appears to be in a smaller size range than most smoke mass. Close to the smoke source, the particulate potassium may not have time to form. For long-range transport, most other smoke mass may settle out more than potassium mass. The concentration and uncertainty of KNON are

$$KNON = (K - 0.6 * FE) \quad (351-32)$$

$$\sigma(KNON) = \sqrt{\sigma^2(K) + [0.6 * \sigma(Fe)]^2} \quad (351-33)$$

$$mdl(KNON) = 0 \quad (351-34)$$

The soil factor of 0.6 may vary slightly with the site; this will produce a small positive or negative offset for baseline values when no smoke is present. Therefore, negative values are retained. KNON is defined for all valid XRF analyses. If a concentration is less than the mdl, the concentration and uncertainty are assumed to be equal to the mdl.

5.4.5 Light-Absorbing Carbon (LAC)

The concentration of carbon on quartz is determined in seven temperature ranges, plus a correction fraction for pyrolyzed organic carbon. The light-absorbing carbon (LAC) component is assumed to be all carbon evolved at 550°C and above after the laser indicates that reflectance has returned to the initial value. The equation for LAC concentration is

$$LAC = E1 + E2 + E3 - OP \quad (351-35)$$

LAC may be negative. The uncertainty of LAC is not the quadratic sum of the uncertainties of the fractions because the components are not independent. Based on replicate measurements, the following empirical formula is used to calculate the uncertainty:

$$\sigma(LAC) = \sqrt{(34)^2 + (0.07 * LAC)^2} \text{ ng/m}^3 \quad (351-36)$$

$$mdl(LAC) = 0 \quad (351-37)$$

The constant term (34) is appropriate for volumes near 32.4 m³, which is typical for 24-hour IMPROVE samples. For other volumes they should be multiplied by (32.4/V).

5.4.6 Organic Mass by Carbon (OMC)

The organic carbon (OC) component is assumed to be all carbon evolved at 550°C and below in a pure helium environment, plus the pyrolyzed organic fraction. To determine the total amount of organic mass associated with the organic carbon, we assume a ratio of organic mass to organic carbon of 1.4. The equation for organic mass concentration is

$$OMC = 1.4 * (O1 + O2 + O3 + O4 + OP) \quad (351-38)$$

OMC may be negative. The uncertainty of OMC is not the quadratic sum of the uncertainties in the fractions because the fractions are not independent. Based on replicate measurements, the following empirical formula is used to calculate the uncertainty:

$$\sigma(OMC) = \sqrt{(168)^2 + (0.05 * OMC)^2} \text{ ng/m}^3 \quad (351-39)$$

$$mdl(OMC) = 0 \quad (351-40)$$

The constant term (168) is appropriate for volumes near 32.4 m³, which is typical for 24-hour samples. For other volumes they should be multiplied by (32.4/V).

5.4.7 Organic Mass by Hydrogen (OMH)

The organic mass can also be estimated from the concentrations of H and S measured on the Teflon® filter. The assumptions are

- All sulfur is present as fully neutralized ammonium sulfate. This is reasonable at all sites except those with high sulfur east of the Mississippi and at marine sites with significant marine sulfate.
- All nitrates and water volatilize during exposure to vacuum, so the PESA hydrogen comes only from sulfates and organics.
- The amount of hydrogen per organic particle is relatively constant throughout both space and time. We assume a ratio of 13.75 organic carbon mass to hydrogen mass.

At sites where the above assumptions are reasonable, the resulting estimate is in good agreement with organic mass by carbon (OMC).

The equations for OMH concentration, uncertainty, and mdl are

$$OMH = 13.75 * (H - 0.25 * S) \quad (351-41)$$

$$\sigma(OMH) = 13.75 * \sqrt{\sigma^2(H) + [0.25 * \sigma(LAC)]^2} \quad (351-42)$$

$$mdl(OMH) = 0 \quad (351-43)$$

If either H or S is invalid or not detected, OMH is not calculated. OMH may be negative when the sulfate is large and not fully neutralized.

5.4.8 Reconstructed Mass from the Teflon® filter (RCMA)

The reconstructed mass is the sum of sulfate, soil, non-sulfate potassium, salt, elemental carbon, and organic carbon. The only components not included are water and nitrate. Some nitrate particles may volatilize from the Teflon® filter during collection and transportation, and thus may not be measured by gravimetric analysis. The nitrate measured in the IMPROVE system is collected on nylon filters and does not volatilize during collection. Therefore, we have chosen to exclude the measured nitrate from the reconstructed mass.

We use two estimates of reconstructed mass: RCMA, based only on the Teflon® filter, using LRNC for elemental carbon and OMH for organic mass; and RCMC, based partly on the carbon measurements from the quartz filter and partly on measurements from the Teflon® filter. The equations for RCMA concentration and uncertainty are

$$RCMA = NSHO + SOIL + 1.4 * KNON + 2.5 * Na + LRNC + OMH \quad (351-44)$$

$$\sigma(RCMA) = \sqrt{\sigma(NHSO)^2 + \sigma(SOIL)^2 + [1.4 * \sigma(KNON)]^2 + [2.5 * \sigma(Na)]^2 + \sigma(LRNC)^2 + \sigma(OMH)^2} \quad (351-45)$$

$$mdl(RCMA) = 0 \quad (351-46)$$

For RCMA to be calculated, XRF, PESA, and HIPS must all have valid analyses. The 1.4 factor for KNON ensures consistency with the soil estimate; it exactly compensates for the removal of non-soil K from soil. Thus potassium is included in the sum as K₂O, whether as soil or as smoke. The sea salt (NaCl) concentration is estimated solely from Na, because some of the chlorine may be volatilized from the Teflon® filter during collection. If Na or LRNC are below the mdl, mdl/2 is used as both concentration and uncertainty. All values are set to zero if negative. RCMA will always be positive.

5.4.9 Reconstructed Mass using Carbon Measurements (RCMC)

An alternative estimate for reconstructed mass is based on organics and elemental carbon from the quartz filter. As in RCMA, the sum does not include water and nitrates. The equations for RCMC are

$$RCMC = NHSO + SOIL + 1.4 * KNON + 2.5 * Na + LAC + OMC \quad (351-47)$$

$$\sigma(RCMC) = \sqrt{\sigma(NHSO)^2 + \sigma(SOIL)^2 + [1.4 * \sigma(KNON)]^2 + [2.5 * \sigma(Na)]^2 + \sigma(LAC)^2 + \sigma(OMC)^2} \quad (351-48)$$

$$mdl(RCMC) = 0 \quad (351-49)$$

For RCMC to be calculated, XRF and carbon combustion must both have valid analyses. If LAC or OMC is negative, a value of zero is used. RCMC will always be positive.

In general, RCMA is preferred during periods of low organic concentrations, such as winter in western United States. RCMC is more relevant at sites where the neutralization of sulfate may be less than 100%, at sites with high nitrate, and at marine sites.

6. DATA VALIDATION

Data validation involves assessing the quality, reliability, and integrity of the data. It occurs throughout the measurement process. This section explains the components of the data validation process that occur once the data are received from the laboratory and field.

6.1 DEFINITION OF STATUS FLAGS

Standardized abbreviations describing the status of individual results in samples, known as status flags, are assigned during the Level 1A and 1B validation processes. The abbreviations and their meanings are listed in Table 3. Those listed as “terminal” flags indicate that the sample could not be processed to a valid result. “Temporary” flags are replaced before final data reporting.

Table 3. Status flags and their definitions

| Abbreviation | Meaning | Result |
|--------------|-------------------------|-----------|
| BI | Bad Install | terminal |
| CG | Clogging | |
| CL | Clogged | terminal |
| EP | Equipment Problem | terminal |
| LF | Low Flow | |
| NA | Not Applicable | |
| NM | Normal | |
| NR | Not Reanalyzed by DRI | |
| NS | No Sample | terminal |
| OL | Off Line | terminal |
| PO | Power Outage | terminal |
| QA | Quality Assurance | temporary |
| QC | Quality Control | temporary |
| QD | Questionable Data | temporary |
| RF | Really Low Flow Rate | |
| SA | Sampling Anomaly | |
| SP | Field Blank/Sample Swap | |
| SW | Swapped Sample Dates | |
| UN | Undetermined Weight | |
| XX | Destroyed/No Filter | terminal |

6.2 LEVEL 1 VALIDATION TESTS

Level 1 validation is conducted throughout the sample handling process. It refers to quality assurance performed on measured or derived data in the weights file, the logs file, and the element, ions, and carbon files. Level 1A validation is conducted as soon as samples return to the laboratory by review of log sheets and flashcard data files. Level 1B validation is conducted numerous times throughout sample handling and processing. During preweight, range checking is performed by the balance program to ensure that only one filter is weighed at a time. During postweight, the balance program warns the operator if the net weight is greater than 1 mg.

6.2.1 Validation of laboratory analyses

Level 1 data validation for the ion, carbon, XRF, and SO₂ analyses is done by the laboratory responsible for the analysis. The contractors' quality assurance procedures are recorded at http://vista.cira.colostate.edu/improve/Publications/IMPROVE_SOPs.htm.

Starting in June 2004, a new Level 1B data validation technique has been applied to the data to help identify analytical problems before site-by-site validation is started. The technique compares the current month of data for the entire network to upper and lower limits. The limits are based on data from the same month in prior years. Network-wide percentiles (medians and 90th percentiles) and maximum values are calculated for each measured parameter and are compared to the limits by a Visual FoxPro-based computer program called "Network Metric Checks." The source code for this program is in Appendix E. For parameters that are greater than zero more than half the time, the limits are set on the median value. For parameters that are less than zero more than half the time, the limits are set on the 90th percentile value. An upper bound is also set on the maximum value for each parameter. In addition, the artifact correction values are determined and compared to limits based on previous values.

The program outputs a table of the parameters that failed one or more of the checks and the corresponding percentiles and limits. The failures are scrutinized to determine if a real event (e.g., new urban site, forest fire) or an analytical problem caused the failure. If necessary, the failures are discussed with the laboratory personnel to determine if procedural changes may have caused the failures.

Work is underway to extend this data validation technique to the site-level data.

6.3 FLOW RATE VALIDATION

Starting with May 2004 data, the data validation manager has looked for flow rate inconsistencies using the option FlowRSD, accessed using the FoxPro program SWAP. The FoxPro source code for SWAP is in Appendix D

Inconsistencies are identified as a variation in flow rate over the 24-hour testing period of greater than 3%. The program, however, flags all inconsistencies, reporting even small deviations regardless of whether they result from a problem with the equipment or from a steady drop in flow rate due to filter loading. The data validation manager looks at each day reported as having inconsistencies, to determine whether the data reported for that day can be considered valid and whether the sampling occurred for the required eighteen hours. The data are flagged if necessary.

6.3.1 Automated flow rate validation

The flow rate validation procedure uses four flow-related flags. The flags used for PM_{2.5} data are listed in Table 4, along with the criteria for applying them. The criteria are applied using computer code integrated into the flashcard database. The PM_{2.5} flow rate criteria for the ‘LF’ flag were revised following the 2006 cyclone characterization tests, which determined that the original Water John equations are applicable to the IMPROVE cyclones. The revised ‘LF’ criteria, listed in Table 4, were applied beginning with the January 2005 samples.

The criteria are based on calculation limitations, performance testing, and cut point requirements. The criterion for applying a CL flag, PM_{2.5} flow rate less than 15 L/min for more than 1 hour, is based on the fact that the flow rate equation is inaccurate below approximately 15 L/min. Figure 6 shows a typical relationship between flow rate and pressure transducer measurements. The dashed line shows the calculated flow rate and the solid line shows the actual flow rate. This figure illustrates that below approximately 15 L/min, the calculated flow rate is not accurate. Only four 15-min flow rates below 15 L/min will be allowed before the air quality data are invalidated (replaced with -999) and flagged as CL. The cyclone cut point is 3.6 µm at 15 L/min.

The criterion for applying a CG flag (PM_{2.5} flow rate less than 18 L/min for more than 1 hour) is based on cut point characterization of the cyclone. The cut point is 3.0 µm at 18 L/min. Only four 15-minute flow rates below 18 L/min will be allowed before the sample is flagged as CG.

Table 4. Flow rate related validation flag definitions and application criteria (John and Reischl, 1980).

| Validation Flag | Definition | Concentration Reported? | Criteria for Application for PM _{2.5} Samples |
|-----------------|-----------------------|-------------------------|---|
| CL | Clogged Filter | No | Flow rate < 15 L/min for more than 1 hour ¹ |
| CG | Clogging Filter | Yes | Flow rate < 18 L/min for more than 1 hour |
| LF | Low/high flow rate | Yes | Average flow rate < 19.7 L/min or > 24.1 L/min ² |
| RF | Really high flow rate | Yes | Average flow rate > 27 L/min ³ |

¹ A small number of filters (5-15) in each quarter may pass this criterion but be classified as clogged (CL) after visual inspection of the flow rate and concentration data.

² Prior to January 2005 samples, the PM_{2.5} LF flow rate limits were 21.3 and 24.3 L/min.

³ Prior to January 2005 samples, PM_{2.5} samples with flow rates less than 19 L/min were flagged RF.

The criterion for applying the LF flag is based on defining a range of particle cut points for a sample to be labeled as PM_{2.5}. The new cyclone characterization tests show that the LF flag is applied when the flow rates yield cut points outside 2.25 and 2.75 µm. The LF flag limits used prior to January 2005 yielded cut points outside 2.23 and 2.54 µm.

The criterion for applying the RF flag (flow rate greater than 27 L/min) is based on the cut point of the cyclone. The cut point is not accurately known because it isn't characterized beyond 26 L/min and is below 2 μm . In addition, as with the CL flag, the flow rate equation is less accurate at 27 L/min.

A similar set of flags is applied to the PM_{10} data, but with several differences, due principally to the lower flow rate at which the PM_{10} sampler operates. The PM_{10} average flow rate limits for the LF flag are > 19 L/min and < 15 L/min, and the limits for the RF flag are > 21 L/min and < 13 L/min. The CG flag is applied if the average flow rate is < 15 L/min, but only if logs values are used to determine the flow rate; if flashcard data are used, then an LF or RF flag is applied. There are no automated criteria for applying a CL flag to PM_{10} data. CL flags are assigned during data validation at the discretion of the data analyst, based on graphical review of the flow rate behavior over the 24-hour period of the sample.

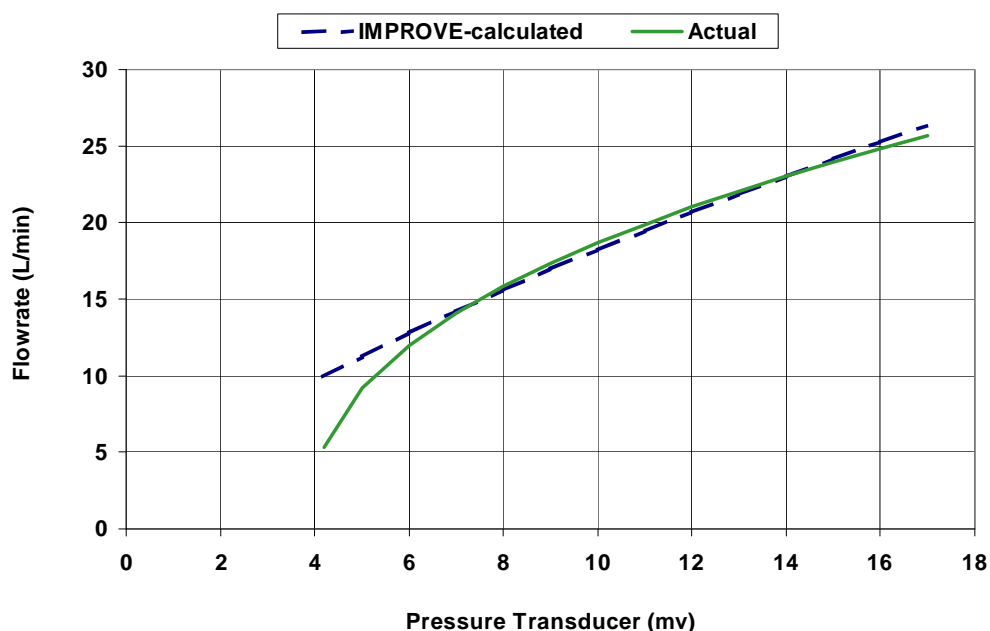


Figure 3. Flow rate versus the pressure transducer reading for the IMPROVE-calculated flow rate and the actual flow rate.

6.4 LEVEL II VALIDATION PROCEDURES

Level II validation is performed by comparing elements or species among modules using the Excel spreadsheet "QA Data.xls". A new spreadsheet is created for each 3-month season as the data are validated. The B (sulfates and nitrates), C (organic and inorganic carbon), and D (PM_{10}) modules measure one or more species that can be compared to those measured by module A (H plus most elements from Na through Pb). The following sampler module comparisons provide information on the quality of the reported data on a site-by-site basis:

6.4.1 Module A versus Module B data

Quality assurance for the A and B modules consists of comparing the measured concentrations of sulfur and sulfate. Sulfur concentrations are reported through elemental analysis of the Teflon®

filter from Module A, while sulfate concentrations are determined by ion chromatography analysis of the nylon filter from Module B. Since both modules sample simultaneously and have the same nominal flow rate and aerosol size cut point, the collected data should correlate.

The SwapCheck button on the SWAP program prompts the analyst for a month and year to examine. It then opens the data file for that time period and examines the data for each site for potential swaps. Successive pairs of data are examined using the algorithm outlined below. In the following equations, the subscripts refer to successive time periods for S3 or SO4. The ratio of S3/SO4 is calculated for each time period, and the ratio is also calculated assuming the sulfate values are swapped. Two indices of a swap are then calculated:

$$Index1 = \frac{\left(\frac{S3_1}{SO4_1} - 1\right)}{\left(\frac{S3_2}{SO4_2} - 1\right)} \text{ and } Index2 = \frac{\left(\frac{S3_1}{SO4_2} - 1\right)}{\left(\frac{S3_2}{SO4_1} - 1\right)}. \quad (351-50)$$

Normally, the corresponding values of S3 and SO4 are very close to each other so their ratio is close to unity. If the data are not subject to a swap, Index1 is close to zero and Index2 is large (and may be either positive or negative). If the S3/SO4 ratio is less than 0.667 or greater than 1.5 the date is flagged for a closer look. The more important indicator of a potential swap is when $Index1 < -0.03$ and $ABS(Index2) < 0.05$. The thresholds of -0.03 and 0.05 have been set empirically. These dates are also flagged in a report for the data validation analyst.

All outlier points are carefully reviewed for flow rate entry errors or analytical errors. If no reason for a discrepancy can be found the data are flagged as normal. Adjustments are made (e.g., elapsed time, flow rate, or terminal flags) only if a definitive cause can be found to support them. Figure 4 shows typical time series and scatter plots of sulfur (*3) and sulfate that are used to look for discrepancies between Module A and Module B. These plots are color-coded in conformance with the color scheme assigned to sampling modules. The red line in the time series and the red axis in the scatter plot corresponds to sulfur (S3=sulfur*3) measured on Module A, while the yellow line and axis corresponds to sulfate measured on Module B. The green dots are the ratio of $S3/SO_4^-$, and are used to look for systematic biases between the two measurements. A systematic bias would reveal itself in the ratio being consistently above or below unity. Ratios within the range 0.8-1.2 are generally acceptable, but if all are high or low it may indicate an error in flow rate for Module A or Module B. The blue dashed line is a calculation of implied sulfate based on the measurements of OMC and H. It is used solely as a guide.

In Figure 4 all but one of the S3 and SO_4^- pairs are in agreement. On 1/25 the sulfate is zero because the B module did not operate. The scatter plot in Figure 4 shows that the regression slope is between 0.95 and 1.05, which is within our acceptance criteria. The black line includes a floating intercept, and the blue line is forced through a zero intercept, and the red line is just a regression of the latest month of data.

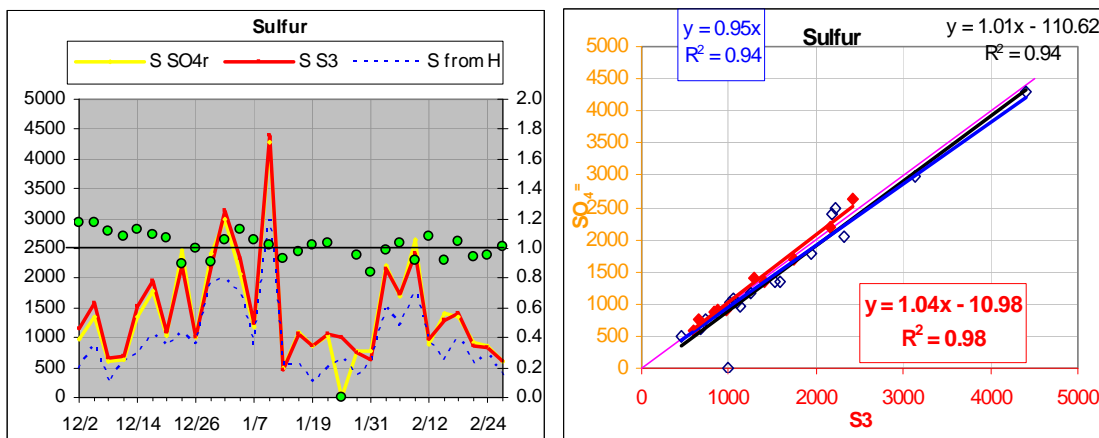


Figure 4. Time series and scatter plot of sulfur (*3) and sulfate for a single site

6.4.2 Module A versus Module C data

Quality assurance for the A and C modules involves correlation plots of four species, two from each sampling module. The Module A measurements are color-coded in red, while the Module C measurements are color-coded in green. The laser absorption (LRNC) measurement on Module A is compared to light absorbing carbon (LAC) on Module C. The organic mass by hydrogen (OMH) measured by PESA on Module A is compared to organic mass by carbon (OMC) measured on Module C.

The SwapCheck button on the SWAP program prompts the analyst for a month and year to examine. It then opens the data file for that time period and examines the data for each site for potential swaps. Successive pairs of data are examined using the algorithm outlined below. In the following equations, the subscripts refer to successive time periods for the sum of OMH and LRNC (CA – two measures of carbon by A module) and OMCN and LACN (CC – two measures of carbon by C module). The ratio of CA/CC is calculated for each time period, and the ratio is also calculated assuming the sulfate values are swapped. Two indices of a swap are then calculated:

$$Index1 = \left(\frac{CA_1}{CC_1} - 1 \right) / \left(\frac{CA_2}{CC_2} - 1 \right) \text{ and } Index2 = \left(\frac{CA_1}{CC_2} - 1 \right) / \left(\frac{CA_2}{CC_1} - 1 \right). \quad (351-51)$$

Normally, the corresponding values of CA and CC are somewhat close to each other so their ratio is close to unity. If the data are not subject to a swap, Index1 is close to zero and Index2 is large (and may be either positive or negative). If the CA/CC ratio is less than 0.5 or greater than 4 the date is flagged for a closer look. The more important indicator of a potential swap is when $Index1 < -1$ and $ABS(Index2) < 0.05$. The thresholds of -1 and 0.05 have been set empirically. These dates are also flagged in a report for the data validation analyst.

Figure 5 shows the time series and scatter plot of laser absorption (LRNC) and light absorbing carbon (LAC). The red line and axis correspond to LRNC on Module A and the green line and axis correspond to LAC. The yellow dots show the ratio of LRNC/LAC that is used to look for systematic biases in the two measurements. The ratio should be close to unity for all data points, although there is considerable scatter for these parameters. If all ratios are much greater than one

or much less than one, underlying causes are sought. A possible cause of this is a systematic error in flow rate.

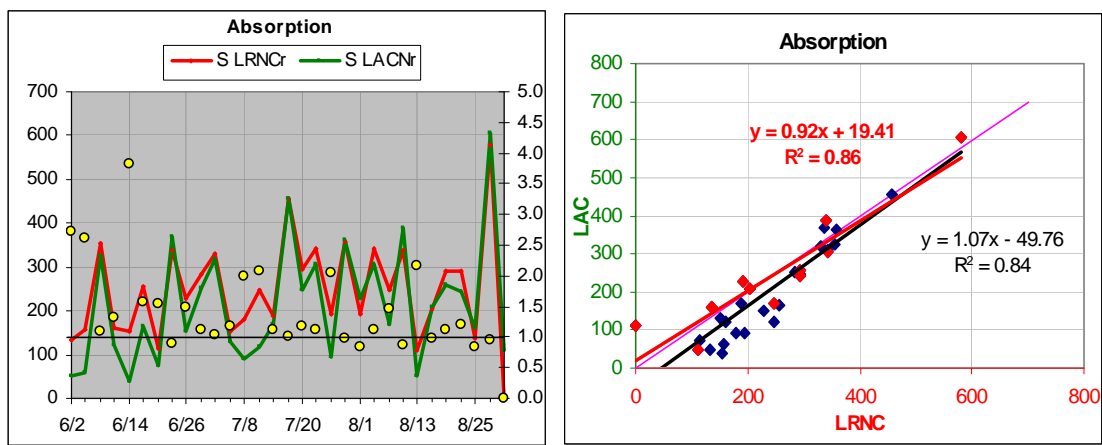


Figure 5. Time series and scatter plot of laser absorption (LRNC) and light absorbing carbon (LAC) for a single site

Figure 6 shows the time series and scatter plot of organic mass by hydrogen (OMH) and organic mass by carbon (OMC). OMH is a result of the PESA analysis on Module A, while OMC is from Module C. These measurements are typically in closer agreement than those in Figure 5, but not as close as sulfur vs. sulfate. The red line and axis correspond to OMH, and the green line and axis correspond to OMC. The yellow dots indicate the ratio of OMH/OMC and are used to look for systematic biases between the two measurements. The ratio should be close to unity for all data points, although there is considerable scatter for these parameters. If all ratios are much greater than one or much less than one, underlying causes are sought. A possible cause of this is a clogged inlet on one of the modules. The solid yellow line is a calculation of organic mass from the H measurement, which subtracts the hydrogen associated with nitrate OMH concentration. This is significant at a few sites in the network where nitrate concentration is high.

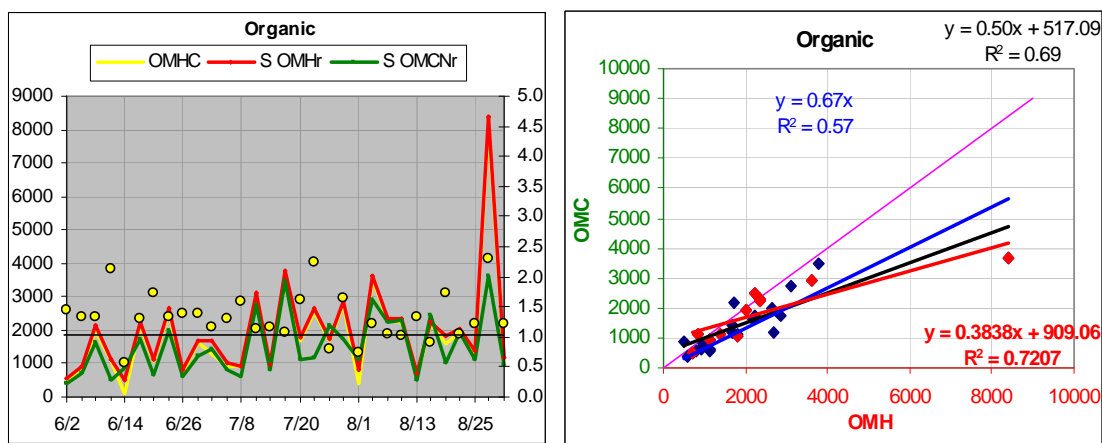


Figure 6. Time series and scatter plot of organic mass by hydrogen (OMH) and organic mass by carbon (OMC) for a single site

Generally, an error observed in one pair of plots is also observed in the other. Exceptions may occur, though, because the measurements on Module A (LRNC and OMH) are performed on different analytical instruments.

6.4.3 Module A versus Module D data

Quality assurance for the A and D modules consists of site-by-site comparisons of the PM_{2.5} mass concentration and the PM₁₀ mass concentration. This procedure is done to verify that no PM_{2.5} mass values are larger by two standard deviations than the corresponding PM₁₀ mass values and as another check of the sampler flow rates. Because MF is a subset of MT these cases should not occur or should be within the uncertainty in both measurements. Although the ratio of PM_{2.5} mass to PM₁₀ mass is fairly consistent at most sites, the correlation plot is meant only to verify that no PM_{2.5} mass values are larger than the corresponding PM₁₀ mass values.

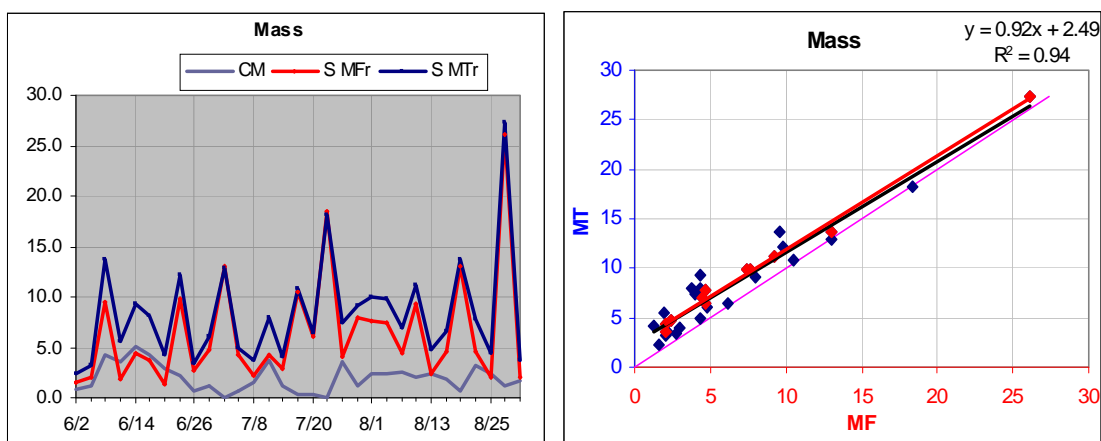


Figure 7. Time series and scatter plot of PM₁₀ mass and PM_{2.5} mass for a single site. The difference between the two [coarse mass (CM)] is also shown with the gray line.

Figure 7 shows the measured PM₁₀ (MT) and PM_{2.5} (MF) mass in a time series and scatter plot. All cases of MF>MT are investigated to determine whether a weighing error has occurred and to correct it if one is found.

6.5 RECONSTRUCTED MASS VERSUS MEASURED MASS (MF)

The reconstructed mass (RCMC) is calculated by equation 351-47. The reconstructed mass is compared to the measured mass as a check of all measured species. Figure 8 shows the time-series and scatter plots for a site where nitrate is not high. The reconstructed mass accurately accounts for the measured mass. Both the time-series and scatter plot show excellent correlation and equivalent values. Figure 9 shows the same plot for a site where nitrate is a significant fraction of the measured mass. The RCMC accounts for only about 60% of the measured mass at this site. Adding ammonium nitrate to the RCMC, where ammonium nitrate = 1.29*nitrate, gives RCMN=RCMC+1.29*NO₃. RCMN, the reconstructed mass including nitrate, is very close to the measured mass.

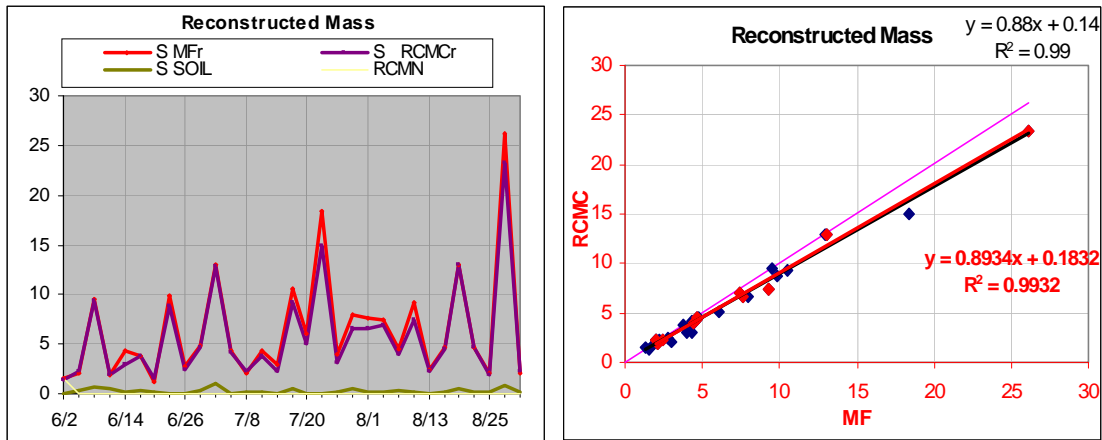


Figure 8. Time series and scatter plot of reconstructed mass (RCMC) and measured fine mass (MF). In this case there is excellent agreement.

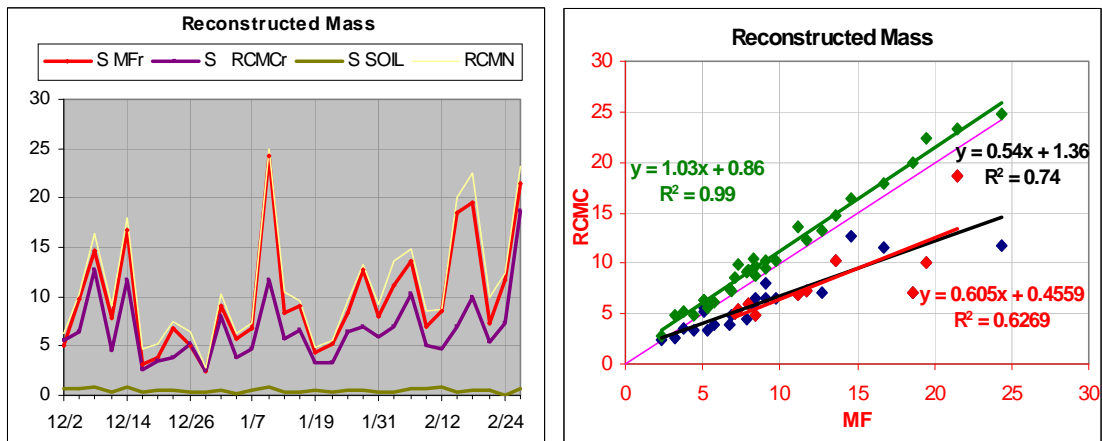


Figure 9. Time series and scatter plot of reconstructed mass (RCMC), reconstructed mass with nitrate (RCMN), and measured fine mass (MF) for a single site. In this case nitrate is a significant contributor to the fine mass.

6.6 TROUBLESHOOTING AND FINAL REVIEW

Starting with October 2004 data, the final data set is subjected to another screening before it is submitted to CIRA by selecting “Final Check” from the SWAP menu. Final Check asks for the month and year to check, then displays all records with MF>MT, all records with RF, CG, or CL in the status flag field, all records with QD in the status field, and all records with either SA or QA in the status field. It also shows all sites with less than 10 or more than 11 records for the month, and displays all records in the HOLDS file.

All records with MF>MT include a measure of the difference between the two mass measurements relative to the uncertainty in the measurement. If the relative difference is less than one, it is within the uncertainty of the measurement. All records with relative difference greater than one are reexamined for possible errors.

The records with status flags of RF, CG, or CL are displayed for information only. No action is called for unless the data validation manager determines there is a pattern or an abnormally large number of these cases.

There should be no records with QD in the status field. If such cases remain in the data set, the reason is found and the status flag is changed. Depending on the reason, the status flag could be set to normal (NM) or to any one of several terminal flags (PO, EP, UN, XX).

If any records contain SA or QA in the status field, or any sites are in the HOLDS file, an explanation for them must accompany the data transmission to CIRA. The SA status flag is set when there is a known sampling anomaly that does not affect the quality of the measurement, but also does not allow the NM flag to be set. An example would be if the sampler ran for 24 hours, but the start and end times were several hours late. The QA status flag can be set for any single module to hold back the data from that module while allowing the remaining data to be transmitted. The SWAP program also allows the data validation manager to hold all data for any site. An entire site can be held back by placing an entry in the HOLDS file.

7. SUBMISSION TO CIRA

After the Final Check option in SWAP is run and the data are deemed acceptable, the data are ready to submit to CIRA. The data file is located in the U:\IMPROVE\TotalDBF folder on the Crocker Nuclear Laboratory computer. The naming convention is QTRtotMO.dbf, where QTR is the three letter code for the sampling quarter (e.g., A05 for the spring season of 2005) and MO is the two digit month (e.g., 03 for March). The data file is sent to the following people as an email attachment:

"Schichtel, Bret" Schichtel@CIRA.colostate.edu

"Debbie Miller" debbie_c_miller@nps.gov

"Copeland, Scott" copeland@cira.colostate.edu

"McClure, Shawn" McClure@cira.colostate.edu

If there are SA flags or QA flags in the data, an explanation must be included in the email message. Any sites held back for further review must be noted in the email transmission.

8. REFERENCES

CNL (2002) IMPROVE Quality Assurance Project Plan (QAPP), Revision 0.0, Crocker Nuclear Laboratory, University of California, Davis, CA.

John, W. and Reischl, G.P. (1980) A Cyclone for Size-Selective Sampling of Ambient Air, J. Air Pollut. Control Assoc., 30 (8), 872-876.

APPENDIX A. IMPROVE DATA DICTIONARY

Logs file (C04logs.dbf)

| # | Parameter | Definition | Description | Flow db |
|----|-----------|--|---|---------|
| 1 | Site | 5-character site name | | Yes |
| 2 | Week | Tuesday date of sampling week | | |
| 3 | Day | Corresponds to the solenoid valve number | | Yes |
| 4 | Samdat | Sample date | | Yes |
| 5 | Strtim | Start time for sample, currently hardwired to 00:00 | | |
| 6 | Tempcur | Logsheet temperature in degrees Celsius | | Yes |
| 7 | Agi | A logsheet maximum orifice pressure transducer reading | An indication of the maximum vacuum pulled by the pump; low values warn of pump failure. These values are collected each week, and same values are filled in for each sample in week. | |
| 8 | Bgi | B logsheet maximum orifice pressure transducer reading | | |
| 9 | Cgi | C logsheet maximum orifice pressure transducer reading | | |
| 10 | Dgi | D logsheet maximum orifice pressure transducer reading | | |
| 11 | Af1gi | A logsheet initial orifice pressure transducer reading | The initial readings from the pressure transducer located in front of the critical orifice . They are collected by the site operator weekly when cartridges are loaded . | Yes |
| 12 | Bf1gi | B logsheet initial orifice pressure transducer reading | | Yes |
| 13 | Cf1gi | C logsheet initial orifice pressure transducer reading | | Yes |
| 14 | Df1gi | D logsheet initial orifice pressure transducer reading | | Yes |
| 15 | Af1mi | A logsheet initial cyclone pressure transducer reading | The initial readings from the pressure transducer located at the cyclone . They are collected by the site operator weekly when cartridges are loaded . | Yes |
| 16 | Bf1mi | B logsheet initial cyclone pressure transducer reading | | Yes |
| 17 | Cf1mi | C logsheet initial cyclone pressure transducer reading | | Yes |
| 18 | Df1mi | D logsheet initial cyclone pressure transducer reading | | Yes |
| 19 | Af1gf | A logsheet final orifice pressure transducer reading | The final readings from the pressure transducer located in front of the critical orifice. They are collected by the site operator weekly when cartridges are unloaded . | Yes |
| 20 | Bf1gf | B logsheet final orifice pressure transducer reading | | Yes |
| 21 | Cf1gf | C logsheet final orifice pressure transducer reading | | Yes |
| 22 | Df1gf | D logsheet final orifice pressure transducer reading | | Yes |

| # | Parameter | Definition | Description | Flow db |
|----|-----------------|---|---|---------|
| 23 | Af1mf | A logsheet final cyclone pressure transducer reading | The initial readings from the pressure transducer located at the cyclone . They are collected by the site operator weekly when cartridges are unloaded . | Yes |
| 24 | Bf1mf | B logsheet final cyclone pressure transducer reading | | Yes |
| 25 | Cf1mf | C logsheet final cyclone pressure transducer reading | | Yes |
| 26 | Df1mf | D logsheet final cyclone pressure transducer reading | | Yes |
| 27 | Af1et | A logsheet elapsed time | The elapsed time values recorded by the site operator weekly when the sample cartridges are unloaded. | Yes |
| 28 | Bf1et | B logsheet elapsed time | | Yes |
| 29 | Cf1et | C logsheet elapsed time | | Yes |
| 30 | Df1et | D logsheet elapsed time | | Yes |
| 31 | Af1nm | A validation flag | Status flags for the samples from each module | Yes |
| 32 | Bf1nm | B validation flag | | Yes |
| 33 | Cf1nm | C validation flag | | Yes |
| 34 | Df1nm | D validation flag | | Yes |
| 35 | Name | Initials of person that checked in the box | | |
| 36 | Indate | Date the sample returned to the lab | | |
| 37 | Intime | Time the sample returned to the lab | | |
| 38 | Dbl | Initials of person that double checked the box | | |
| 39 | Area | A module filter area | A module area rarely changes | |
| 40 | Protocol | Sample collection timing (1 day in 3 or Wed/Sat) | All sites are now 1 in 3 except MALO | |
| 41 | Pstdone | Logic: Post-weight measurement collected | | |
| 42 | Predone | Logic: Pre-weight measurement collected | | |
| 43 | Position | Simply a count of the number of samples at each site in the quarter | Indicates position in the analysis tray | |
| 44 | Qtr | Seasonal quarter | | |
| 45 | SO2/ SO2site | Old logs logical/new logs character field, all blanks | | |
| 46 | Dblc | Logic: carbon after-filter collected | Same sites are always used | |
| 47 | Leakcked | Logic: cartridges leak checked | | |
| 48 | Ppname | Pre-weight technician | This information is copied from the "weights" files. | |
| 49 | Predate | Pre-weight date | | |
| 50 | Pptime | Pre-weight time | | |
| 51 | Ppname | Post-weight technician | | |
| 52 | Fbchan | Module that had field blank | | |
| 53 | Fbinhouse | Logic: Field blank cassette in-house | Used to control flow of field blanks sites | |

| # | Parameter | Definition | Description | Flow db |
|----|------------|---|--|---------|
| 54 | Comments | Operator logsheet comments | | |
| 55 | Flag | Message to balance technician | Appears when filters are weighed | |
| 56 | Balpstdone | Logical: post-weight measured | | |
| 57 | Sampler | Sampler version number | | Yes |
| 58 | So2site | Site name for SO ₂ data (e.g.,INGAS) | | |
| 59 | Bdiam | B module filter diameter | Hasn't changed since 2000 | |
| 60 | Lkchkinit | Initials of person that performed leak check | | |
| 61 | Flshcode | Flashcard serial number | New compact flashcards don't have serial numbers | |
| 62 | Opinit | Site operator initials | Taken from logsheet | Yes |
| 63 | Usela | A flow measurement indicator* | | Yes |
| 64 | Uselb | B flow measurement indicator* | | Yes |
| 65 | Uselc | C flow measurement indicator* | | Yes |
| 66 | Useld | D flow measurement indicator* | | Yes |
| 67 | Loadate | Date sample cartridge loaded into sampler | | |
| 68 | Unldate | Date sample cartridge unloaded from sampler | | |
| 69 | Loadtim | Time sample cartridge loaded into sampler | | |
| 70 | Unldtim | Time sample cartridge unloaded from sampler | | |
| 71 | Uselt | temp measurement indicator | | Yes |

* These parameters indicate which flowrate measurement to use: T = use “mag” values recorded on logsheet, F = use flashcard “mag” data, V = use flashcard “vac” data, U = use “vac” values recorded on logsheet, or N = use nominal values in calibration file.

Ions files (d04qtrions.dbf)

| # | Parameter | Description |
|----|-----------------|---|
| 1 | Site | Sampler name |
| 2 | Samdat | Sample date |
| 3 | Strtim | Sample start time |
| 4 | Status | Either NM-normal, RP-replicate analysis, FB-field blank, FR-replicate field blank. NM records are routine filters. FB records are used to calculate artifact corrections. |
| 5 | Cl | Chloride ion filter density in ug/filter |
| 6 | NO ₂ | Nitrite ion filter density in ug/filter |
| 7 | NO ₃ | Nitrate ion filter density in ug/filter |
| 8 | SO ₄ | Sulfate ion filter density in ug/filter |
| 9 | NH ₄ | Ammonium ion filter density in ug/filter |
| 10 | Comments | RTI lab comments about filter analysis |

Laser files (B04laser.dbf)

| # | Parameter | Description |
|----|-----------|---|
| 1 | Site | Sampler name |
| 2 | Samdat | Sample date |
| 3 | Strtim | Sample start time |
| 4 | Status | Contains the exact validation flag in the logs file for the A module. |
| 5 | Chan | Indicates the filter module analyzed (values are often not filled in) |
| 6 | T1 | Transmitted intensity measured by the integrating plate |
| 7 | R1 | Reflected intensity measured by the integrating sphere |
| 8 | Inname1 | Name of technician doing the measurement (not used currently) |
| 9 | Indate1 | Date of measurement (not used currently) |
| 10 | Intime1 | Time of measurement (not used currently) |
| 11 | T2 | Repeat of transmitted intensity measured by the integrating plate |
| 12 | R2 | Repeat of reflected intensity measured by the integrating sphere |
| 13 | Inname2 | Name of technician doing the repeat measurement (not used currently) |
| 14 | Indate2 | Date of repeat measurement (not used currently) |
| 15 | Intime2 | Time of repeat measurement (not used currently) |

Weights files (D04wts.dbf, note older files may contain other fields)

| # | Parameter | Description |
|----|-----------|--|
| 1 | Site | Sampler name |
| 2 | Samdat | Sample date |
| 3 | Strtim | Sample start time |
| 4 | QTR | Seasonal quarter |
| 5 | Chan | Indicates the filter module analyzed (e.g., A1 or D1) |
| 6 | Status | Either NM-normal, FB-field blank, XX-destroyed filter, NS-no sample, SW-swapped filter. Data are exported with this flag if the logs flag is non-terminal. |
| 7 | Prebalwt | Pre-sampling weight as recorded by the balance |
| 8 | Pstbalwt | Post-sampling weight as recorded by the balance |
| 9 | Prindate | Date when pre-sample weight was collected |
| 10 | Printime | Time when pre-sample weight was collected |
| 11 | Prbalance | Balance that was used to weigh the filter pre-sampling |
| 12 | Prname | Person that weighed the filter pre-sampling |
| 13 | Poindate | Date when post-sample weight was collected |
| 14 | Pointime | Time when post-sample weight was collected |
| 15 | Pobalance | Balance used to weigh the filter post-sampling |
| 16 | Poname | Person that weighed the filter post-sampling |
| 17 | RH | Relative humidity in the weigh room (recorded post-sampling) |
| 18 | Temp | Temperature in the weigh room (recorded post-sampling) |
| 19 | Avrh | Not currently used |

Carbon files (D04car12.dbf)

| # | Parameter | Description |
|----|-----------|---|
| 1 | QID | Filter index, assigned by DRI |
| 2 | SITE | Sampler name |
| 3 | SAMDAT | Sample date |
| 4 | STRTIM | Sample start time |
| 5 | STATUS | Either CP-primary filter; FB-primary filter, field blank; PR-primary filter, replicate analysis; FR-primary filter, field blank, replicate analysis; CS-secondary filter; FS-secondary filter, field blank; SR-secondary filter, replicate analysis; or BR-secondary filter, field blank, replicate analysis. CP records are routine filters and CS records are used to calculate the artifact corrections. |
| 6 | OETF | DRI chemical analysis validation flags |
| 7 | OCTC | Total organic carbon filter loading in ug/filter |
| 8 | OCTU | Total organic carbon filter loading uncertainty in ug/filter |
| 9 | ECTC | Total elemental carbon filter loading in ug/filter |
| 10 | ECTU | Total elemental carbon filter loading uncertainty in ug/filter |
| 11 | EHTC | Elemental carbon evolved at temperature > 550°C filter loading in ug/filter |
| 12 | EHTU | Elemental carbon evolved at temperature > 550°C filter loading uncertainty in ug/filter |
| 13 | TCTC | Total carbon filter loading in ug/filter |
| 14 | TCTU | Total carbon filter loading uncertainty in ug/filter |
| 15 | O1TC | Organic carbon evolved in He atmosphere at 120°C filter loading in ug/filter |
| 16 | O1TU | Organic carbon evolved in He atmosphere at 120°C filter loading uncertainty in ug/filter |
| 17 | O2TC | Organic carbon evolved in He atmosphere at 250°C filter loading in ug/filter |
| 18 | O2TU | Organic carbon evolved in He atmosphere at 250°C filter loading uncertainty in ug/filter |
| 19 | O3TC | Organic carbon evolved in He atmosphere at 450°C filter loading in ug/filter |
| 20 | O3TU | Organic carbon evolved in He atmosphere at 450°C filter loading uncertainty in ug/filter |
| 21 | O4TC | Organic carbon evolved in He atmosphere at 550°C filter loading in ug/filter |
| 22 | O4TU | Organic carbon evolved in He atmosphere at 550°C filter loading uncertainty in ug/filter |
| 23 | OPTC | Pryrolyzed organic carbon filter loading in ug/filter |
| 24 | OPTU | Pryrolyzed organic carbon filter loading uncertainty in ug/filter |
| 25 | E1TC | Elemental carbon evolved in He/O ₂ atmosphere at 550°C filter loading in ug/filter |
| 26 | E1TU | Elemental carbon evolved in He/O ₂ atmosphere at 550°C filter loading uncertainty in ug/filter |
| 27 | E2TC | Elemental carbon evolved in He/O ₂ atmosphere at 700°C filter loading in ug/filter |
| 28 | E2TU | Elemental carbon evolved in He/O ₂ atmosphere at 700°C filter loading uncertainty in ug/filter |
| 29 | E3TC | Elemental carbon evolved in He/O ₂ atmosphere at 800°C filter loading in ug/filter |
| 30 | E3TU | Elemental carbon evolved in He/O ₂ atmosphere at 800°C filter loading uncertainty in ug/filter |
| 31 | Comment | DRI lab comments about filter analysis |

XRF files (D04ele12.dbf, note older files also contain PIXE parameters)

| # | Parameter | Description |
|-------|-----------------------|---|
| 1 | Site | Sampler name |
| 2 | Samdat | Sample date |
| 3 | Strtim | Sample start time |
| 4 | Pesstat | PESA status flag, all values are NM |
| 5 | Xrmstat | Molybdenum-anode XRF status flag, all values are NM |
| 6 | Xrcstat | Copper-anode XRF status flag, all values are NM |
| 7-9 | H, H_err, H_mdl | Hydrogen filter loading, uncertainty, and minimum detection limit from PESA in ng/cm ² |
| 10-12 | S, S_err, S_mdl | Sulfur filter loading, uncertainty, and minimum detection limit from molybdenum-anode XRF in ng/cm ² (not reported to CIRA) |
| 13-15 | Cl, Cl_err, Cl_mdl | Chlorine filter loading, uncertainty, and minimum detection limit from molybdenum-anode XRF in ng/cm ² (not reported to CIRA) |
| 16-18 | K, K_err, K_mdl | Potassium filter loading, uncertainty, and minimum detection limit from molybdenum-anode XRF in ng/cm ² (not reported to CIRA) |
| 19-21 | Ca, Ca_err, Ca_mdl | Calcium filter loading, uncertainty, and minimum detection limit from molybdenum-anode XRF in ng/cm ² (not reported to CIRA) |
| 22-24 | Ba, Ba_err, Ba_mdl | Barium filter loading, uncertainty, and minimum detection limit from molybdenum-anode XRF in ng/cm ² (not reported to CIRA) |
| 25-27 | Ti, Ti_err, Ti_mdl | Titanium filter loading, uncertainty, and minimum detection limit from molybdenum-anode XRF in ng/cm ² (not reported to CIRA) |
| 28-30 | V, V_err, V_mdl | Vanadium filter loading, uncertainty, and minimum detection limit from molybdenum-anode XRF in ng/cm ² (not reported to CIRA) |
| 31-33 | Cr, Cr_err, Cr_mdl | Chromium filter loading, uncertainty, and minimum detection limit from molybdenum-anode XRF in ng/cm ² (not reported to CIRA) |
| 34-36 | Mn, Mn_err, Mn_mdl | Manganese filter loading, uncertainty, and minimum detection limit from molybdenum-anode XRF in ng/cm ² (not reported to CIRA) |
| 37-39 | Fe, Fe_err, Fe_mdl | Iron filter loading, uncertainty, and minimum detection limit from molybdenum-anode XRF in ng/cm ² (not reported to CIRA) |
| 40-42 | Co, Co_err, Co_mdl | Cobalt filter loading, uncertainty, and minimum detection limit from molybdenum-anode XRF in ng/cm ² (not reported to CIRA) |
| 43-45 | Ni, Ni_err, Ni_mdl | Nickel filter loading, uncertainty, and minimum detection limit from molybdenum-anode XRF in ng/cm ² |
| 46-48 | Cu, Cu_err, Cu_mdl | Copper filter loading, uncertainty, and minimum detection limit from molybdenum-anode XRF in ng/cm ² |
| 49-51 | Zn, Zn_err, Zn_mdl | Zinc filter loading, uncertainty, and minimum detection limit from molybdenum-anode XRF in ng/cm ² |
| 52-54 | Ga, Ga_err, Ga_mdl | Galium filter loading, uncertainty, and minimum detection limit from molybdenum-anode XRF in ng/cm ² (not reported to CIRA) |
| 55-57 | Au, Au_err, Au_mdl | Gold filter loading, uncertainty, and minimum detection limit from molybdenum-anode XRF in ng/cm ² (not reported to CIRA) |
| 58-60 | Hg, Hg_err, Hg_mdl | Mercury filter loading, uncertainty, and minimum detection limit from molybdenum-anode XRF in ng/cm ² (not reported to CIRA) |

| | | |
|---------|--------------------------|---|
| 61-63 | As, As_err, As_mdl | Arsenic filter loading, uncertainty, and minimum detection limit from molybdenum-anode XRF in ng/cm ² |
| 64-66 | Pb, Pb_err, Pb_mdl | Lead filter loading, uncertainty, and minimum detection limit from molybdenum-anode XRF in ng/cm ² |
| 67-69 | Se, Se_err, Se_mdl | Selenium filter loading, uncertainty, and minimum detection limit from molybdenum-anode XRF in ng/cm ² |
| 70-72 | Br, Br_err, Br_mdl | Bromine filter loading, uncertainty, and minimum detection limit from molybdenum-anode XRF in ng/cm ² |
| 73-75 | Rb, Rb_err, Rb_mdl | Rubidium filter loading, uncertainty, and minimum detection limit from molybdenum-anode XRF in ng/cm ² |
| 76-78 | Sr, Sr_err, Sr_mdl | Mercury filter loading, uncertainty, and minimum detection limit from molybdenum-anode XRF in ng/cm ² |
| 79-81 | Y, Y_err, Y_mdl | Yttrium filter loading, uncertainty, and minimum detection limit from molybdenum-anode XRF in ng/cm ² (not reported to CIRA) |
| 82-84 | Zr, Zr_err, Zr_mdl | Zirconium filter loading, uncertainty, and minimum detection limit from molybdenum-anode XRF in ng/cm ² |
| 85-87 | Cd, Cd_err, Cd_mdl | Cadmium filter loading, uncertainty, and minimum detection limit from molybdenum-anode XRF in ng/cm ² (not reported to CIRA) |
| 88-90 | Ag, Ag_err, Ag_mdl | Silver filter loading, uncertainty, and minimum detection limit from molybdenum-anode XRF in ng/cm ² (not reported to CIRA) |
| 91-93 | Yna, Yna_err, Yna_mdl | Sodium filter loading, uncertainty, and minimum detection limit from copper-anode XRF in ng/cm ² |
| 94-96 | Ymg, Ymg_err, Ymg_mdl | Magnesium filter loading, uncertainty, and minimum detection limit from copper-anode XRF in ng/cm ² |
| 97-99 | Yal, Yal_err, Yal_mdl | Aluminum filter loading, uncertainty, and minimum detection limit from copper-anode XRF in ng/cm ² |
| 100-102 | Ysi, Ysi_err, Ysi_mdl | Silicon filter loading, uncertainty, and minimum detection limit from copper-anode XRF in ng/cm ² |
| 103-105 | Yp, Yp_err, Yp_mdl | Phosphorous filter loading, uncertainty, and minimum detection limit from copper-anode XRF in ng/cm ² |
| 106-108 | Ys, Ys_err, Ys_mdl | Sulfur filter loading, uncertainty, and minimum detection limit from copper-anode XRF in ng/cm ² |
| 109-111 | Ycl, Ycl_err, Ycl_mdl | Chlorine filter loading, uncertainty, and minimum detection limit from copper-anode XRF in ng/cm ² |
| 112-114 | Yk, Yk_err, Yk_mdl | Potassium filter loading, uncertainty, and minimum detection limit from copper-anode XRF in ng/cm ² |
| 115-117 | Yca, Yca_err, Yca_mdl | Calcium filter loading, uncertainty, and minimum detection limit from copper-anode XRF in ng/cm ² |
| 118-120 | Yba, Yba_err, Yba_mdl | Barium filter loading, uncertainty, and minimum detection limit from copper-anode XRF in ng/cm ² (not reported to CIRA) |
| 121-123 | Yti, Yti_err, Yti_mdl | Yttrium filter loading, uncertainty, and minimum detection limit from copper-anode XRF in ng/cm ² |
| 124-126 | Yv, Yv_err, Yv_mdl | Vanadium filter loading, uncertainty, and minimum detection limit from copper-anode XRF in ng/cm ² |
| 127-129 | Ycr, Ycr_err, Ycr_mdl | Chromium filter loading, uncertainty, and minimum detection limit from copper-anode XRF in ng/cm ² |

| | | |
|---------|--------------------------|--|
| 130-132 | Ymn, Ymn_err, Ymn_mdl | Manganese filter loading, uncertainty, and minimum detection limit from copper-anode XRF in ng/cm ² |
| 133-135 | Yfe, Yfe_err, Yfe_mdl | Iron filter loading, uncertainty, and minimum detection limit from copper-anode XRF in ng/cm ² |
| 136-138 | Yco, Yco_err, Yco_mdl | Cobalt filter loading, uncertainty, and minimum detection limit from copper - anode XRF in ng/cm ² (not reported to CIRA) |
| 139-141 | Yni, Yni_err, Yni_mdl | Nickel filter loading, uncertainty, and minimum detection limit from copper - anode XRF in ng/cm ² (not reported to CIRA) |
| 142 | Pesalt | PESA sample-specific detector live time (not reported to CIRA) |
| 143 | Pixelt | NOTE: This parameter should have been removed in D04 but wasn't so the columns are shifted incorrectly (not reported to CIRA) |
| 144 | Xrmlt | Molybdenum-anode XRF sample-specific detector live time (not reported to CIRA) |
| 145 | Xrcit | Copper-anode XRF sample-specific detector live time (not reported to CIRA) |

Arthist file (last modified 10/28/05) The values stored in this file are used to correct the ions and carbon data for sampling artifacts. The file contains records that summarize the artifacts for 3-month seasonal quarters and records that summarize the artifacts for individual months.

| # | Parameter | Description |
|----|-----------|--|
| 1 | Qtr | Seasonal quarter |
| 2 | Month | Integer month value |
| 3 | Cl | Median chloride ion field blank loading (ug/filter) |
| 4 | Dcl | Standard deviation of the chloride field blank loadings (ug/filter) |
| 5 | NO2 | Median nitrite ion field blank loading (ug/filter) |
| 6 | DNO2 | Standard deviation of the nitrite field blank loadings (ug/filter) |
| 7 | NO3 | Median nitrate ion median field blank loading (ug/filter) |
| 8 | DNO3 | Standard deviation of the nitrate field blank loadings (ug/filter) |
| 9 | SO4 | Median sulfate ion median field blank loading (ug/filter) |
| 10 | DSO4 | Standard deviation of the sulfate field blank loadings (ug/filter) |
| 11 | NH4 | Median ammonium ion field blank loading (ug/filter) |
| 12 | DNH4 | Standard deviation of the ammonium field blank loadings (ug/filter) |
| 13 | O1 | Median of the first OC fraction loadings on the secondary filters (ug/filter) |
| 14 | DO1 | Standard deviation of the first OC fraction loadings on the secondary filters (ug/filter) |
| 15 | O2 | Median of the second OC fraction loadings on the secondary filters (ug/filter) |
| 16 | DO2 | Standard deviation of the second OC fraction loadings on the secondary filters (ug/filter) |
| 17 | O3 | Median of the third OC fraction loadings on the secondary filters (ug/filter) |
| 18 | DO3 | Standard deviation of the third OC fraction loadings on the secondary filters (ug/filter) |
| 19 | O4 | Median of the fourth OC fraction loadings on the secondary filters (ug/filter) |
| 20 | DO4 | Standard deviation of the fourth OC fraction loadings on the secondary filters (ug/filter) |
| 21 | OP | Median of the first OC fraction loadings on the secondary filters (ug/filter) |
| 22 | DOP | Standard deviation of the first OC fraction loadings on the secondary filters (ug/filter) |
| 23 | E1 | Median of the first EC fraction loadings on the secondary filters (ug/filter) |
| 24 | DE1 | Standard deviation of the first EC fraction loadings on the secondary filters (ug/filter) |
| 25 | E2 | Median of the second EC fraction loadings on the secondary filters (ug/filter) |
| 26 | DE2 | Standard deviation of the second EC fraction loadings on the secondary filters (ug/filter) |
| 27 | E3 | Median of the third EC fraction loadings on the secondary filters (ug/filter) |
| 28 | DE3 | Standard deviation of the third EC fraction loadings on the secondary filters (ug/filter) |
| 29 | So2 | Median of the SO ₂ loadings on the field blanks (ug/filter) |
| 30 | Dso2 | Standard deviation of the SO ₂ loadings on the field blanks (ug/filter) |

Lotnums file (last modified 11/8/05)

| # | Parameter | Description |
|---|-----------|---|
| 1 | Site | Name of the sampler that was being loaded when the lot number changed |
| 2 | Samdat | Date of the sample that was being loaded when the lot number changed |
| 3 | Strtim | Start time for the sample that was being loaded when the lot number changed |
| 4 | Lotnum | Filter lot number |
| 5 | Strtdate | Date that the filter lot number changed |
| 6 | Strttime | Time that the filter lot number changed |
| 7 | Comments | |

Conterr file (last modified 7/26/01) This file contains the assumed volume and analytical errors for the ions and carbon fractions (note: there are currently two records in this file). These errors are used to calculate the uncertainty in the reported concentrations.

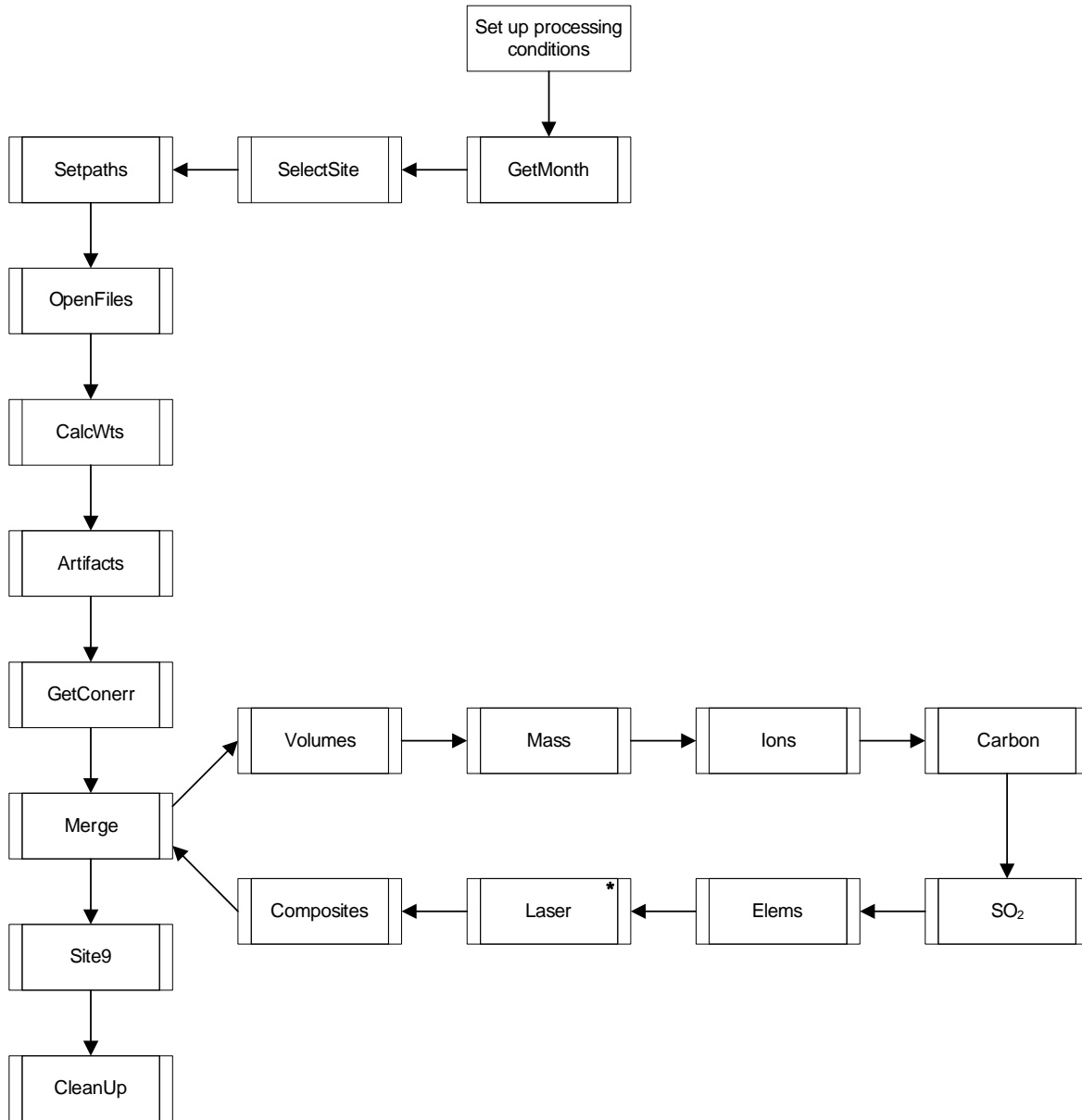
| # | Parameter | Description |
|----|-----------|---|
| 1 | DATE | First date that these errors were assumed |
| 2 | MK | Not sure what this is, it is not used in Lowell's builddb code |
| 3 | MFVOL | Assumed error (percent) in the volume measurements, applies to all module |
| 4 | O1 | Assumed analytical error (percent) in the first organic carbon fraction concentrations |
| 5 | O2 | Assumed analytical error (percent) in the second organic carbon fraction concentrations |
| 6 | O3 | Assumed analytical error (percent) in the third organic carbon fraction concentrations |
| 7 | O4 | Assumed analytical error (percent) in the fourth organic carbon fraction concentrations |
| 8 | OP | Assumed analytical error (percent) in the pyrolyzed carbon fraction concentrations |
| 9 | E1 | Assumed analytical error (percent) in the first elemental carbon fraction concentrations |
| 10 | E2 | Assumed analytical error (percent) in the second elemental carbon fraction concentrations |
| 11 | E3 | Assumed analytical error (percent) in the third elemental carbon fraction concentrations |
| 12 | CL | Assumed analytical error (percent) in chloride ion concentrations |
| 13 | NO2 | Assumed analytical error (percent) in nitrite ion concentrations |
| 14 | NO3 | Assumed analytical error (percent) in nitrate ion concentrations |
| 15 | SO4 | Assumed analytical error (percent) in sulfate ion concentrations |
| 16 | NH4 | Assumed analytical error (percent) in ammonium ion concentrations |
| 17 | SO2 | Assumed analytical error (percent) in sulfur dioxide concentrations |
| 18 | Clmdl | Chloride analytical detection limit (used if field blank median = 0) |
| 19 | NO2mdl | Nitrite analytical detection limit (used if field blank median = 0) |
| 20 | NO3mdl | Nitrate analytical detection limit (used if field blank median = 0) |
| 21 | SO4mdl | Sulfate analytical detection limit (used if field blank median = 0) |
| 22 | NH4mdl | Ammonium analytical detection limit (used if field blank median = 0) |

BalEq files: Each balance, excluding the Cahn 31 reference balance, used in the weigh room has a separate file (e.g., baleq31a.dbf). The equations are used to adjust the weight measured on a particular balance to the corresponding weight on the Cahn 31 balance.

| # | Parameter | Description |
|---|-----------|---|
| 1 | Tareint | Balance intercept for regression to the Cahn 31 balance |
| 2 | Tareslp | Balance slope for the regression to the Cahn 31 balance |
| 3 | Begdat | Starting date for the regression equation |

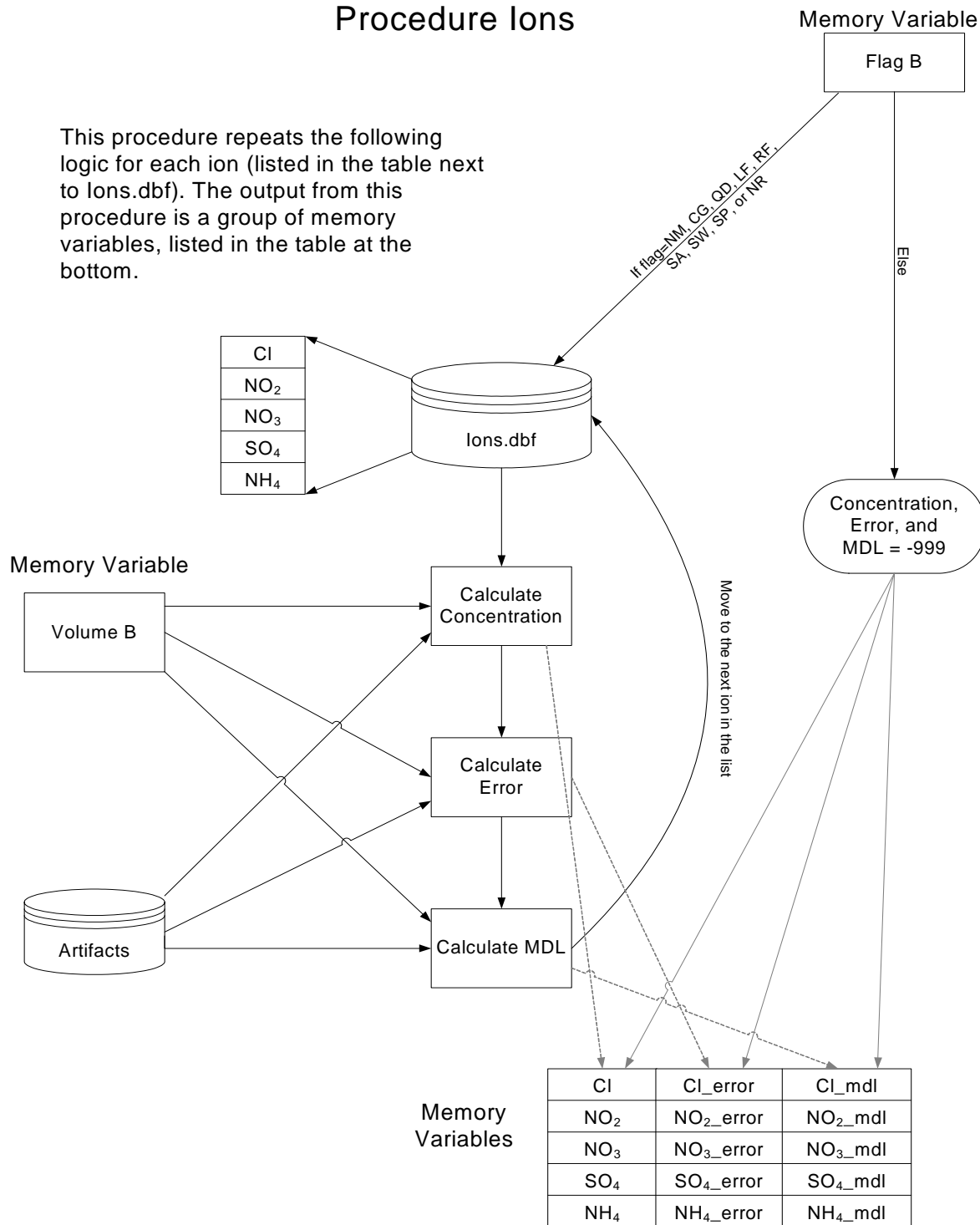
APPENDIX B. BUILDDBF CODE DIAGRAMS

BuildDBF Main Code



Procedure Ions

This procedure repeats the following logic for each ion (listed in the table next to Ions.dbf). The output from this procedure is a group of memory variables, listed in the table at the bottom.



Procedure Elems

This procedure uses a matrix to perform the calculations.

Ele.dbf contains density, error, & MDL for each of these parameters.

H is from PESA, Y preceding element name implies Cu-XRF. Otherwise element is from M0-XRF.

| | | |
|-----|-----|----|
| H | YCa | As |
| YNa | YTi | Pb |
| YMg | YV | Se |
| YAl | YCr | Br |
| YSi | YMn | Rb |
| YP | YFe | Sr |
| YS | Ni | Zr |
| YCl | Cu | |
| YK | Zn | |

Memory Variables

Volume A,
Filter area

Logs.dbf

Memory Variables
Concentration,
error, & MDL for
each parameter
in table

| | | |
|-----|-----|----|
| H | YCa | As |
| YNa | YTi | Pb |
| YMg | YV | Se |
| YAl | YCr | Br |
| YSi | YMn | Rb |
| YP | YFe | Sr |
| YS | Ni | Zr |
| YCl | Cu | |
| YK | Zn | |

Ele.dbf

Calculate
Concentration,
error, and md for
each element in
matrix

Memory Variable

Flag A

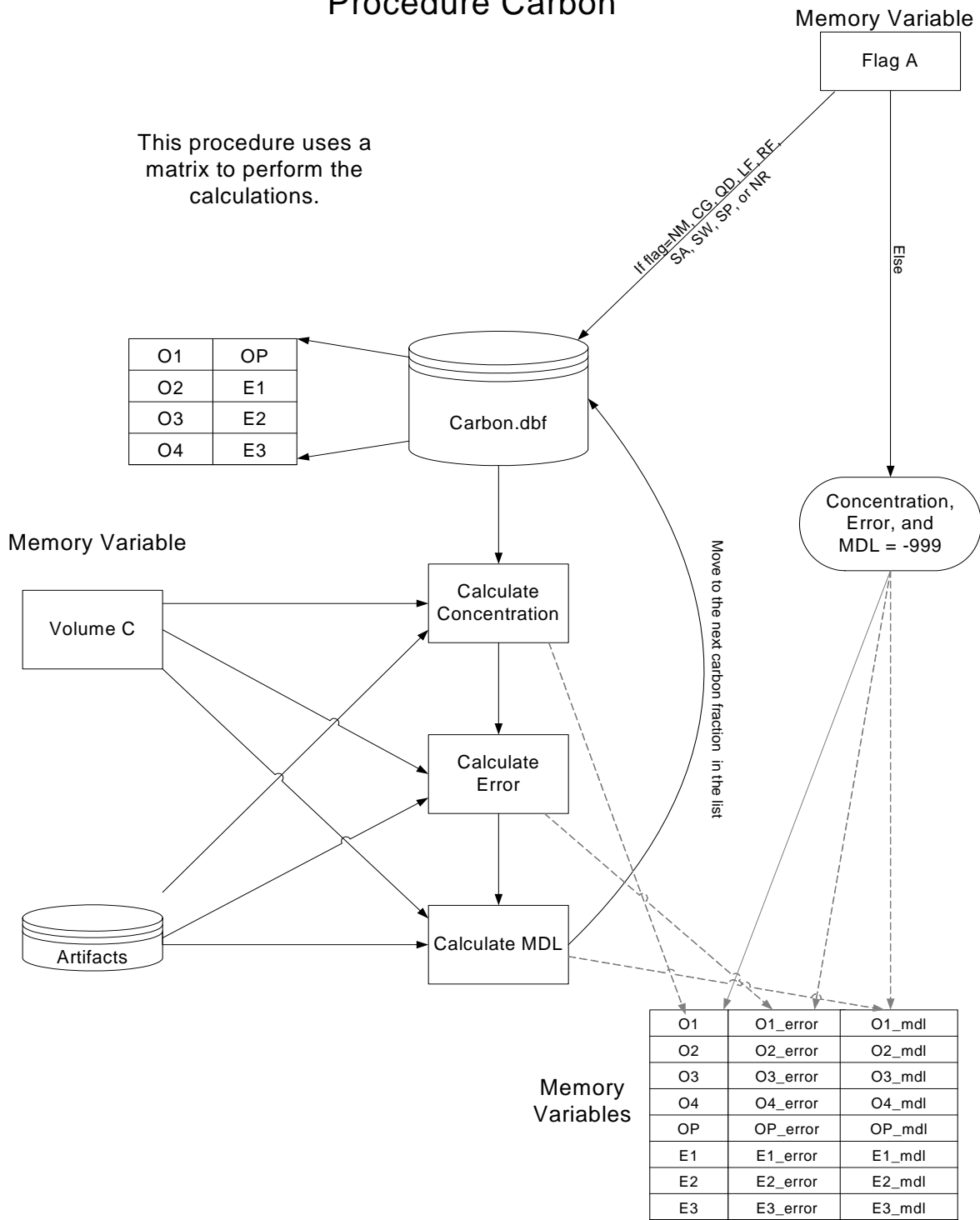
Concentration,
Error, and
MDL = -999

If flag=MM,CG,OD,LF,RF,
SA, SW, SP, or NR

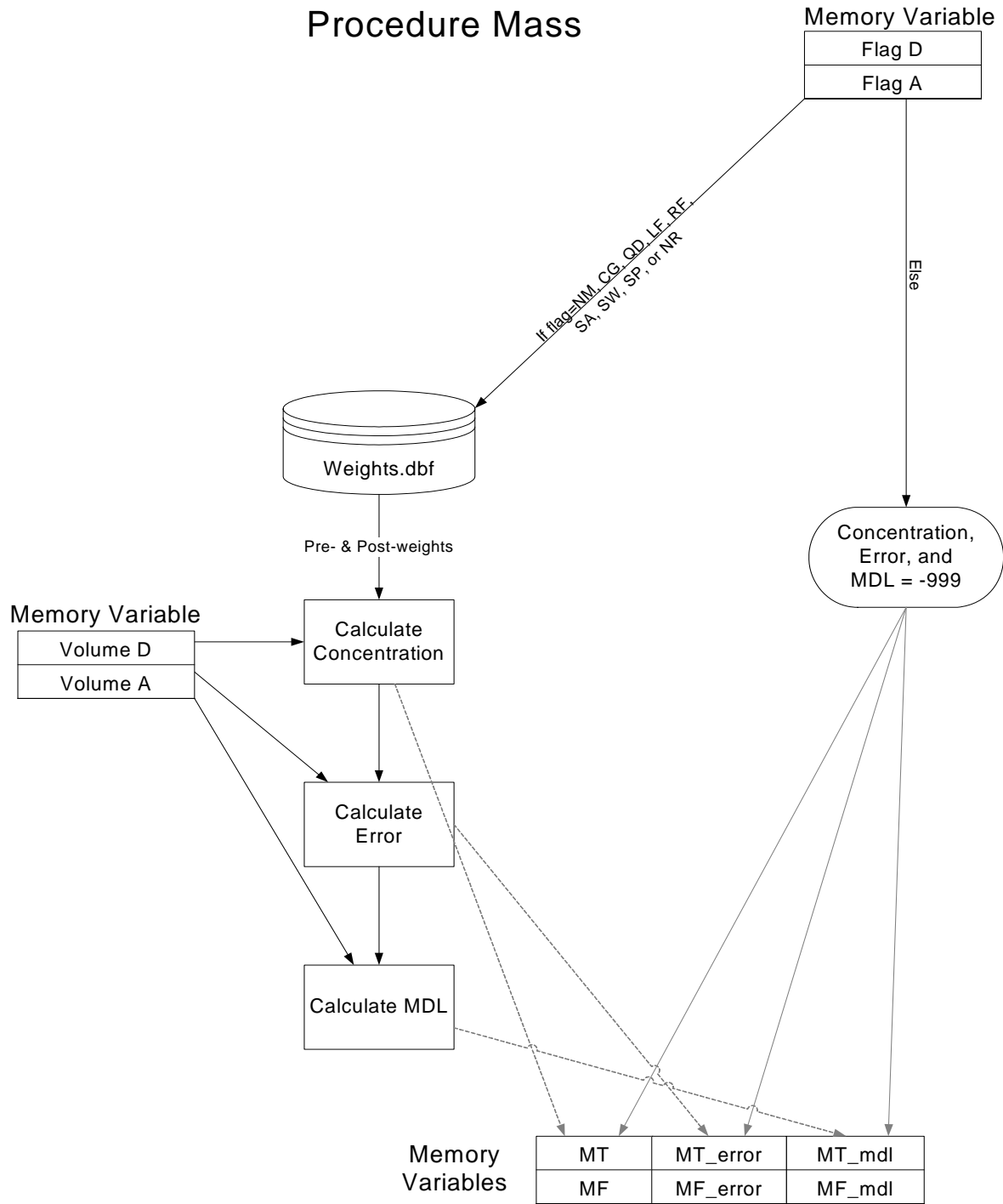
Else

Copy errors & MDLs into memory variables

Procedure Carbon

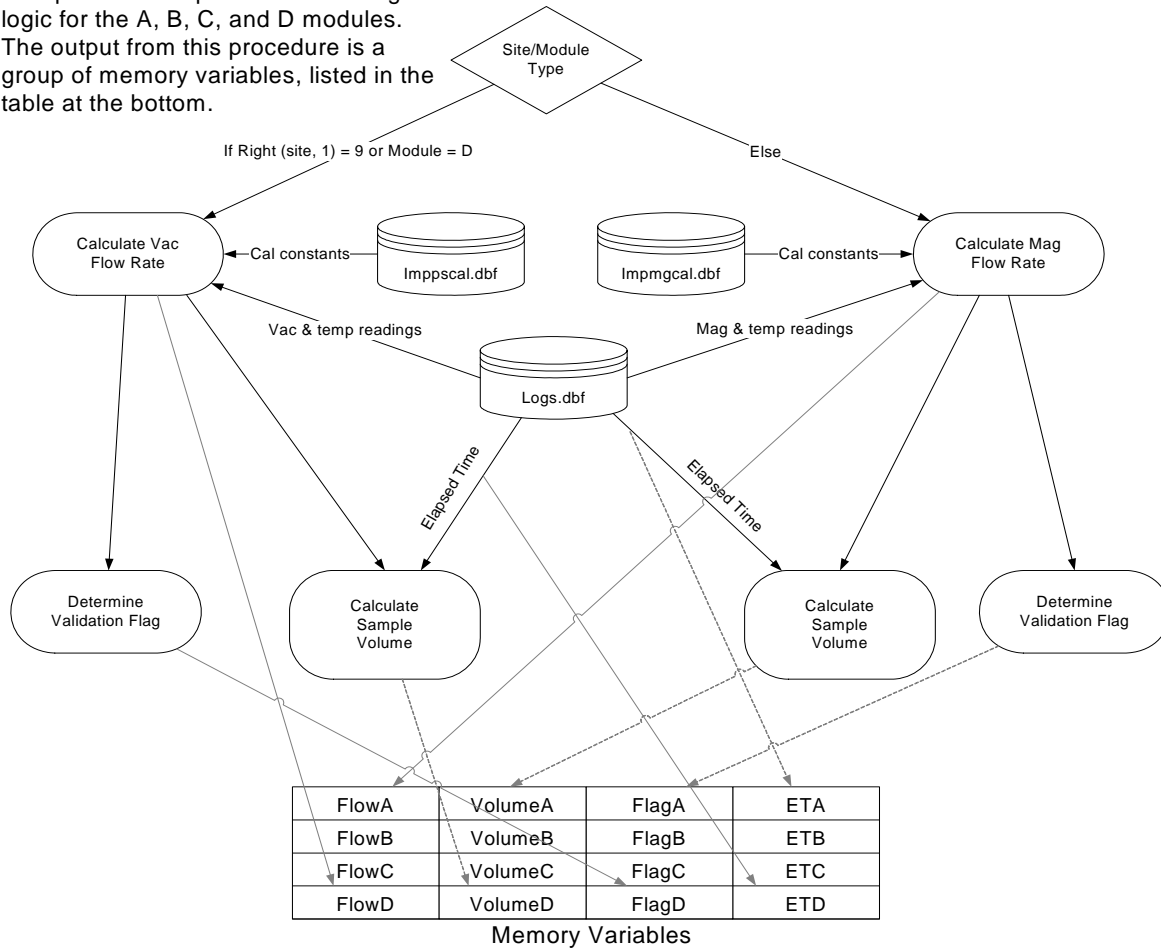


Procedure Mass

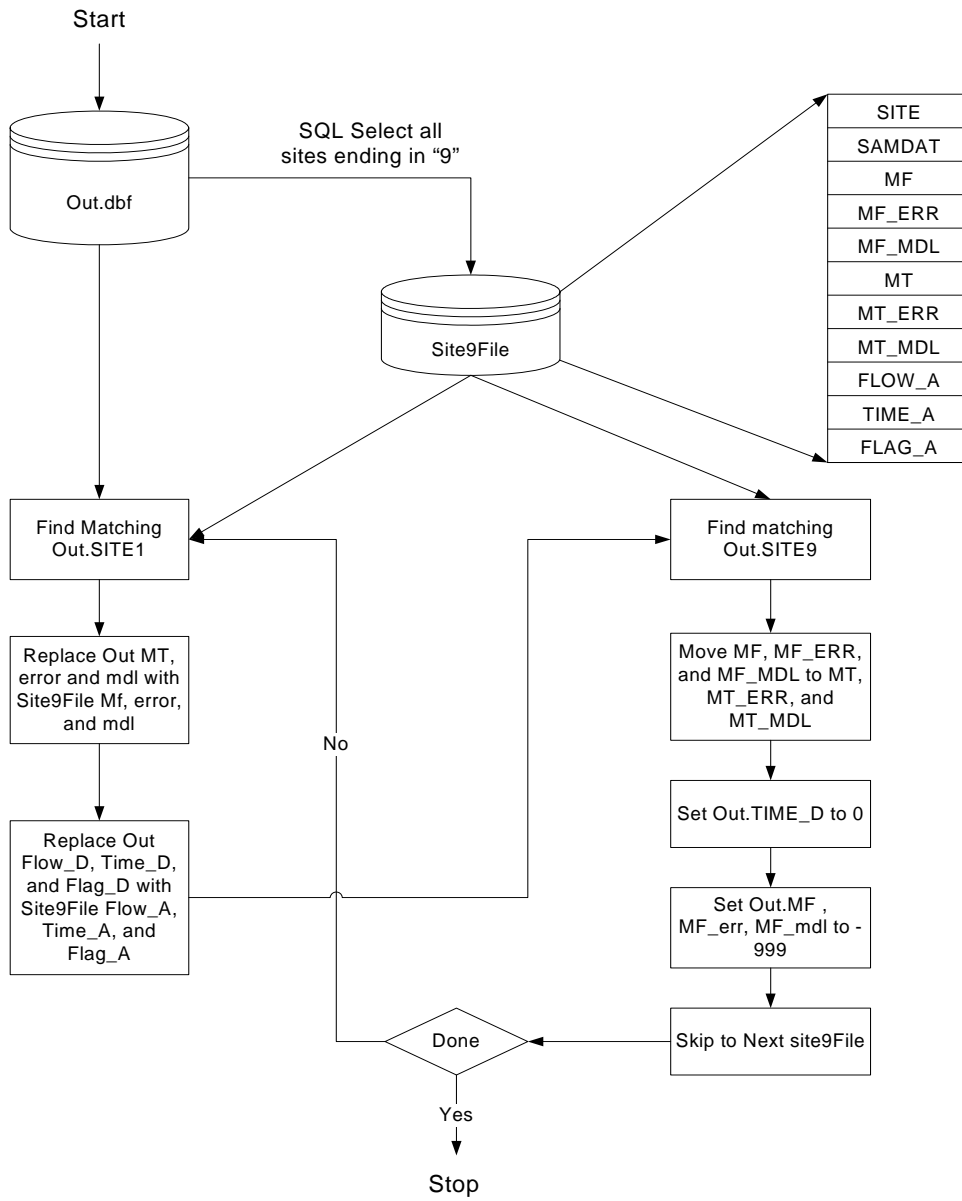


Procedure Volumes

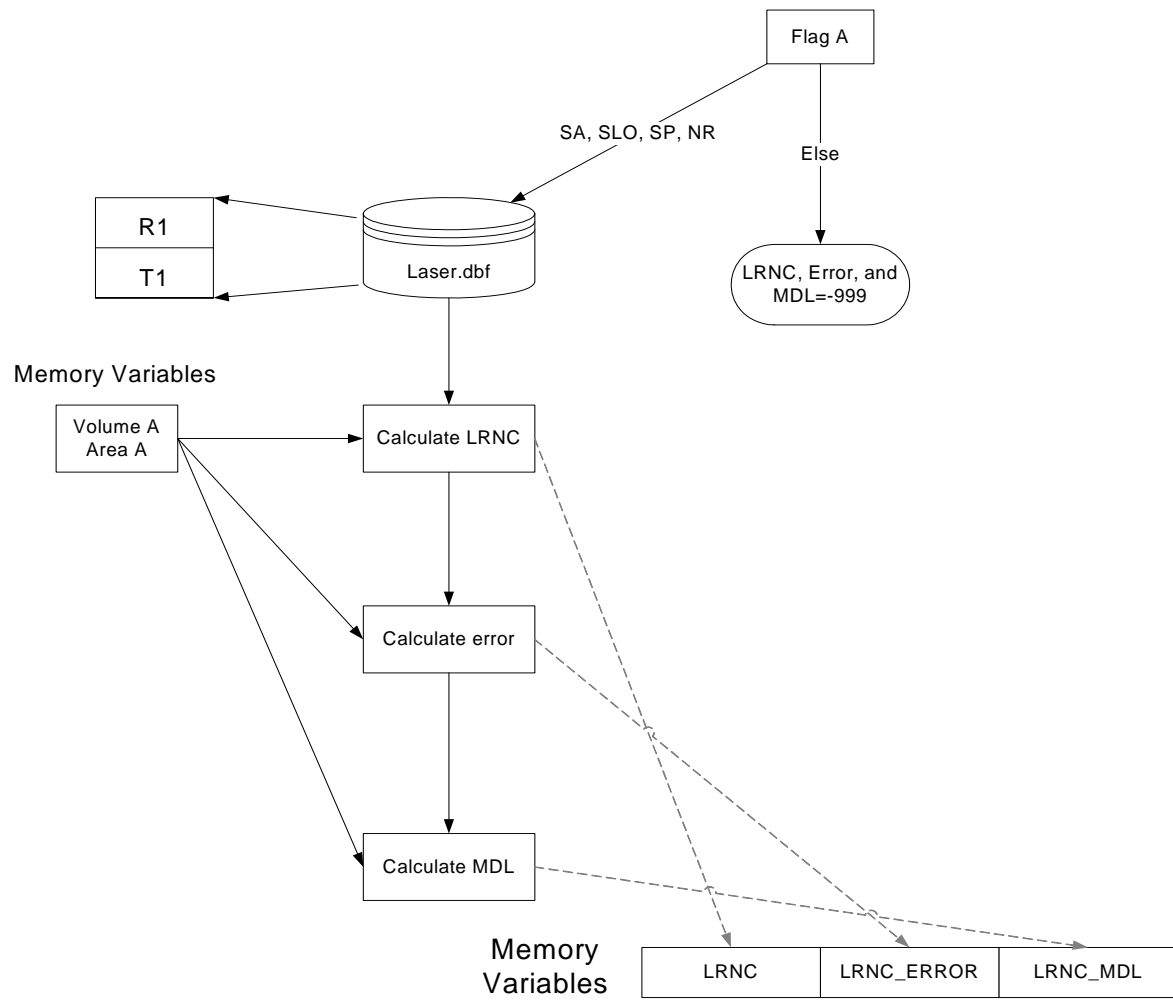
This procedure repeats the following logic for the A, B, C, and D modules. The output from this procedure is a group of memory variables, listed in the table at the bottom.



Procedure Site9



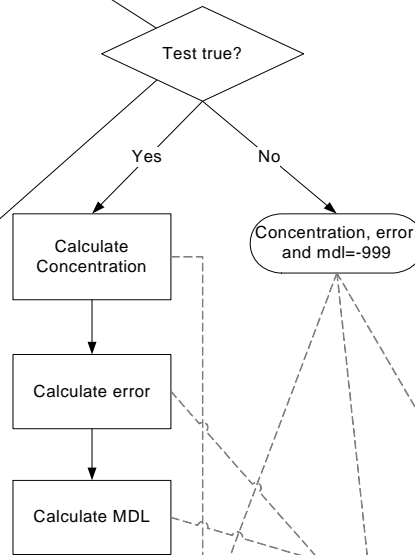
Procedure Laser



Procedure Composites

The following tests control calculation of the corresponding composite variable

| | |
|--------------------------|------|
| H>0 and S>S_mdl | OMH |
| O1>0 or O1_err>0 | OMCN |
| E1>0 or E1_err>0 | LACN |
| Si_mdl>0 and Fe_mdl>0 | SOIL |
| S>0 | NHSO |
| NO3>0 | NHNO |
| Fe>0 and K>0 | KNON |
| LRNC>0 | RCMA |
| OMC_err>0 and LACN_err>0 | RCMC |
| SO4_err>0 and LRCN_mdl>0 | RCMI |
| S>0 | S3 |
| MF>0 | MC |
| SO4>0 and NO3>0 | RCNH |
| OMH_err>0 and NO3_err>0 | OMHC |



Memory Variables

| | | |
|------|------------|----------|
| OMH | OMH_error | OMH_mdl |
| OMCN | OMCN_error | OMCN_mdl |
| LACN | LACN_error | LACN_mdl |
| SOIL | SOIL_error | SOIL_mdl |
| NHSO | NHSO_error | NHSO_mdl |
| NHNO | NHNO_error | NHNO_mdl |
| KNON | KNON_error | KNON_mdl |
| RCMA | RCMA_error | RCMA_mdl |
| RCMC | RCMC_error | RCMC_mdl |
| RCMI | RCMI_error | RCMI_mdl |
| S3 | S3_error | S3_mdl |
| MC | MC_error | MC_mdl |
| RCNH | RCNH_error | RCNH_mdl |
| OMHC | OMHC_error | OMHC_mdl |

APPENDIX C. BUILDDBF CODE

Table of Contents

| | | |
|-------|-----------------------|------|
| C.1. | BuildDBF..... | C-2 |
| C.2. | GetMonth..... | C-6 |
| C.3. | SelectSite..... | C-9 |
| C.4. | SetPaths..... | C-12 |
| C.5. | OpenFiles..... | C-14 |
| C.6. | CalcWts..... | C-18 |
| C.7. | Artifacts..... | C-22 |
| C.8. | Merge..... | C-30 |
| C.9. | Volumes..... | C-33 |
| C.10. | Mass..... | C-44 |
| C.11. | Ions..... | C-45 |
| C.12. | Carbon..... | C-49 |
| C.13. | SO ₂ | C-52 |
| C.14. | Elms..... | C-53 |
| C.15. | Laser..... | C-56 |
| C.16. | Composites..... | C-57 |
| C.17. | Site9..... | C-60 |
| C.18. | CleanUp..... | C-61 |
| C.19. | SiteXFile..... | C-63 |
| C.20. | Xmodule_Add_A..... | C-65 |
| C.21. | Xmodule_Add_B..... | C-67 |
| C.22. | Xmodule_Add_C..... | C-68 |
| C.23. | Xmodule_Add_D..... | C-69 |
| C.24. | Xcomposites..... | C-70 |
| C.25. | QlogsErr..... | C-73 |
| C.26. | Errhand..... | C-74 |

C.1. BuildDBF

```
* Program BUILDdbf
* Create the IMPROVE monthly database
*
*****
*
* Modified 11/17/2005 to add analytical mdl for Cl, NO2, NO3, SO4, and NH4
* to conterr.dbf and use them if field blanks stdev is zero.
*
* Modified 1/4/2006 by Lowell to remove code used to build 2000-2004 data
* sequentially. The code is commented out with *&& at the beginning of
* each line if there's any reason to reinstate it later.
*****

PARAMETERS mSwap
PUBLIC mseason, mmonth, mUser, mCalled
PUBLIC mdisk, mrootpath, mlogs, mlogpath
PUBLIC validstat, NMLFRF, BailOut
PUBLIC ntimes, nindex, nindex2
*&&PUBLIC bldMonth(12), bldSeason(21)
PUBLIC mdlc1, mdlno2, mdlno3, mdlso4, mdlnh4
mCalled = mSwap

validstat = " 'NM', 'CG', 'QD', 'LF', 'RF', 'SA', 'SW', 'SP', 'NR' "
NMLFRF = " 'NM', 'LF', 'RF', 'NR' "
mUSER = GETENV('USERNAME')

*&&bldMonth(1) = "01"
*&&bldMonth(2) = "02"
*&&bldMonth(3) = "03"
*&&bldMonth(4) = "04"
*&&bldMonth(5) = "05"
*&&bldMonth(6) = "06"
*&&bldMonth(7) = "07"
*&&bldMonth(8) = "08"
*&&bldMonth(9) = "09"
*&&bldMonth(10) = "10"
*&&bldMonth(11) = "11"
*&&bldMonth(12) = "12"

*&&bldSeason( 1) = "D99"
*&&bldSeason( 2) = "A00"
*&&bldSeason( 3) = "B00"
*&&bldSeason( 4) = "C00"
*&&bldSeason( 5) = "D00"
*&&bldSeason( 6) = "A01"
*&&bldSeason( 7) = "B01"
*&&bldSeason( 8) = "C01"
*&&bldSeason( 9) = "D01"
*&&bldSeason(10) = "A02"
*&&bldSeason(11) = "B02"
*&&bldSeason(12) = "C02"
*&&bldSeason(13) = "D02"
*&&bldSeason(14) = "A03"
*&&bldSeason(15) = "B03"
*&&bldSeason(16) = "C03"
*&&bldSeason(17) = "D03"
*&&bldSeason(18) = "A04"
*&&bldSeason(19) = "B04"
*&&bldSeason(20) = "C04"
*&&bldSeason(21) = "D04"

SET TALK OFF
SET SAFETY OFF

ON ERROR DO errhand WITH ;
    ERROR( ), MESSAGE( ), MESSAGE(1), PROGRAM( ), LINENO( )

CLEAR
DO WHILE .T.
```



```

*IF DIRECTORY("M:\")
*   mdisk="m:\"
   IF DIRECTORY("U:\")
       mdisk="u:\"
   ELSE
       mdisk="c:\"
   ENDIF
   mrootpath = mdisk + "improve\"
   mlogpath  = mrootpath + "logs\"
   STORE SYS(5)+SYS(2003) TO mdefdrv
   IF DIRECTORY("M:\zdefault\foxpro")
       SET DEFAULT TO M:\zdefault\FOXPRO
   ELSE
       MKDIR M:\zdefault\FOXPRO
       SET DEFAULT TO M:\zdefault\FOXPRO
   ENDIF

   SET PATH TO M:\IMPROVE\PROGRAMS

   IF NOT mCalled
       CLOSE DATABASES ALL
   ENDIF
   BailOut = .F.

**&&WAIT WINDOW AT 10,40 "Press 'A' to process 2000-2004" + CHR(13) + CHR(10) +
   "Any other key for a single month" TO mlst5
**&&IF INLIST (mlst5, "A", "a")
**&&   numMonths = 60
**&&   ProcessAll = .T.
**&&   mseason = bldSeason(1)
**&&   mmonth = bldMonth(1)
**&&   mQtr = LEFT(mseason,1)
**&&ELSE
   numMonths = 1
   ProcessAll = .F.
   DO GetMonth
**&&ENDIF

   mlogs = mlogpath + mseason + "logs.dbf"
   DO SelectSite
   blbsite = msite
   FOR nTimes = 1 TO numMonths
       CLOSE DATABASES ALL
**&&   IF ProcessAll
**&&       nIndex = INT(nTimes/3)+1
**&&       mSeason = bldSeason(nIndex)
**&&       nIndex2 = nTimes - INT((nTimes-1)/12)*&&12
**&&       mMonth = bldMonth(nIndex2)
**&&       mQtr = LEFT(mseason,1)
**&&       msite = blbsite
**&&? nTimes, nIndex2, mSeason, mMonth
**&&wait
**&&   ELSE
**&&   ENDIF
**&&   DO SetPaths
**&&   DO OpenFiles
**&&   DO CalcWts
**&&   SELECT wts
**&&   COPY TO mPathTemp + mseason + "weights.dbf" FOX2X

** Allow chance to bail out
   IF BailOut
       CANCEL
   ENDIF
*   WAIT WINDOW AT 10,40 "Press 'C' to continue" + CHR(13) + CHR(10) + "Any other
   key to stop" TO mcont
*   IF NOT INLIST (mcont, "C", "c")
*       CANCEL
*   ENDIF

*****

```

```

*           Main Program           *
*****
      SET TALK OFF
      DO Artifacts                && Get the artifact information
      DO getconerr                && Get the "standard" uncertainty data
*ON ERROR Do Cleanup

      SET TALK OFF
      SET VIEW OFF
      DO Merge                    && Calculate and merge the data from the
      various databases

* Process PM10 speciation samples
      DO SITE9                    && Merge the PM10 mass data into the
      corresponding SITE1 sites

* Moved SiteX processing to Cleanup

* DO QACheck
      DO CleanUp                  && Clean up loose ends and delete
      temporary files

      ENDFOR
ENDDO
CANCEL

*****
PROCEDURE getconerr

PUBLIC mmfvol, mcocl, mcono2, mcono3, mcoso4, mconh4, mcool, mcoo2, mcoo3, mcoo4,
      mcoop, mcoe1, mcoe2, mcoe3, mcoso2
ON ERROR DO COERERR
USE mPATHSUPP + 'CONTERR.DBF' IN 0 ALIAS err
SELECT err
GO BOTTOM
ON ERROR
mmfvol = err.MFVOL
mcocl  = err.CL
mcono2 = err.NO2
mcono3 = err.NO3
mcoso4 = err.SO4
mconh4 = err.NH4
mcool  = err.O1
mcoo2  = err.O2
mcoo3  = err.O3
mcoo4  = err.O4
mcoop  = err.OP
mcoe1  = err.E1
mcoe2  = err.E2
mcoe3  = err.E3
mcoso2 = err.SO2
mdlcl  = err.CLmdl
mdlno2 = err.NO2mdl
mdlno3 = err.NO3mdl
mdlso4 = err.SO4mdl
mdlh4  = err.NH4mdl
USE
RETURN
*****
PROCEDURE COERERR
WAIT WINDOW mPATHSUPP + 'CONTERR.DBF is already open' TIMEOUT 1
IF LASTKEY() = 23
      DEACTIVATE WINDOW ALL
      CLOSE DATABASES
      SET COLOR SET TO          && Set the color back to default
      QUIT
ELSE
      RETRY                    &&
ENDIF
RETURN
*****
PROCEDURE MGCALERR

```

```

WAIT WINDOW mPATHCAL+ 'IMPMGCAL.DBF is already open' TIMEOUT 1
IF LASTKEY() = 23
    DEACTIVATE WINDOW ALL
    CLOSE DATABASES
    SET COLOR SET TO                                && Set the color back to default
    QUIT
ELSE
    RETRY                                           &&
ENDIF
*****
PROCEDURE PSCALERR
WAIT WINDOW mPATHCAL+ 'IMPPSCAL.DBF is already open' TIMEOUT 1
IF LASTKEY() = 23
    DEACTIVATE WINDOW ALL
    CLOSE DATABASES
    SET COLOR SET TO                                && Set the color back to default
    QUIT
ELSE
    RETRY                                           &&
ENDIF
*****

```

C.2. GetMonth

```
* This procedure displays a window to select a month and year to build
* the IMPROVE database.
*
* Parameters returned:
* mMonth:   Character (width 2) e.g. "12" for December
* mQTR:     Character (width 1) e.g. "B" for summer season
* mSeason:  Character (width 3) e.b. "B03" for summer 2003
*

PROCEDURE getmonth

**CLEAR

PUBLIC mMonth, mQtr, mSeason, aMonth, mYear
PUBLIC mPMonth, mYearMonth
STORE "00" to mMonth
STORE "000" to mSeason
DIMENSION gaMonth(12), gaYear(20), cQtr(12)
STORE "0000" TO mYear

cQtr(12) = "D"
cQtr( 1) = "D"
cQtr( 2) = "D"
cQtr( 3) = "A"
cQtr( 4) = "A"
cQtr( 5) = "A"
cQtr( 6) = "B"
cQtr( 7) = "B"
cQtr( 8) = "B"
cQtr( 9) = "C"
cQtr(10) = "C"
cQtr(11) = "C"
FOR gnCount = 1 to 12  && Fill the month array
    STORE cmonth(date(year(date()),gncount,1)) TO gaMonth(gnCount)
NEXT
FOR gnCount = 1 to 20  && Fill the year array
    STORE str(1987+gncount,4,0) TO gaYear(gnCount)
NEXT

frmMyForm = CREATEOBJECT('Form')  && Create a Form
frmMyForm.Caption = "Build data for month"
frmMyForm.Closable = .f.  && Disable the Control menu box
frmMyForm.Move(150,10,550,400)  && Move the form
frmMyForm.AddObject('ListBox1','lstMyListBox')  && Add ListBox control
frmMyForm.AddObject('ListBox2','2ndMyListBox')  && Add ListBox control
frmMyForm.AddObject('cmbCommand1','cmdMyCmdBtn')  && Add "Continue" Command
    button
frmMyForm.AddObject('cmbCommand2','QuitBtn')  && Add "Quit" Command button
frmMyForm.AddObject('MyLabel', 'Caution')
*frmMyForm.AddObject('Label1', 'SiteLabel')
*frmMyForm.AddObject('mSite', 'SiteBox')

frmMyForm.ListBox1.RowSourceType = 5  && Specifies an array
frmMyForm.ListBox1.RowSource = 'gaMonth' && Array containing listbox items
frmMyForm.ListBox2.RowSourceType = 5  && Specifies an array
frmMyForm.ListBox2.RowSource = 'gaYear' && Array containing listbox items

*frmMyForm.cmbCommand1.Visible = .T.  && "Continue" Command button visible
frmMyForm.cmbCommand2.Visible = .T.  && "Quit" Command button visible
frmMyForm.ListBox1.Visible = .T.  && "List Box visible
frmMyForm.MyLabel.Visible = .T.
*frmMyForm.ListBox2.Visible = .F.  && "List Box not visible

frmMyForm.SHOW  && Display the form
frmMyForm.ListBox2.Visible = .F.  && "List Box not visible
READ EVENTS      && Start event processing
```

```

DEFINE CLASS Caution AS Label
  Caption = "  Be sure the following files are up to date" ;
  + CHR(13) + CHR(10) + "          - Elements" ;
  + CHR(13) + CHR(10) + "          - Ions" ;
  + CHR(13) + CHR(10) + "          - Carbon" ;
  + CHR(13) + CHR(10) + "          - Laser"
  Left = 270
  Top = 20
  Height = 80
  Width = 240
  BorderStyle = 1
  BackStyle = 1
  FontBold = .T.
  BackColor = RGB(128, 255, 255)
  ForeColor = RGB(128, 0, 0)
ENDEDEFINE

DEFINE CLASS cmdMyCmdBtn AS CommandButton  && Create Command button
  Caption = '\<Continue'  && Caption on the Command button
  Cancel = .T.  && Default Cancel Command button (Esc)
  Left = 10  && Command button column
  Top = 310  && Command button row
  Height = 25  && Command button height

  PROCEDURE Click
    IF mMonth = "00" OR mSeason = "000"
**      CLEAR
      ? 'Select MONTH and YEAR'
      Wait
**      CLEAR
      frmMyForm.cmbCommand1.Visible = .T.  && "Continue" Command button visible
      frmMyForm.cmbCommand2.Visible = .T.  && "Quit" Command button visible
      frmMyForm.ListBox1.Visible = .T.  && "List Box visible
      frmMyForm.ListBox2.Visible = .F.  && "List Box visible
      frmMyForm.SHOW
    ELSE
      CLEAR EVENTS  && Stop event processing, close Form
*      CLEAR  && Clear main Visual FoxPro window
      RELEASE gaMonth, gaYear, cQtr
    ENDIF
    WAIT CLEAR
ENDEDEFINE

DEFINE CLASS QuitBtn AS CommandButton  && Create Command button
  Caption = '\<Quit'  && Caption on the Command button
  Cancel = .T.  && Default Cancel Command button (Esc)
  Left = 10  && Command button column
  Top = 260  && Command button row
  Height = 25  && Command button height

  PROCEDURE Click
    WAIT CLEAR
    SET SAFETY ON
    POP KEY ALL
    IF INLIST(mUser, "ashbaugh", "sengupta")
      IF mCalled
        RETURN TO master
      ENDIF
      CANCEL
    ENDIF
    QUIT
  ENDPROC
ENDEDEFINE

DEFINE CLASS lstMyListBox AS ListBox  && Create ListBox control
  Left = 10  && List Box column
  Top = 25  && List Box row
  MultiSelect = .F.  && Don't allow selecting more than 1 item
  Height = 210

  PROCEDURE Click
    ACTIVATE SCREEN

```

```

** CLEAR
FOR nCnt = 1 TO ThisForm.ListBox1.ListCount
  IF ThisForm.ListBox1.Selected(nCnt)  && Is item selected?
    aMonth = gaMonth(nCnt)
    mMonth = IIF(nCnt<10,"0"+str(nCnt,1,0),str(nCnt,2,0))
    mYearMonth = mYear + mMonth
    mQtr = cQtr(nCnt)
    mPMonth = IIF(nCnt=1,"12", IIF(nCnt<11,"0"+str(nCnt-1,1,0),str(nCnt-
1,2,0)))
    frmMyForm.ListBox2.Visible = .T.  && "List Box visible
  ENDIF
ENDIF
ENDFOR
ENDDDEFINE

DEFINE CLASS 2ndMyListBox AS ListBox  && Create ListBox control
  Left = 140                && List Box column
  Top = 25                  && List Box row
  MultiSelect = .F.        && Don't allow selecting more than 1 item
  Height = 355

PROCEDURE Click
  ACTIVATE SCREEN
** CLEAR
FOR mCnt = 1 TO ThisForm.ListBox2.ListCount
  IF ThisForm.ListBox2.Selected(mCnt)  && Is item selected?
    ? aMonth+ " "+GaYear(mCnt)
    STORE GaYear(mCnt) TO mYear
    mYearMonth = mYear + mMonth
    mSeason = mQtr + IIF(val(mMonth)>2,;
right(ThisForm.ListBox2.List(mCnt),2),;
right(ThisForm.ListBox2.List(mCnt-1),2))
    frmMyForm.cmbCommand1.Visible = .T.  && "Continue" Command button visible
  ENDIF
ENDIF
ENDFOR
ENDDDEFINE

```

C.3. SelectSite

```
PROCEDURE SelectSite

*****
* This procedure opens a list to select a site name
* to process for output data
*
* Returns msite
*
*****

PUBLIC mSite

USE &mlogs ALIAS logs
SELECT logs
GO TOP
*mSite = logs.site
SELECT DISTINCT Logs.site;
  FROM logs;
  ORDER BY Logs.site;
  INTO TABLE Sites
INSERT INTO Sites (site) values (" ALL")
INDEX ON site TAG site
GO TOP
SELECT logs
USE

frmInput = CREATEOBJECT('Form')
frmInput.closable = .F.
frmInput.movable = .T.
frmInput.BorderStyle = 3
frmInput.move(100,10,400,200)
frmInput.Caption = "Select site"
frmInput.Name = "Form1"
frmInput.AddObject('Label1', 'SiteLabel')
frmInput.AddObject('mSite', 'SiteBox')
*frmInput.AddObject('Label2', 'SeasonLabel')
*frmInput.AddObject('mSeason', 'SeasonBox')
frmInput.AddObject('Command1', 'ContinueBtn')
frmInput.AddObject('QuitBtn', 'QuitBtn1')
frmInput.Label1.Visible = .T.
frmInput.mSite.Visible = .T.
*frmInput.Label2.Visible = .T.
*frmInput.mSeason.Visible = .T.
frmInput.Command1.Visible = .T.
frmInput.QuitBtn.Visible = .T.

frmInput.show
READ EVENTS
RETURN

* Object definitions for display form follow
DEFINE CLASS SiteLabel AS Label
  AutoSize = .F.
*   FontBold = .T.
  FontSize = 9
  Alignment = 1
  Caption = "Site Name:"+chr(13)+"('ALL' for report)"
  Height = 40
  Left = 35
  Top = 24
  Width = 93
  Name = "Label1"
ENDDDEFINE

DEFINE CLASS SiteBox AS ListBox
*   FontBold = .T.
  FontSize = 9
  Value = sites.Site
```

```

ControlSource = sites.site
Top = 10
Left = 144
Width = 75
Height = 110
MaxLength = 5
SpecialEffect = 0
ColumnCount = 1
Name = "mSite"
SelStart = 1
Sellength = 5
SelectOnEntry = .T.
RowSourceType = 3
RowSource = "select site from sites into cursor site ORDER BY site"
DisplayValue = 1
NumberOfElements = 180
InputMask = "!!!!!"
ReadOnly = .F.
IncrementalSearch = .T.
PROCEDURE VALID
    m.msite = site.site
    CLEAR EVENTS
    thisform.release
    WAIT CLEAR
ENDPROC
ENDDDEFINE

DEFINE CLASS ContinueBtn as CommandButton
    Top = 24
    Left = 240
    Height = 25
    Width = 121
*   FontBold = .T.
    FontSize = 10
    Cancel = .F.
    Caption = "\<Continue"
    Default = .T.
    TerminateRead = .F.
    Name = "Command1"

    PROCEDURE Click
*       IIF(len(msite)=0, m.msite=" ", m.msite = site.site)
        m.msite = site.site
        IF SELECT("Flash")>0
            SELECT Flash
            USE
        ENDIF
        CLEAR EVENTS
        thisform.release
        WAIT CLEAR
    ENDPROC
ENDDDEFINE

DEFINE CLASS QuitBtn1 as CommandButton
    Top = 85
    Left = 240
    Height = 25
    Width = 121
*   FontBold = .T.
    FontSize = 10
    Cancel = .F.
    Caption = "\<Quit"
    TerminateRead = .F.
    Name = "QuitBtn"

    PROCEDURE Click
*       IF SELECT("Flash")>0
*           SELECT Flash
*           USE
*       ENDIF
*       ON ERROR
*       SET DEFAULT TO &DefaultDir

```



```
BailOut = .T.  
CLEAR EVENTS  
thisform.release  
WAIT CLEAR  
* Cancel  
ENDPROC  
ENDDFINE
```

C.4. SetPaths

```
PROCEDURE setpaths
* Define data paths and file names

PUBLIC mweights, mcarbon, mions, mlaser, mele &&, mlogs, mlogpath
PUBLIC mso2, mmag, mvac, mout, mconterr, mart, mouttemp, msites
PUBLIC mpcarbon, mpions, moutfinl, mfinl, mspcl, mfinlxls
PUBLIC mwtspath, mcarpath, mionpath, mlaspath,;
    melepath, mso2path, mcalpath, moutpath, mpathsupp, ;
    mPathTemp, mNewWtTemp, mNewWt, mFlash, mFlowFlags

* 11/4/2003 - may have corrected the problem noted here.
* The Temporary directory mPathTemp may cause program failure
* if there is not an M: drive. The following code will create
* mPathTemp if it does not exist on the M: drive, but will do
* nothing if the M: drive does not exist. Then references to
* mPathTemp will fail.

IF DIRECTORY("M:\")
    mPathTemp = "m:\improve\Temp\"
    IF NOT DIRECTORY("&mPathTemp")
        RUN MD &mPathTemp
    ENDIF
    IF NOT DIRECTORY("m:\IMPROVE\PROGRAMS")
        RUN MD "m:\IMPROVE\PROGRAMS"
    ENDIF
    SET PATH TO "m:\IMPROVE\PROGRAMS", mdisk + "IMPROVE\PROGRAMS"
ELSE
    SET PATH TO mdisk + "IMPROVE\PROGRAMS"
    IF NOT DIRECTORY(mdisk + "improve\Temp\")
        tdisk = &mdisk + "m:\improve\Temp\"
        RUN MD &tdisk
    ENDIF
    mPathTemp = mdisk + "improve\Temp\"
ENDIF
mrootpath = mdisk + "improve\"
*SET DEFAULT TO mrootpath + "programs\"
SET DEFAULT TO "m:\improve\programs\"

mlogpath = mrootpath + "logs\"
mwtspath = mrootpath + "weights\"
mcarpath = mrootpath + "carbon\"
mionpath = mrootpath + "ions\"
mlaspath = mrootpath + "laser\"
melepath = mrootpath + "pixexrf\"
mso2path = mrootpath + "so2\"
mcalpath = mrootpath + "sitecal\"
moutpath = mrootpath + "totalDBF\"
mPATHSUPP = mrootpath + "Support\"
mFlashPath= mrootpath + "FlashDBF\Reports\"
*mQAPath = mrootpath + "QAPlots\QA Data\"
mQAPath = mpathtemp

*Do GetMonth
*mlogs = mlogpath + mseason + "logs.dbf"
*DO SelectSite

mlogs = mlogpath + mseason + "logs.dbf"
mweights = mwtspath + mseason + "wts.dbf"
mcarbon = mcarpath + mseason + "car" + ".dbf"
mions = mionpath + mseason + "ions" + ".dbf"
mlaser = mlaspath + mseason + "laser.dbf"
mele = melepath + mseason + "ele" + ".dbf"
mso2 = mso2path + mseason + "so2.dbf"
mmag = mcalpath + "impmgcal.dbf"
mvac = mcalpath + "imppscal.dbf"
mFlash = mFlashPath + mseason + "flows.dbf"
IF mSite = " ALL"
    mout = mQAPath + mseason + "QA" + mmonth + ".dbf"
```

```
mfinl = moutpath + mseason + "tot" + mmonth + ".dbf"
mspcl = moutpath + mseason + "spc" + mmonth + ".dbf"
ELSE
  mout = mQAPath + mSite + mseason + "QA" + mmonth + ".dbf"
  mfinl = mQAPath + mSite + mseason + mmonth + ".dbf"
  mspcl = mQAPath + mSite + "spc" + mseason + mmonth + ".dbf"
ENDIF
*mfinlxls = mPathTemp + mseason + "tot" + mmonth + ".xls"
mconterr = mpathsupp + "conterr.dbf"
mart = mpathsupp + "arthist.dbf"
mouttemp = mPathTemp + "outtemp.dbf"
moutfinl = mPathTemp + "FinalTot.dbf"
msites = mcalpath + "sitefile.dbf"
mFlowFlags = mFlashPath + "Flow_Flags_All.dbf"
return
```

C.5. OpenFiles

PROCEDURE openfiles

* Open all databases, set indexes, and set relations

CLOSE DATABASES ALL

mDefPath = SYS(5)+SYS(2003)

SET DEFAULT TO &mPathTemp

*Delete temporary files if they exist

IF FILE('tlogs.dbf')

 DELETE FILE tlogs.dbf

ENDIF

IF FILE('tcarbon.dbf')

 DELETE FILE tcarbon.dbf

ENDIF

IF FILE('tions.dbf')

 DELETE FILE tions.dbf

ENDIF

IF FILE('tlaser.dbf')

 DELETE FILE tlaser.dbf

ENDIF

IF FILE('tele.dbf')

 DELETE FILE tele.dbf

ENDIF

IF FILE('tso2.dbf')

 DELETE FILE tso2.dbf

ENDIF

IF FILE('tmag.dbf')

 DELETE FILE tmag.dbf

ENDIF

IF FILE('tvac.dbf')

 DELETE FILE tvac.dbf

ENDIF

IF FILE('twts.dbf')

 DELETE FILE twts.dbf

ENDIF

IF FILE('tsites.dbf')

 DELETE FILE tsites.dbf

ENDIF

*Copy data to temporary files

USE &mlogs

* Select only the current month

 COPY TO tlogs.dbf FOR MONTH(samdat) = VAL(mmonth)

USE &mweights

 COPY ALL TO twts

USE &mcarbon

* Select only the current month for artifact calculation

* Modified 7/7/05 by Lowell to copy all months. A filter

* will be set in Artifacts to use the correct time period.

 COPY all TO tcarbon && FOR MONTH(samdat) = VAL(mmonth)

USE &mions

* Select only the current month for artifact calculation

* Modified 7/7/05 by Lowell to copy all months. A filter

* will be set in Artifacts to use the correct time period.

 COPY all TO tions && FOR MONTH(samdat) = VAL(mmonth)

USE &mlaser

 COPY ALL TO tlaser

USE &mele

 COPY ALL TO tele

USE &mso2

 COPY ALL TO tso2

USE &mmag

 COPY ALL TO tmag

USE &mvac

 COPY ALL TO tvac

USE &msites

 COPY ALL TO tsites

```

USE
IF FILE('&mout')
  IF FILE('tout.dbf')
    DELETE FILE tout.dbf
    DELETE FILE tfinl.dbf
  ENDIF
  CLOSE DATABASES ALL
  DELETE FILE &mout
  DELETE FILE &mfinl
  DELETE FILE &mspcl
* DELETE FILE &mNewWt
* CREATE &mNewWt FROM &mNewWtTemp
* COPY TO tNewWt.dbf FOX2X
* USE
  CREATE &mout FROM &mouttemp
* IF VAL(mYearMonth) > 200412
*   ALTER TABLE add COLUMN opt N(9,1)
* ENDIF
  COPY TO tout.dbf FOX2X
  USE
  CREATE mspcl FROM &moutfinl
* IF VAL(mYearMonth) > 200412
*   ALTER TABLE add COLUMN opt N(9,1)
* ENDIF
  COPY TO &mspcl FOX2X
  CREATE mfinl FROM &moutfinl
* IF VAL(mYearMonth) > 200412
*   ALTER TABLE add COLUMN opt N(9,1)
* ENDIF
  COPY TO &mfinl FOX2X
  COPY TO tfinl.dbf FOX2X
  USE
ELSE
* CREATE &mNewWt FROM &mNewWtTemp
* COPY TO tNewWt.dbf FOX2X
* USE
  CREATE &mout FROM &mouttemp
  COPY TO tout.dbf FOX2X
  USE
  CREATE mspcl FROM &moutfinl
  COPY TO &mspcl FOX2X
  CREATE mfinl FROM &moutfinl
  COPY TO &mfinl FOX2X
  COPY TO tfinl.dbf FOX2X
  USE
ENDIF
*COPY FILE &mconterr TO terr.dbf

*Open temporary files and assign alias names
USE tlogs IN 0 ALIAS logs
USE twts IN 0 ALIAS wts
USE tcarbon IN 0 ALIAS carbon
USE tions IN 0 ALIAS ions
USE tlaser IN 0 ALIAS laser
USE tele IN 0 ALIAS elements
USE tso2 IN 0 ALIAS so2
USE tmag IN 0 ALIAS mag
USE tvac IN 0 ALIAS vac
USE tsites IN 0 ALIAS sites
*SELECT DISTINCT Logs.site;
* FROM logs;
* ORDER BY Logs.site;
* INTO CURSOR Sites

USE tout IN 0 ALIAS out
  SELECT out
  DELETE ALL
  PACK
USE tfinl IN 0 ALIAS finl
  SELECT finl
  DELETE all
  PACK

```

```

*USE tNewWt IN 0 ALIAS NewWt
* SELECT NewWt
* DELETE all
* PACK
*USE terr IN 0 ALIAS err

USE m:\improve\programs\nh4sites IN 0 ALIAS nh4sites
SELECT nh4sites
INDEX ON site TAG site

* Set indexes
SELECT logs
INDEX ON trim(site)+ dtoc(samdat)+ strtim TAG sitdatim
SELECT wts
INDEX ON TRIM(site)+ dtoc(samdat)+ strtim TAG sitdatim
SELECT carbon
INDEX ON trim(site)+dtoc(samdat)+strtim TAG sitdatim
INDEX ON O1tc tag O1
INDEX ON O2tc tag O2
INDEX ON O3tc tag O3
INDEX ON O4tc tag O4
INDEX ON OPtc tag OP
INDEX ON E1tc tag E1
INDEX ON E2tc tag E2
INDEX ON E3tc tag E3
SET ORDER TO sitdatim
SELECT ions
INDEX ON trim(site)+dtoc(samdat)+strtim TAG sitdatim
INDEX ON cl tag CL
INDEX ON NO2 tag NO2
INDEX ON NO3 tag NO3
INDEX ON SO4 tag SO4
INDEX ON NH4 tag NH4
SET ORDER TO sitdatim
SELECT laser
INDEX ON trim(site)+dtoc(samdat)+strtim TAG sitdatim
SELECT elements
INDEX ON trim(site)+dtoc(samdat)+strtim TAG sitdatim
SELECT so2
INDEX ON trim(site)+dtoc(samdat)+strtim TAG sitdatim
INDEX ON so4 TAG so4
SELECT mag
INDEX ON sitecode TAG site
SELECT vac
INDEX ON sitecode TAG site
SELECT sites
INDEX ON site TAG site
SELECT logs

* Relate databases as needed
SET RELATION TO logs.site+ DTOC(logs.samdat)+ logs.strtim INTO carbon ADDITIVE
SET RELATION TO logs.site+ DTOC(logs.samdat)+ logs.strtim INTO ions ADDITIVE
SET RELATION TO logs.site+ DTOC(logs.samdat)+ logs.strtim INTO elements ADDITIVE
SET RELATION TO Logs.so2site+DTOC(Logs.samdat)+Logs.strtim INTO So2 ADDITIVE
SET RELATION TO logs.site+ DTOC(logs.samdat)+ logs.strtim INTO laser ADDITIVE
SET RELATION TO logs.site INTO mag ADDITIVE
SET RELATION TO logs.site INTO vac ADDITIVE
SET RELATION TO logs.site INTO Sites ADDITIVE
SET RELATION TO site INTO Nh4sites ADDITIVE
IF logs.samdat>={9/1/2004}
  USE &mFlash IN 0 ALIAS flash
  SELECT flash
  INDEX ON LEFT(site,4) + DTOC(date) + "0000" TAG site4date
  INDEX ON site + DTOC(date) + "0000" TAG sitedate
  SET ORDER TO sitedate
  SELECT logs
  SET RELATION TO logs.site + DTOC(logs.samdat) INTO flash additive
*
* This section was moved to the VOLUMES routine so as to set the relationship
* depending on the site name. SITEX files have to use site4date. All others
* use sitedate.
*

```

```
* SET RELATION TO LEFT(logs.site,4) + DTOC(logs.samdat) INTO flash additive
ENDIF

* Open the flow flags file and relate it to logs for processing in Volumes
USE &mFlowFlags IN 0 ALIAS FlowFlags
SELECT FlowFlags
INDEX ON site+DTOC(date) TAG sitedate
SET ORDER TO SITEDATE    && SITE+DTOC(DATE)
SELECT logs
SET RELATION TO logs.site+dtoc(logs.samdat) INTO FlowFlags ADDITIVE

SET DEFAULT TO &mDefPath
RETURN
```

C.6. CalcWts

PROCEDURE CalcWts

* Recalculates prewgt, postwt, and diff in the WTS file for selected quarter

PUBLIC mPrewght, mPstwght

mDefPath = SYS(5)+SYS(2003)
SET DEFAULT TO &mPathTemp

*Delete temporary files if they exist

IF FILE('tbal25.dbf')
DELETE FILE tbal25.dbf

ENDIF

IF FILE('tbal30.dbf')
DELETE FILE tbal30.dbf

ENDIF

IF FILE('tbal30a.dbf')
DELETE FILE tbal30a.dbf

ENDIF

IF FILE('tbal31a.dbf')
DELETE FILE tbal31a.dbf

ENDIF

USE "u:\improve\support\baleq25.dbf" IN 0 Alias bal
SELECT bal

COPY ALL TO tbal25

USE "u:\improve\support\baleq30.dbf"

COPY ALL TO tbal30

USE "u:\improve\support\baleq30a.dbf"

COPY ALL TO tbal30a

USE "u:\improve\support\baleq31a.dbf"

COPY ALL TO tbal31a

USE

USE tbal25 IN 0 ALIAS bal25

USE tbal30 IN 0 ALIAS bal30

USE tbal30a IN 0 ALIAS bal30a

USE tbal31a IN 0 ALIAS bal31a

SELECT wts

lPrewght = .F.

FOR n = 1 TO FCOUNT("wts")

IF FIELD(n) = "PREWGHT"

lPrewght = .T.

ENDIF

ENDFOR

IF NOT lPrewght

ALTER TABLE wts ADD COLUMN Prewght N(6,3)

ALTER TABLE wts ADD COLUMN Postwght N(6,3)

ALTER TABLE wts ADD COLUMN Diff N(6,3)

REPLACE ALL Prewght WITH Prebalwt

REPLACE ALL Postwght WITH Pstbalwt

REPLACE ALL Diff WITH Postwght-Prewght

ENDIF

GO TOP

DO WHILE NOT EOF('wts')

DO GETPREWT

DO GETPSTWT

IF (wts.prewght <> 0 .and. (wts.prewght <> mPrewght)) .or. ;

(wts.postwght <> 0 .and. (wts.postwght <> mPstwght)) .or. ;

wts.diff <> mPstwght - mPrewght .and. ;

wts.postwght<>0

REPLACE wts.prewght WITH mPrewght

REPLACE wts.postwght WITH mPstwght

REPLACE wts.diff WITH mPstwght - mPrewght


```

        ENDIF
        skip in wts
ENDDO

* Clean up files
SELECT bal25
USE
SELECT bal30
USE
SELECT bal30a
USE
SELECT bal31a
USE
DELETE FILE tbal25.dbf
DELETE FILE tbal30.dbf
DELETE FILE tbal30a.dbf
DELETE FILE tbal31a.dbf

RETURN

*****
Procedure GETPREWT
SELECT wts
PUBLIC mprecaldate
mPRINDATE = wts.prindate
mPRINTIME = wts.printime
SET DELETED ON
IF FIELD(1,"wts") = "SITE"
    STORE "Prebalwt" to Fprebalwt    && Select the balance prewt regardless of
    field name
ELSE
    STORE FIELD(8,"wts") to Fprebalwt
ENDIF
mprebalwt = &Fprebalwt
* IF mseason="B02"
*   mprebalwt = wts.cahn25wt
* ELSE
*   mprebalwt = wts.prebalwt
* ENDIF
Do Case
Case wts.prbalance = '31' .or. mprebalwt = 0 .or. wts.SAMDAT < {^1996-03-01}
    && Use the PREWGHT for all before A96
    mPREWGHT = wts.prewght
    && This is because of the 20 mg addition stuff on the
    25
    mprecaldate = wts.prindate
Case wts.prbalance = '25'
    SELECT bal25
    Locate For BEGDAT > mPRINDATE
    If Found()
        Skip - 1
    Else
        Goto Bottom
    Endif
    mTAREINT = tareint
    mTARESLP = tareslp
    mPREWGHT=round(mTAREINT+(mprebalwt*mTARESLP),3)
    mprecaldate = begdat
Case wts.prbalance = '30'
    Sele bal30
    Locate For BEGDAT > mPRINDATE
    If Found()
        Skip - 1
    Else
        Goto Bottom
    Endif
    mTAREINT = TAREINT
    mTARESLP = TARESLP
    mPREWGHT=round(mTAREINT+(mprebalwt*mTARESLP),3)
    mprecaldate = begdat
Case wts.prbalance = '3A'
    Sele bal30a

```

```

Locate For BEGDAT > mPRINDATE
If Found()
    Skip - 1
Else
    Goto Bottom
Endif
mTAREINT = TAREINT
mTARESLLP = TARESLLP
mPREWGHT=round(mTAREINT+(mprebalwt*mTARESLLP),3)
mprecaldate = begdat
Case wts.prbalance = '1A'
Sele bal31a
Locate For BEGDAT > mPRINDATE
If Found()
    Skip - 1
Else
    Goto Bottom
Endif
mTAREINT = TAREINT
mTARESLLP = TARESLLP
mPREWGHT=round(mTAREINT+(mprebalwt*mTARESLLP),3)
mprecaldate = begdat
OTHERWISE
mPREWGHT=mprebalwt
mprecaldate = begdat
Endcase
SELECT wts
Return
*****
Procedure GETPSTWT
SELECT wts
PUBLIC mpstcaldate
mPOINDATE = wts.poindate
mPOINTIME = wts.pointime
Set Dele on
IF FIELD(1,"wts") = "SITE"
    STORE "Pstbalwt" to Fpstbalwt    && Select the balance pstwt regardless of
field name
ELSE
    STORE FIELD(10,"wts") to Fpstbalwt
ENDIF
mpstbalwt = &Fpstbalwt
Do Case
Case wts.pobalance = '25'
    SELECT bal25
    Locate For BEGDAT > mPoINDATE
    If Found()
        Skip - 1
    Else
        Goto Bottom
    Endif
    mTAREINT = tareint
    mTARESLLP = taresllp
    mPstwght =ROUND(mTAREINT+(mpstbalwt*mTARESLLP),3)
    mpstcaldate = begdat
Case wts.pobalance = '30'
    Sele bal30
    Locate For BEGDAT > mPOINDATE
    If Found()
        Skip - 1
    Else
        Goto Bottom
    Endif
    mTAREINT = TAREINT
    mTARESLLP = TARESLLP
    mPstwght =ROUND(mTAREINT+(mpstbalwt*mTARESLLP),3)
    mpstcaldate = begdat
Case wts.pobalance = '3A'
    Sele bal30a
    Locate For BEGDAT > mPOINDATE
    If Found()
        Skip - 1

```

```

Else
    Goto Bottom
Endif
mTAREINT = TAREINT
mTARESLP = TARESLP
mPstwght =ROUND(mTAREINT+(mpstbalwt*mTARESLP),3)
mpstcaldate = begdat
Case wts.pobalance = '1A'
    Sele bal31a
    Locate For BEGDAT > mPOINDATE
    If Found()
        Skip - 1
    Else
        Goto Bottom
    Endif
    mTAREINT = TAREINT
    mTARESLP = TARESLP
    mPstwght =ROUND(mTAREINT+(mpstbalwt*mTARESLP),3)
    mpstcaldate = begdat
Otherwise
    POBALANCE = ' '
    mPstwght = wts.postwght
    addition stuff on the 25
    mpstcaldate = wts.poindate
Endcase
Return

```

C.7. Artifacts

PROCEDURE artifacts

* Calculate ions artifacts

```
*****
* Code added 7/7/2005 by Lowell to calculate quarterly artifacts
* if a month prior to June 2002 (B02) is selected. This allows use
* of the correct quarterly artifacts for all dates prior to B02.
*
* Modified 9/23/2005 by Lowell to calculate separate artifacts in B00
* for 25mm and 37mm filters. Applies only to B00 quarter.
*****

RELEASE mCLStd, mNO2Std, mNO3Std, mSO4Std, mNH4Std
RELEASE mCLArt, mNO2Art, mNO3Art, mSO4Art, mNH4Art
RELEASE mO1Std, mO2Std, mO3Std, mO4Std, mOPStd, mE1Std, mE2Std, mE3Std
RELEASE mO1Art, mO2Art, mO3Art, mO4Art, mOPArt, mE1Art, mE2Art, mE3Art
RELEASE mO1BStd, mO2BStd, mO3BStd, mO4BStd, mOPBStd, mE1BStd, mE2BStd, mE3BStd
RELEASE mCLOStd, mNO2oStd, mNO3oStd, mSO4oStd, mNH4oStd      &&Previous month
RELEASE mCLOArt, mNO2oArt, mNO3oArt, mSO4oArt, mNH4oArt      &&Previous month
RELEASE mCLStd25, mNO2Std25, mNO3Std25, mSO4Std25, mNH4Std25
RELEASE mCLArt25, mNO2Art25, mNO3Art25, mSO4Art25, mNH4Art25
RELEASE mCLStd37, mNO2Std37, mNO3Std37, mSO4Std37, mNH4Std37
RELEASE mCLArt37, mNO2Art37, mNO3Art37, mSO4Art37, mNH4Art37
* RELEASE mO1oStd, mO2oStd, mO3oStd, mO4oStd, mOPoStd, mE1oStd, mE2oStd,
  mE3oStd      &&Previous month
* RELEASE mO1oArt, mO2oArt, mO3oArt, mO4oArt, mOPoArt, mE1oArt, mE2oArt,
  mE3oArt      &&Previous month

PUBLIC mCLStd, mNO2Std, mNO3Std, mSO4Std, mNH4Std
PUBLIC mCLArt, mNO2Art, mNO3Art, mSO4Art, mNH4Art
PUBLIC mO1Std, mO2Std, mO3Std, mO4Std, mOPStd, mE1Std, mE2Std, mE3Std
PUBLIC mO1Art, mO2Art, mO3Art, mO4Art, mOPArt, mE1Art, mE2Art, mE3Art
PUBLIC mO1BStd, mO2BStd, mO3BStd, mO4BStd, mOPBStd, mE1BStd, mE2BStd, mE3BStd
PUBLIC mCLOStd, mNO2oStd, mNO3oStd, mSO4oStd, mNH4oStd      &&Previous
  month
PUBLIC mCLOArt, mNO2oArt, mNO3oArt, mSO4oArt, mNH4oArt      &&Previous
  month
PUBLIC mCLStd25, mNO2Std25, mNO3Std25, mSO4Std25, mNH4Std25
PUBLIC mCLArt25, mNO2Art25, mNO3Art25, mSO4Art25, mNH4Art25
PUBLIC mCLStd37, mNO2Std37, mNO3Std37, mSO4Std37, mNH4Std37
PUBLIC mCLArt37, mNO2Art37, mNO3Art37, mSO4Art37, mNH4Art37
* PUBLIC mO1oStd, mO2oStd, mO3oStd, mO4oStd, mOPoStd, mE1oStd, mE2oStd,
  mE3oStd      &&Previous month
* PUBLIC mO1oArt, mO2oArt, mO3oArt, mO4oArt, mOPoArt, mE1oArt, mE2oArt,
  mE3oArt      &&Previous month

*IF logs.samdat<{^2002-06-01}      && Get quarterly artifact information from
  ARTHIST file*
* USE &mart IN 0 ALIAS art
* SELE art
* GO TOP
* NewQtr = .F.
* LOCATE FOR art.qtr=UPPER(mseason)
* IF EOF("art")
*   WAIT WINDOW AT 10,40 "&mseason not found in ARTHIST.DBF"
* ENDIF
* mCLArt = art.cl
* mNO2Art = art.no2
* mNO3Art = art.no3
* mSO4Art = art.so4
* mNH4Art = art.nh4
* mCLStd = art.dcl
* mNO2Std = art.dno2
* mNO3Std = art.dno3
* mSO4Std = art.ds04
* mNH4Std = art.dnh4
* mO1Art = art.o1
* mO2Art = art.o2
```

```

* mO3Art = art.o3
* mO4Art = art.o4
* mOPArt = art.op
* mE1Art = art.e1
* mE2Art = art.e2
* mE3Art = art.e3
* mO1Std = art.do1
* mO2Std = art.do2
* mO3Std = art.do3
* mO4Std = art.do4
* mOPStd = art.dop
* mE1Std = art.del
* mE2Std = art.de2
* mE3Std = art.de3

SELECT ions
IF logs.samdat<{^2002-06-01} && Calculate quarterly artifacts
  SET FILTER TO STATUS="FB" AND .NOT. DELETED()
ELSE && Calculate artifact for a single month
  SET FILTER TO MONTH(samdat) = VAL(mmonth) AND STATUS="FB" AND .NOT. DELETED()
ENDIF

NumArt=1
IF mSeason="B00" && B00 uses 25mm and 37mm filters. Calculate separate
  artifacts
  NumArt = 2
  SET FILTER TO STATUS="FB" AND .NOT. DELETED() AND bdiam<>"37" && Calc 25mm
  artifact first
ENDIF

FOR Nart = 1 TO NumArt
  IF SYS(103) != 'ON'
    mTALK = .T.
  ELSE
    mTALK = .F.
  ENDIF
  SET TALK OFF
  COUNT TO ioncount

*Calculate standard deviations of Field Blanks
  CALCULATE STD(c1) TO mCLStd
  CALCULATE STD(no2) TO mNO2Std
  CALCULATE STD(no3) TO mNO3Std
  CALCULATE STD(SO4) TO mSO4Std
*CALCULATE Std(NH4) TO mNH4Std
  CALCULATE STD(nh4) TO mNH4Std FOR nh4>=0

* Calculate median values of Field Blanks
* Chloride
  SET ORDER TO c1
  GO TOP
  IF MOD(ioncount,2) <> 0
    SKIP (ioncount-1)/2
    mCLArt = c1
  ELSE
    SKIP ioncount/2-1
    mCLArt = c1
  SKIP
  mCLArt = (mCLArt + c1)/2
  ENDIF

* Nitrite
  SET ORDER TO no2
  GO TOP
  IF MOD(ioncount,2) <> 0
    SKIP (ioncount-1)/2
    mNO2Art = no2
  ELSE
    SKIP ioncount/2-1
    mNO2Art = no2
  SKIP

```

```

        mNO2Art = (mNO2Art + no2)/2
    ENDIF

* Nitrate
    SET ORDER TO no3
    GO TOP
    IF MOD(ioncount,2) <> 0
        SKIP (ioncount-1)/2
        mNO3Art = no3
    ELSE
        SKIP ioncount/2-1
        mNO3Art = no3
        SKIP
        mNO3Art = (mNO3Art + no3)/2
    ENDIF

* Sulfate
    SET ORDER TO SO4
    GO TOP
    IF MOD(ioncount,2) <> 0
        SKIP (ioncount-1)/2
        mSO4Art = SO4
    ELSE
        SKIP ioncount/2-1
        mSO4Art = SO4
        SKIP
        mSO4Art = (mSO4Art + SO4)/2
    ENDIF

* Ammonium
    SET ORDER TO nh4
    SET FILTER TO STATUS="FB" AND .NOT. DELETED() AND nh4 >= 0
    IF logs.samdat<{^2002-06-01} && Calculate quarterly artifacts
        SET FILTER TO STATUS="FB" AND .NOT. DELETED() AND nh4 >= 0
    ELSE
        && Calculate artifact for a single month
        SET FILTER TO MONTH(samdat) = VAL(mmonth) AND STATUS="FB" AND .NOT.
        DELETED() AND nh4 >= 0
    ENDIF
    COUNT TO nh4count
    GO TOP
    IF MOD(nh4count,2) <> 0
        SKIP (nh4count-1)/2
* mNH4Art = NH4
        mNH4Art = nh4
    ELSE
        SKIP nh4count/2-1
* mNH4Art = NH4
        mNH4Art = nh4
        SKIP
* mNH4Art = (mNH4Art + nh4)/2
        mNH4Art = (mNH4Art + nh4)/2
    ENDIF
    IF mSeason="B00"
        DO CASE
            CASE Nart=1
                calculate 37mm artifact
                && Store 25 mm artifact data and
                    mCLStd25 = mCLStd
                    mNO2Std25 = mNO2Std
                    mNO3Std25 = mNO3Std
                    mSO4Std25 = mSO4Std
                    mNH4Std25 = mNH4Std
                    mCLArt25 = mCLArt
                    mNO2Art25 = mNO2Art
                    mNO3Art25 = mNO3Art
                    mSO4Art25 = mSO4Art
                    mNH4Art25 = mNH4Art
                    SET FILTER TO STATUS="FB" AND .NOT. DELETED() AND bdiam="37"
                && Calc 37mm artifact first
            CASE Nart=2
                && Store 37 mm artifact data
                    mCLStd37 = mCLStd
                    mNO2Std37 = mNO2Std
                    mNO3Std37 = mNO3Std

```

```

        mSO4Std37 = mSO4Std
        mNH4Std37 = mNH4Std
        mCLArt37  = mCLArt
        mNO2Art37 = mNO2Art
        mNO3Art37 = mNO3Art
        mSO4Art37 = mSO4Art
        mNH4Art37 = mNH4Art

    ENDCASE
ENDIF
ENDFOR
SET FILTER TO
SET ORDER TO sitdatim

* Calculate SO2 artifacts
SELECT so2
SET FILTER TO STATUS="FB" AND .NOT. DELETED()
COUNT TO so2count
CALCULATE STD(SO4) TO mso2Std

SET ORDER TO SO4
GO TOP
IF MOD(so2count,2) <> 0
    SKIP (so2count-1)/2
    mSO2Art = SO4
ELSE
    IF so2count>0
        SKIP MAX(0,so2count/2-1)
        mSO2Art = SO4
        SKIP
        mSO2Art = (mSO2Art + SO4)/2
    ELSE
        *This is for the case of no FB
        mSO2Art = -999
    ENDIF
ENDIF

SET FILTER TO
SET ORDER TO sitdatim

* Calculate organic artifacts using Secondary Filters
SELECT carbon&& Carbon data file is already selected for a single month
IF logs.samdat<{^2002-06-01} && Calculate quarterly artifacts
    SET FILTER TO STATUS="CS" AND .NOT. DELETED()
ELSE && Calculate artifact for a single month
    SET FILTER TO MONTH(samdat) = VAL(mmonth) AND STATUS="CS" AND .NOT. DELETED()
ENDIF

COUNT TO carcount

* Calculate standard deviations of Secondary Filters
CALCULATE STD(O1TC) TO mO1Std
CALCULATE STD(O2TC) TO mO2Std
CALCULATE STD(O3TC) TO mO3Std
CALCULATE STD(O4TC) TO mO4Std
CALCULATE STD(OPTC) TO mOPStd
CALCULATE STD(E1TC) TO mE1Std
CALCULATE STD(E2TC) TO mE2Std
CALCULATE STD(E3TC) TO mE3Std

* Calculate median values of Secondary Filters
SET ORDER TO o1
GO TOP
IF MOD(carcount,2) <> 0
    SKIP (carcount-1)/2
    mO1Art = O1TC
ELSE
    SKIP carcount/2-1
    mO1Art = O1TC
    SKIP
    mO1Art = (mO1Art + O1TC)/2
ENDIF

```

```

SET ORDER TO o2
GO TOP
IF MOD(carcount,2) <> 0
  SKIP (carcount-1)/2
  mO2Art = O2TC
ELSE
  SKIP carcount/2-1
  mO2Art = O2TC
  SKIP
  mO2Art = (mO2Art + O2TC)/2
ENDIF

SET ORDER TO o3
GO TOP
IF MOD(carcount,2) <> 0
  SKIP (carcount-1)/2
  mO3Art = O3TC
ELSE
  SKIP carcount/2-1
  mO3Art = O3TC
  SKIP
  mO3Art = (mO3Art + O3TC)/2
ENDIF

SET ORDER TO o4
GO TOP
IF MOD(carcount,2) <> 0
  SKIP (carcount-1)/2
  mO4Art = O4TC
ELSE
  SKIP carcount/2-1
  mO4Art = O4TC
  SKIP
  mO4Art = (mO4Art + O4TC)/2
ENDIF

SET ORDER TO op
GO TOP
IF MOD(carcount,2) <> 0
  SKIP (carcount-1)/2
  mOPart = OPTC
ELSE
  SKIP carcount/2-1
  mOPart = OPTC
  SKIP
  mOPart = (mOPart + OPTC)/2
ENDIF

SET ORDER TO e1
GO TOP
IF MOD(carcount,2) <> 0
  SKIP (carcount-1)/2
  mE1Art = E1TC
ELSE
  SKIP carcount/2-1
  mE1Art = E1TC
  SKIP
  mE1Art = (mE1Art + E1TC)/2
ENDIF

SET ORDER TO e2
GO TOP
IF MOD(carcount,2) <> 0
  SKIP (carcount-1)/2
  mE2Art = E2TC
ELSE
  SKIP carcount/2-1
  mE2Art = E2TC
  SKIP
  mE2Art = (mE2Art + E2TC)/2
ENDIF

```



```

SET ORDER TO e3
GO TOP
IF MOD(carcount,2) <> 0
    SKIP (carcount-1)/2
    mE3Art = E3TC
ELSE
    SKIP carcount/2-1
    mE3Art = E3TC
    SKIP
    mE3Art = (mE3Art + E3TC)/2
ENDIF

IF logs.samdat<{^2002-06-01} && Calculate quarterly artifacts
    SET FILTER TO STATUS="FB" AND .NOT. DELETED()
ELSE && Calculate artifact for a single month
    SET FILTER TO MONTH(samdat) = VAL(mmonth) AND STATUS="FB" AND .NOT. DELETED()
ENDIF

* Calculate standard deviations of Field Blanks
CALCULATE STD(O1TC) TO mO1BStd
CALCULATE STD(O2TC) TO mO2BStd
CALCULATE STD(O3TC) TO mO3BStd
CALCULATE STD(O4TC) TO mO4BStd
CALCULATE STD(OPTC) TO mOPBStd
CALCULATE STD(E1TC) TO mE1BStd
CALCULATE STD(E2TC) TO mE2BStd
CALCULATE STD(E3TC) TO mE3BStd

SET FILTER TO STATUS="CP" AND .NOT. DELETED()
SET ORDER TO sitdatim

IF mTALK
    SET TALK ON
ENDIF

USE &mart IN 0 ALIAS art
SELECT art
GO TOP
NewQtr = .F.
LOCATE FOR art.qtr=UPPER(mSeason)
IF EOF("art")
    NewQtr = .T.
ENDIF
LOCATE FOR art.qtr=UPPER(mSeason) AND art.MONTH=VAL(mmonth)
IF EOF("art")
    APPEND BLANK
    GO BOTTOM
ENDIF
REPLACE art.qtr WITH UPPER(mSeason)
REPLACE art.MONTH WITH VAL(mmonth)
REPLACE art.cl WITH mCLArt
REPLACE art.no2 WITH mNO2Art
REPLACE art.no3 WITH mNO3Art
REPLACE art.SO4 WITH mSO4Art
REPLACE art.nh4 WITH mNH4Art
REPLACE art.so2 WITH mSO2Art
REPLACE art.o1 WITH mO1Art
REPLACE art.o2 WITH mO2Art
REPLACE art.o3 WITH mO3Art
REPLACE art.o4 WITH mO4Art
REPLACE art.op WITH mOPArt
REPLACE art.e1 WITH mE1Art
REPLACE art.e2 WITH mE2Art
REPLACE art.e3 WITH mE3Art
REPLACE art.dcl WITH mCLStd
REPLACE art.dno2 WITH mNO2Std
REPLACE art.dno3 WITH mNO3Std
REPLACE art.dso4 WITH mSO4Std
REPLACE art.dnh4 WITH mNH4Std
REPLACE art.dSO2 WITH mso2Std
REPLACE art.do1 WITH mO1Std
REPLACE art.do2 WITH mO2Std

```

```

REPLACE art.do3 WITH mO3Std
REPLACE art.do4 WITH mO4Std
REPLACE art.dop WITH mOPStd
REPLACE art.de1 WITH mE1Std
REPLACE art.de2 WITH mE2Std
REPLACE art.de3 WITH mE3Std

IF mSeason="B00"      &&Store 25mm artifact
  LOCATE FOR art.qtr=UPPER(mSeason) AND art.MONTH=VAL(mmonth+"2")
  IF EOF("art")
    APPEND BLANK
    GO BOTTOM
  ENDIF
  REPLACE art.qtr WITH UPPER(mSeason)
  REPLACE art.MONTH WITH VAL(mmonth+"2")
  REPLACE art.cl WITH mCLArt25
  REPLACE art.no2 WITH mNO2Art25
  REPLACE art.no3 WITH mNO3Art25
  REPLACE art.SO4 WITH mSO4Art25
  REPLACE art.nh4 WITH mNH4Art25
  REPLACE art.so2 WITH mSO2Art
  REPLACE art.o1 WITH mO1Art
  REPLACE art.o2 WITH mO2Art
  REPLACE art.o3 WITH mO3Art
  REPLACE art.o4 WITH mO4Art
  REPLACE art.op WITH mOPArt
  REPLACE art.e1 WITH mE1Art
  REPLACE art.e2 WITH mE2Art
  REPLACE art.e3 WITH mE3Art
  REPLACE art.dcl WITH mCLStd25
  REPLACE art.dno2 WITH mNO2Std25
  REPLACE art.dno3 WITH mNO3Std25
  REPLACE art.dso4 WITH mSO4Std25
  REPLACE art.dnh4 WITH mNH4Std25
  REPLACE art.dSO2 WITH mso2Std
  REPLACE art.do1 WITH mO1Std
  REPLACE art.do2 WITH mO2Std
  REPLACE art.do3 WITH mO3Std
  REPLACE art.do4 WITH mO4Std
  REPLACE art.dop WITH mOPStd
  REPLACE art.de1 WITH mE1Std
  REPLACE art.de2 WITH mE2Std
  REPLACE art.de3 WITH mE3Std
ENDIF
* Retrieve December 2003 artifacts to use on January 1 & 4, 2004
* for sites that used older nylon filters on those two dates.
IF UPPER(mSeason)="D03" AND VAL(mmonth)=1
  LOCATE FOR art.qtr="D03" AND art.MONTH=12
  IF EOF("art")
    WAIT WINDOW AT 10,40 "No artifacts found for December 2003"
  ENDIF
  mCLoStd = art.dcl
  mNO2oStd = art.dno2
  mNO3oStd = art.dno3
  mSO4oStd = art.dso4
  mNH4oStd = art.dnh4
  mCLoArt = art.cl
  mNO2oArt = art.no2
  mNO3oArt = art.no3
  mSO4oArt = art.SO4
  mNH4oArt = art.nh4
  * mO1oStd = art.d01      &&The December 2003 carbon artifacts are not
  * needed for January 2004
  * mO2oStd = art.d02
  * mO3oStd = art.d03
  * mO4oStd = art.d04
  * mOPoStd = art.dOP
  * mE1oStd = art.de1
  * mE2oStd = art.de2
  * mE3oStd = art.de3
  * mO1oArt = art.O1
  * mO2oArt = art.O2

```

```
* mO3oArt = art.O3
* mO4oArt = art.O4
* mOPoArt = art.OP
* mE1oArt = art.E1
* mE2oArt = art.E2
* mE3oArt = art.E3
ENDIF

SELECT art
USE          && Close Artifacts file
SELECT ions
SET FILTER TO LEFT(STATUS,1) <> "F" AND LEFT(STATUS,1) <> "R"

RETURN
```

C.8. Merge

```
PROCEDURE merge
*****
*
* Merge the data from weights, XRF, laser, ions, and carbon into the output
  database
* Updated August 30, 2004 to add LRNC to the end of the record.
*
* Updated August 4, 2005 to remove DAVI2, MALO1, and MALO2 from processing
*
*****

DIMENSION outdata(184)
STORE -999 TO outdata

*DO SelectSite

SELECT logs
GO TOP
IF mSite=" ALL"
  SET FILTER TO
    wsite = logs.site
ELSE
  SET FILTER TO site = mSite
  GO TOP
  wsite = mSite
ENDIF

mkeytest=0
DO WHILE NOT EOF('logs')

* DO WHILE EOF('sites') OR INLIST(logs.site, "BLNK1 ", "DAVI1 ", "DAVIS ",
  "HANCS ", "INGAS ")
* DO WHILE INLIST(logs.site, "BLNK1", "DAVI1", "DAVIS", "HANCS", "INGAS",
  "DAVI2", "MALO1", "MALO2")
  DO WHILE INLIST(logs.site, "BLNK1", "DAVI1", "DAVIS", "HANCS", "INGAS", ;
    "DAVI2", "FARM1", "FARM2", "FARM3", "FARM4", "FARM5", "FARM6")
    SKIP IN logs
  ENDDO && sites

* Display name of site being processed
  IF wsite <> logs.site
    wsite = logs.site
    mkeytest = LASTKEY()
    WAIT WINDOW AT 10,40 wsite NOWAIT
  ENDIF

  SELECT out
  APPEND BLANK
  SCATTER TO outdata

  DO volumes
  DO MASS
  DO ions
  DO carbon
  DO SO2
  DO elems
  DO laser
  DO composites

* Create output record
  STORE logs.site TO outdata(1)
  outdata(2) = logs.samdat
  outdata(3) = logs.strtim
  outdata(4) = mflowa
  outdata(5) = mflowb
  outdata(6) = mflowc
  outdata(7) = mflowd
  outdata(8) = meta
```

```

outdata(9) = metb
outdata(10) = metc
outdata(11) = metd
outdata(12) = mflaga
outdata(13) = mflagb
outdata(14) = mflagc
outdata(15) = mflagd

outdata(16) = mmf
outdata(17) = mmf_err
outdata(18) = mmf_mdl
outdata(19) = mmt
outdata(20) = mmt_err
outdata(21) = mmt_mdl

outdata(22) = mCL
outdata(23) = mCL_Err
outdata(24) = mCL_MDL
outdata(25) = mNO2
outdata(26) = mNO2_Err
outdata(27) = mNO2_MDL
outdata(28) = mNO3
outdata(29) = mNO3_Err
outdata(30) = mNO3_MDL
outdata(31) = mSO4
outdata(32) = mSO4_Err
outdata(33) = mSO4_MDL
outdata(34) = mNH4
outdata(35) = mNH4_Err
outdata(36) = mNH4_MDL

outdata(37) = mO1
outdata(38) = mO1_ERR
outdata(39) = mO1_MDL
outdata(40) = mO2
outdata(41) = mO2_ERR
outdata(42) = mO2_MDL
outdata(43) = mO3
outdata(44) = mO3_ERR
outdata(45) = mO3_MDL
outdata(46) = mO4
outdata(47) = mO4_ERR
outdata(48) = mO4_MDL
outdata(49) = mOP
outdata(50) = mOP_ERR
outdata(51) = mOP_MDL
outdata(52) = mE1
outdata(53) = mE1_ERR
outdata(54) = mE1_MDL
outdata(55) = mE2
outdata(56) = mE2_ERR
outdata(57) = mE2_MDL
outdata(58) = mE3
outdata(59) = mE3_ERR
outdata(60) = mE3_MDL

outdata(61) = mSO2
outdata(62) = mSO2_err
outdata(63) = mSO2_mdl

FOR N = 1 TO 75
    outdata(63+N) = elements(N)
ENDFOR

outdata(139) = mlrnc
outdata(140) = mlrnc_err
outdata(141) = mlrnc_mdl
outdata(142) = momh
outdata(143) = momh_err
outdata(144) = momh_mdl
outdata(145) = momcn
outdata(146) = momcn_err

```

```

outdata(147) = momcn_mdl
outdata(148) = mlacn
outdata(149) = mlacn_err
outdata(150) = mlacn_mdl

outdata(151) = msoil
outdata(152) = msoil_err
outdata(153) = msoil_mdl
outdata(154) = mnhso
outdata(155) = mnhso_err
outdata(156) = mnhso_mdl
outdata(157) = mnhno
outdata(158) = mnhno_err
outdata(159) = mnhno_mdl
outdata(160) = mknon
outdata(161) = mknon_err
outdata(162) = mknon_mdl
outdata(163) = mrcma
outdata(164) = mrcma_err
outdata(165) = mrcma_mdl
outdata(166) = mrcmc
outdata(167) = mrcmc_err
outdata(168) = mrcmc_mdl
outdata(169) = mrcmi
outdata(170) = mrcmi_err
outdata(171) = mrcmi_mdl
outdata(172) = ms3
outdata(173) = ms3_err
outdata(174) = ms3_mdl
outdata(175) = momhc
outdata(176) = momhc_err
outdata(177) = momhc_mdl
outdata(178) = mrcnh
outdata(179) = mrcnh_err
outdata(180) = mrcnh_mdl
outdata(181) = logs.comments
IF VAL(mYearMonth) > 200412
    outdata(182) = mOPT
    outdata(183) = mOPT_err
    outdata(184) = mOPT_mdl
ENDIF
SELECT out
GATHER FROM outdata

SELECT logs

SKIP
ENDDO && logs
RETURN

```

C.9. Volumes

```
*****
*
* Modified 8/19/2003 to correct error in processing status flags
* The problem occurred if FlagA was "NM", "QD", "LF", or "RF" and
* FlagB, FlagC, or FlagD was not. The correct FlagB, C, or D would
* be overwritten, usually with "RF" because the flow rate was 0
* The error was corrected for 12/2002 data.
*
* Modified 1/20/2005 to set flags to PO if ET<18 hours. This should be
* dealt with more comprehensively when we examine flags.
*
* Modified 7/6/2005 by Lowell to use flashcard flow measurements
* starting with C04. The code uses a file of mean flow transducer
* measurements created separately to replace the flow values in logs.
* It does not use the flashcard ET information, but retains the ET
* from logs. The code also finds the flashcard data from Channel E
* to use in the correct Module X channel.
*
* Modified 7/11/2005 by Lowell to add ability to use Vac flow rates. If
* usela, uselb, or uselc are set to "V", the flow for that module will
* be calculated using the Vac readings
*
* Modified 8/3/2005 by Lowell to correct coding error. The error resulted
* in assigning an incorrect TempC for calculating flow rate on the A
* channel (rarely on the B or C). The TempC variable was assigned before
* repositioning the pointer to the correct spot in the calibration file.
* It is now assigned correctly after filtering the file and positioning
* the record pointer at the bottom.
*
* Modified 8/9/2005 by Lowell to correct error in handling temperature.
* The error occurred when useld<>"T". The program would then replace
* logs.tempcur with flash.TempC. The code should have tested uselt, not
* useld.
*
* Modified 8/10/2005 by Lowell to correct another error associated with
* incorporating FC data into logs. The SITEX relation between LOGS and
* FLASH must use only the first four characters of the site name. All
* other names should use all five characters, particularly SITE9 and SITE5
*
* Modified 9/05/05 to calculate D module flow rate using MAG for Version I
* sampler. Looks in LOGS file for version to decide.
*
* Modified 9/14/05 to correct error in linking to MAG and VAC when using
* "N" flags. Had to add "sitecode=logs.site" to filter condition.
*
* Modified 10/10/05 to add "AND NOT DELETED()" to filter condition on
* all calibration files. Calibrations may stay in the cal files marked
* for deletion, but now they are not used.
*
* Modified 10/12/2005 by Lowell to remove date dependence of temperature
* correction on flow calculation.
*
* Modified 10/12/2005 by Lowell to insert A, B, and C flow flags from
* Nicole's file into temporary logs file for processing. NOTE: THIS WILL
* NEED TO BE ADDRESSED IN FUTURE DATA DELIVERIES
*
* Modified 10/13/2005 by Lowell to replace code to create my flow flags
* prior to using Nicole's flow flags. Also reset limits to match Nicole's
* flow limits. There were some quarters with no flow flags in Nicole's file.
*
* Modified 10/14/2005 by Nicole to correct modifications on 10/13.
*
* Modified 12/6/2005 by Lowell to add 0.00001 to the mag value applied to
* the LOG10 calculation. When the Mag values are zero, this sets a non-zero
* lower limit that still results in a zero flow. It also does not affect
* the actual flow rate in any noticeable way.
*
* Modified 12/6/2005 to add " AND NOT EOF('flash')" to the CASE statement
```

```

* when replacing logs.mag readings with flash readings. This allows the
* logs file to be the default in case there is no flash data.
*****

```

```

PROCEDURE volumes

```

```

PUBLIC mFLOWA, mFLOWB, mFLOWC, mFLOWD
PUBLIC mETA, mETB, mETC, mETD
PUBLIC mVOLA, mVOLB, mVOLC, mVOLD
PUBLIC mAreaA, mAreaB, mAreaC, mAreaD
PUBLIC mCONVA, mCONVB, mCONVC, mCONVD
PUBLIC mFlagA, mFlagB, mFlagC, mFlagD
PUBLIC mBDiam, mInlet

```

```

* Set flag to "PO" if ET<18
IF logs.aflet < 18 AND logs.af1nm ="NM"
  REPLACE logs.af1nm WITH "PO"
ENDIF
IF logs.bflet < 18 AND logs.bf1nm ="NM"
  REPLACE logs.bf1nm WITH "PO"
ENDIF
IF logs.cflet < 18 AND logs.cf1nm ="NM"
  REPLACE logs.cf1nm WITH "PO"
ENDIF
IF logs.dflet < 18 AND logs.df1nm ="NM"
  REPLACE logs.df1nm WITH "PO"
ENDIF

```

```

*****
* If 1/1/2005 or later, use ET and flow information from the      *
* Flashcard Database program                                     *
*****

```

```

*****
* If C04 or later, replace log flow readings with flashcard averages *
*   But don't make the replacement if the logs flag is set      *
*****
*IF logs.site="SAFO1"
*  SUSPEND
*ENDIF
IF logs.samdat>{9/1/2004}
  IF RIGHT(logs.site,1)="X"      && Insert Channel E flash data into X-mod
    channel
      SELECT flash
      SET ORDER TO site4date
      SELECT logs
      SET RELATION OFF INTO flash
      SET RELATION TO LEFT(logs.site,4) + DTOC(logs.samdat) INTO flash ADDITIVE
      xmod = sites.config
      IF xmod="A"
        DO CASE
          CASE UPPER(logs.usela)="N"
            SELECT mag
            SET FILTER TO sitecode=logs.site AND channel="A1" AND
DATE<=logs.samdat AND NOT DELETED()
            GO BOTTOM
            REPLACE logs.af1mi WITH mag.nom_flow
            REPLACE logs.af1mf WITH mag.nom_flow
            SELECT vac
            SET FILTER TO sitecode=logs.site AND channel="A1" AND
DATE<=logs.samdat AND NOT DELETED()
            GO BOTTOM
            REPLACE logs.af1gi WITH vac.nom_flow
            REPLACE logs.af1gf WITH vac.nom_flow
          CASE INLIST(UPPER(logs.usela),"F", " ", "V") AND NOT
EOF('flash')
            REPLACE logs.af1mi WITH flash.emagrsd
            REPLACE logs.af1mf WITH flash.emagrsd
            REPLACE logs.af1gi WITH flash.evacrds
            REPLACE logs.af1gf WITH flash.evacrds
        ENDCASE
      ENDIF
    ENDIF
  ENDIF

```



```

ENDIF
IF xmod="B"
  DO CASE
    CASE UPPER(logs.uselb)="N"
      SELECT mag
      SET FILTER TO sitecode=logs.site AND channel="B1" AND
DATE<=logs.samdat AND NOT DELETED()
      GO BOTTOM
      REPLACE logs.bf1mi WITH mag.nom_flow
      REPLACE logs.bf1mf WITH mag.nom_flow
      SELECT vac
      SET FILTER TO sitecode=logs.site AND channel="B1" AND
DATE<=logs.samdat AND NOT DELETED()
      GO BOTTOM
      REPLACE logs.bf1gi WITH vac.nom_flow
      REPLACE logs.bf1gf WITH vac.nom_flow
    CASE INLIST(UPPER(logs.uselb),"F", " ", "V") AND NOT
EOF('flash')
      REPLACE logs.bf1mi WITH flash.emagrds
      REPLACE logs.bf1mf WITH flash.emagrds
      REPLACE logs.bf1gi WITH flash.evacrds
      REPLACE logs.bf1gf WITH flash.evacrds
  ENDCASE
ENDIF
IF xmod="C"
  DO CASE
    CASE UPPER(logs.uselc)="N"
      SELECT mag
      SET FILTER TO sitecode=logs.site AND channel="C1" AND
DATE<=logs.samdat AND NOT DELETED()
      GO BOTTOM
      REPLACE logs.cf1mi WITH mag.nom_flow
      REPLACE logs.cf1mf WITH mag.nom_flow
      SELECT vac
      SET FILTER TO sitecode=logs.site AND channel="C1" AND
DATE<=logs.samdat AND NOT DELETED()
      GO BOTTOM
      REPLACE logs.cf1gi WITH vac.nom_flow
      REPLACE logs.cf1gf WITH vac.nom_flow
    CASE INLIST(UPPER(logs.uselc),"F", " ", "V") AND NOT
EOF('flash')
      REPLACE logs.cf1mi WITH flash.emagrds
      REPLACE logs.cf1mf WITH flash.emagrds
      REPLACE logs.cf1gi WITH flash.evacrds
      REPLACE logs.cf1gf WITH flash.evacrds
  ENDCASE
ENDIF
IF xmod="D"
  DO CASE
    CASE UPPER(logs.useld)="N"
      SELECT vac
      SET FILTER TO sitecode=logs.site AND channel="D1" AND
DATE<=logs.samdat AND NOT DELETED()
      GO BOTTOM
      REPLACE logs.df1gi WITH vac.nom_flow
      REPLACE logs.df1gf WITH vac.nom_flow
    CASE INLIST(UPPER(logs.useld),"F", " ", "V") AND NOT
EOF('flash')
      REPLACE logs.df1gi WITH flash.evacrds
      REPLACE logs.df1gf WITH flash.evacrds
  ENDCASE
ENDIF
DO CASE
  CASE UPPER(logs.uselt)="N"
    SELECT mag
    SET FILTER TO sitecode=logs.site AND channel="A1" AND
DATE<=logs.samdat AND NOT DELETED()
    GO BOTTOM
    REPLACE logs.tempcur WITH mag.tempc
  CASE UPPER(logs.uselt)<>"T"
    REPLACE logs.tempcur WITH flash.tempc
  ENDCASE

```

```

ELSE          && Not an X-mod, so insert all flash data into logs
  SELECT flash
  SET ORDER TO sitedate
  SELECT logs
  SET RELATION OFF INTO flash
  SET RELATION TO logs.site + DTOC(logs.samdat) INTO flash ADDITIVE
  DO CASE
    CASE UPPER(logs.usela)="N"
      SELECT mag
      SET FILTER TO sitecode=logs.site AND channel="A1" AND
DATE<=logs.samdat AND NOT DELETED()
      GO BOTTOM
      REPLACE logs.af1mi WITH mag.nom_flow
      REPLACE logs.af1mf WITH mag.nom_flow
      SELECT vac
      SET FILTER TO sitecode=logs.site AND channel="A1" AND
DATE<=logs.samdat AND NOT DELETED()
      GO BOTTOM
      REPLACE logs.af1gi WITH vac.nom_flow
      REPLACE logs.af1gf WITH vac.nom_flow
      CASE INLIST(UPPER(logs.usela),"F", " ", "V") AND NOT EOF('flash')
      REPLACE logs.af1mi WITH flash.amagrsd
      REPLACE logs.af1mf WITH flash.amagrsd
      REPLACE logs.af1gi WITH flash.avacrsd
      REPLACE logs.af1gf WITH flash.avacrsd
    ENDCASE
  DO CASE
    CASE UPPER(logs.uselb)="N"
      SELECT mag
      SET FILTER TO sitecode=logs.site AND channel="B1" AND
DATE<=logs.samdat AND NOT DELETED()
      GO BOTTOM
      REPLACE logs.bf1mi WITH mag.nom_flow
      REPLACE logs.bf1mf WITH mag.nom_flow
      SELECT vac
      SET FILTER TO sitecode=logs.site AND channel="B1" AND
DATE<=logs.samdat AND NOT DELETED()
      GO BOTTOM
      REPLACE logs.bf1gi WITH vac.nom_flow
      REPLACE logs.bf1gf WITH vac.nom_flow
      CASE INLIST(UPPER(logs.uselb),"F", " ", "V") AND NOT EOF('flash')
      REPLACE logs.bf1mi WITH flash.bmagrsd
      REPLACE logs.bf1mf WITH flash.bmagrsd
      REPLACE logs.bf1gi WITH flash.bvacrsd
      REPLACE logs.bf1gf WITH flash.bvacrsd
    ENDCASE
  DO CASE
    CASE UPPER(logs.uselc)="N"
      SELECT mag
      SET FILTER TO sitecode=logs.site AND channel="C1" AND
DATE<=logs.samdat AND NOT DELETED()
      * SELECT logs
      * SELECT mag
      GO BOTTOM
      REPLACE logs.cf1mi WITH mag.nom_flow
      REPLACE logs.cf1mf WITH mag.nom_flow
      SELECT vac
      SET FILTER TO sitecode=logs.site AND channel="C1" AND
DATE<=logs.samdat AND NOT DELETED()
      * SELECT logs
      * SELECT vac
      GO BOTTOM
      REPLACE logs.cf1gi WITH vac.nom_flow
      REPLACE logs.cf1gf WITH vac.nom_flow
      CASE INLIST(UPPER(logs.uselc),"F", " ", "V") AND NOT EOF('flash')
      REPLACE logs.cf1mi WITH flash.cmagrsd
      REPLACE logs.cf1mf WITH flash.cmagrsd
      REPLACE logs.cf1gi WITH flash.cvacrsd
      REPLACE logs.cf1gf WITH flash.cvacrsd
    ENDCASE
  DO CASE
    CASE UPPER(logs.useld)="N"

```

```

                SELECT vac
                SET FILTER TO sitecode=logs.site AND channel="D1" AND
DATE<=logs.samdat AND NOT DELETED()
                GO BOTTOM
                REPLACE logs.dflgi WITH vac.nom_flow
                REPLACE logs.dflgf WITH vac.nom_flow
                CASE INLIST(UPPER(logs.useld),"F", " ") AND NOT EOF('flash')
                REPLACE logs.dflgi WITH flash.dvacrsd
                REPLACE logs.dflgf WITH flash.dvacrsd
            ENDCASE
        DO CASE
            CASE UPPER(logs.uselt)="N"
                SELECT mag
                SET FILTER TO sitecode=logs.site AND channel="A1" AND
DATE<=logs.samdat AND NOT DELETED()
                GO BOTTOM
                REPLACE logs.tempcur WITH mag.tempc
            CASE UPPER(logs.uselt)<>"T"
                REPLACE logs.tempcur WITH flash.tempc
        ENDCASE
    ENDIF
ENDIF
*
* Done with replacing logs flows with flashcard flows. Now to calculate actual
  flows
*

*Module A *****
IF RIGHT(logs.site,1)="9"
    SELECT vac
    SET FILTER TO sitecode=logs.site AND channel="A1" AND DATE <= logs.samdat AND
    NOT DELETED()
    GO BOTTOM
    * mTempc = IIF(logs.samdat>={^2004-01-01}, 293., vac.tempc+273.)
    mTempc = 293.    && Changed 12/12/2005 to properly calculate flow rates
    mFLOWA = IIF(INLIST(logs.af1nm, &ValidStat),;
        ROUND((vac.A_Davis+vac.B_Davis*(logs.af1gi+logs.af1gf)/2+0.00001)*;
        vac.Elevfact*;
        Sqrt((logs.tempcur+273.)/(mTempc)),1),0)
    mETA = logs.af1et
    mAreaA = 3.53
    *
    * For SITE9, set a flow flag based on calculated flow
    *
    IF INLIST(logs.af1nm, &NMLFRF)
        DO CASE
            CASE vac.inlet = 'WED' .AND.;
                ( BETWEEN(mFLOWA,15.0,17.0) .OR. ;
                BETWEEN(mFLOWA,21.0,23.0) ).AND. ;
                .NOT. INLIST(logs.site,'BLNK1')
                mFlagA = 'LF'
            CASE vac.inlet = 'WED' .AND. ;
                ( mFLOWA < 15.0 .OR. ;
                mFLOWA > 23.0) .AND. ;
                .NOT. INLIST(logs.site,'BLNK1')
                mFlagA = 'RF'
            CASE vac.inlet <> 'WED' .AND.;
                ( BETWEEN(mFLOWA ,13.0,15.0) .OR. ;
                BETWEEN(mFLOWA ,19.0,21.0) ).AND. ;
                .NOT. INLIST(logs.site,'BLNK1')
                mFlagA = 'LF'
            CASE vac.inlet <> 'WED' .AND. ;
                ( mFLOWA < 13.0 .OR. ;
                mFLOWA > 21.0) .AND. ;
                .NOT. INLIST(logs.site,'BLNK1')
                mFlagA = 'RF'
            OTHERWISE
                mFlagA = logs.af1nm
        ENDCASE
    ELSE
        mFlagA = logs.af1nm
    ENDIF

```

```

mVOLA = IIF(INLIST(logs.af1nm, &ValidStat),;
  mFLOWA*mETA*60/1000,0)
mCONVA = IIF(INLIST(logs.af1nm, &ValidStat),;
  mVOLA/mAreaA,3)
mInlet = vac.inlet
ELSE  && Not a Site 9
  SELECT mag&& Calculate mag flow rate
  SET FILTER TO sitecode=logs.site AND channel="A1" AND DATE <= logs.samdat AND
  NOT DELETED()
  GO BOTTOM
* mTempc = IIF(logs.samdat>={^2004-01-01}, 293., mag.tempc+273.)
mTempc = 293.  && Changed 12/12/2005 to properly calculate flow rates
mFLOWA = IIF(INLIST(logs.af1nm, &ValidStat),;

  ROUND(10^(mag.A_Davis+mag.B_Davis*LOG10((logs.af1mi+logs.af1mf)/2+0.00001)
) *;
  mag.Elevfact*;
  Sqrt((logs.tempcur+273.)/(mTempc)),1),0)
IF logs.samdat>{9/1/2004}
  IF logs.usela = "V" && Calculate vac flow rate
    SELECT vac
    SET FILTER TO sitecode=logs.site AND channel="A1" AND DATE <=
logs.samdat AND NOT DELETED()
    GO BOTTOM
    mFLOWA = IIF(INLIST(logs.af1nm, &ValidStat),;

      ROUND((vac.A_Davis+vac.B_Davis*(logs.af1gi+logs.af1gf)/2+0.00001)*;
      vac.Elevfact*;
      Sqrt((logs.tempcur+273.)/(mTempc)),1),0)

    ENDIF
  ENDIF
mETA = logs.af1et
mAreaA = logs.area
* IF INLIST(logs.af1nm, &ValidStat)
IF INLIST(logs.af1nm, &NMLFRF)
  DO CASE
    CASE mFLOWA < 15          && Added by Nicole 10/14/05
      mFlagA = 'CL'
    CASE mFLOWA < 18          && Added by Nicole 10/14/05
      mFlagA = 'CG'
    CASE (BETWEEN(mFLOWA,19.0,21.3) .OR. BETWEEN(mFLOWA,24.3,27.0));
      .AND. .NOT. INLIST(logs.site,'BLNK1')
      mFlagA = 'LF'
    CASE (mFLOWA < 19.0 .OR. mFLOWA > 27.0);
      .AND. .NOT. INLIST(logs.site,'BLNK1')
      mFlagA = 'RF'
    OTHERWISE
      mFlagA = logs.af1nm
  ENDCASE
ELSE
  mFlagA = logs.af1nm
ENDIF

* Apply flow flags from Nicole's file for years 2000-2004
SELECT logs
IF BETWEEN(logs.samdat, {1/1/2000}, {12/31/2004})
  IF NOT EOF("FlowFlags")
    DO CASE
      CASE mFlagA = "CG" AND FlowFlags.a_flag = "CL"
        mFlagA = FlowFlags.a_flag
      CASE INLIST(mFlagA,"NM", "QD", "UN", "SW", "SP", "AA", "LF",
"RF", "NR");
        AND FlowFlags.a_flag <> "NM"
        mFlagA = FlowFlags.a_flag
    ENDCASE
  ENDIF
ENDIF
* mFlagA = logs.af1nm          &&deleted by Nicole 10/14/05

mVOLA = IIF(INLIST(logs.af1nm, &ValidStat),;
  mFLOWA*mETA*60/1000,0)

```

```

mCONVA = IIF(INLIST(logs.af1nm, &ValidStat),;
mVOLA/mAreaA,3)
ENDIF
*Module B *****
IF RIGHT(logs.site,1)="9"
SELECT vac
SET FILTER TO sitecode=logs.site AND channel="B1" AND DATE <= logs.samdat AND
NOT DELETED()
GO BOTTOM
* mTempc = IIF(logs.samdat>={^2004-01-01}, 293., vac.tempc+273.)
mTempc = 293.    && Changed 12/12/2005 to properly calculate flow rates
mFLOWB = IIF(INLIST(logs.bf1nm, &ValidStat),;
ROUND((vac.A_Davis+vac.B_Davis*(logs.bf1gi+logs.bf1gf)/2+0.00001)*;
vac.Elevfact*;
SQRT((logs.tempcur+273.)/(mTempc)),1),0)
mETB = logs.bf1et
mAreaB = 3.53
* IF INLIST(logs.af1nm, &ValidStat)
IF INLIST(logs.bf1nm, &NMLFRF)
DO CASE
CASE vac.inlet = 'WED' .AND. ;
( BETWEEN(mFLOWB,15.0,17.0) .OR. ;
BETWEEN(mFLOWB,21.0,23.0) ).AND. ;
.NOT. INLIST(logs.site,'BLNK1')
mFlagB = 'LF'
CASE vac.inlet = 'WED' .AND. ;
( mFLOWB < 15.0 .OR. ;
mFLOWB > 23.0 ) .AND. ;
.NOT. INLIST(logs.site,'BLNK1')
mFlagB = 'RF'
CASE vac.inlet <> 'WED' .AND. ;
( BETWEEN(mFLOWB ,13.0,15.0) .OR. ;
BETWEEN(mFLOWB ,19.0,21.0) ).AND. ;
.NOT. INLIST(logs.site,'BLNK1')
mFlagB = 'LF'
CASE vac.inlet <> 'WED' .AND. ;
( mFLOWB < 13.0 .OR. ;
mFLOWB > 21.0 ) .AND. ;
.NOT. INLIST(logs.site,'BLNK1')
mFlagB = 'RF'
OTHERWISE
mFlagB = logs.bf1nm
ENDCASE
ELSE
mFlagB = logs.bf1nm
ENDIF

mVOLB = IIF(INLIST(logs.bf1nm, &ValidStat),;
mFLOWB*mETB*60/1000,0)
***** What is mCONVB used for?
mCONVB = IIF(INLIST(logs.bf1nm, &ValidStat),;
mVOLB/mAreaB,3)
mInlet = vac.inlet
ELSE
SELECT mag&& Calculate mag flow rate
SET FILTER TO sitecode=logs.site AND channel="B1" AND DATE <= logs.samdat AND
NOT DELETED()
GO BOTTOM
* mTempc = IIF(logs.samdat>={^2004-01-01}, 293., mag.tempc+273.)
mTempc = 293.    && Changed 12/12/2005 to properly calculate flow rates
mFLOWB = IIF(INLIST(logs.bf1nm, &ValidStat),;

ROUND(10^(mag.A_Davis+mag.B_Davis*LOG10((logs.bf1mi+logs.bf1mf)/2+0.00001)
)*;
mag.Elevfact*;
SQRT((logs.tempcur+273.)/(mTempc)),1),0)
IF logs.samdat>{9/1/2004}
IF logs.use1b = "V" && Calculate vac flow rate
SELECT vac
SET FILTER TO sitecode=logs.site AND channel="B1" AND DATE <=
logs.samdat AND NOT DELETED()
GO BOTTOM

```

```

        mFLOWB = IIF(INLIST(logs.bflnm, &ValidStat),;

ROUND((vac.A_Davis+vac.B_Davis*(logs.bflgi+logs.bflgf)/2+0.00001)*;
      vac.Elevfact*;
      Sqrt((logs.tempcur+273.)/(mTempc)),1),0)

    ENDIF
ENDIF
mETB = logs.bflet
* mAreaB = 3.53
mAreaB = 6.15    && 37mm diameter filter - 28mm diameter deposit
mBDiam = IIF(logs.samdat<{10/1/1999}, "25", MAX("25", logs.bdiam))
* IF INLIST(logs.bflnm, &ValidStat)
IF INLIST(logs.bflnm, &NMLFRF)
  DO CASE
    CASE mFLOWB < 15          && Added by Nicole 10/14/05
      mFlagB = 'CL'
    CASE mFLOWB < 18          && Added by Nicole 10/14/05
      mFlagB = 'CG'
    CASE (BETWEEN(mFLOWB,19.0,21.3) .OR. BETWEEN(mFLOWB,24.3,27.0));
      .AND. .NOT. INLIST(logs.site,'BLNK1')
      mFlagB = 'LF'
    CASE (mFLOWB < 19.0 .OR. mFLOWB > 27.0);
      .AND. .NOT. INLIST(logs.site,'BLNK1')
      mFlagB = 'RF'
    OTHERWISE
      mFlagB = logs.bflnm
  ENDCASE
ELSE
  mFlagB = logs.bflnm
ENDIF

* Apply flow flags from Nicole's file for years 2000-2004
SELECT logs
IF BETWEEN(logs.samdat, {1/1/2000}, {12/31/2004})
  IF NOT EOF("FlowFlags")
    DO CASE
      CASE mFlagB = "CG" AND FlowFlags.b_flag = "CL"
        mFlagB = FlowFlags.b_flag
      CASE INLIST(mFlagB,"NM", "QD", "UN", "SW", "SP", "AA", "LF",
"RF", "NR");
        AND FlowFlags.b_flag <> "NM"
        mFlagB = FlowFlags.b_flag
    ENDCASE
  ENDIF
ENDIF
* mFlagB = logs.bflnm          &&deleted by Nicole 10/14/05

mVOLB = IIF(INLIST(logs.bflnm, &ValidStat),;
mFLOWB*mETB*60/1000,0)
mCONVB = IIF(INLIST(logs.bflnm, &ValidStat),;
mVOLB/mAreaB,3)
ENDIF
*Module C *****
IF RIGHT(logs.site,1)="9"
  SELECT vac
  SET FILTER TO sitecode=logs.site AND channel="C1" AND DATE <= logs.samdat AND
  NOT DELETED()
  GO BOTTOM
* mTempc = IIF(logs.samdat>={^2004-01-01}, 293., vac.tempc+273.)
mTempc = 293.    && Changed 12/12/2005 to properly calculate flow rates
mFLOWC = IIF(INLIST(logs.cflnm, &ValidStat),;
  ROUND((vac.A_Davis+vac.B_Davis*(logs.cflgi+logs.cflgf)/2+0.00001)*;
  vac.Elevfact*;
  Sqrt((logs.tempcur+273.)/(mTempc)),1),0)
mETC = logs.cflet
mAreaC = 3.53
* IF INLIST(logs.af1nm, &ValidStat)
IF INLIST(logs.cflnm, &NMLFRF)
  DO CASE
    CASE vac.inlet = 'WED' .AND.;
      ( BETWEEN(mFLOWC,15.0,17.0) .OR. ;
        BETWEEN(mFLOWC,21.0,23.0) ).AND. ;

```

```

        .NOT. INLIST(logs.site,'BLNK1')
        mFlagC = 'LF'
CASE vac.inlet = 'WED' .AND. ;
      ( mFLOWC < 15.0 .OR. ;
        mFLOWC > 23.0 ) .AND. ;
      .NOT. INLIST(logs.site,'BLNK1')
        mFlagC = 'RF'
CASE vac.inlet <> 'WED' .AND. ;
      ( BETWEEN(mFLOWC ,13.0,15.0) .OR. ;
        BETWEEN(mFLOWC ,19.0,21.0) ) .AND. ;
      .NOT. INLIST(logs.site,'BLNK1')
        mFlagC = 'LF'
CASE vac.inlet <> 'WED' .AND. ;
      ( mFLOWC < 13.0 .OR. ;
        mFLOWC > 21.0 ) .AND. ;
      .NOT. INLIST(logs.site,'BLNK1')
        mFlagC = 'RF'
OTHERWISE
        mFlagC = logs.bflnm
      ENDCASE
ELSE
        mFlagC = logs.bflnm
ENDIF

mVOLC = IIF(INLIST(logs.cflnm, &ValidStat),;
            mFLOWC*mETC*60/1000,0)
mCONVC = IIF(INLIST(logs.cflnm, &ValidStat),;
            mVOLC/mAreaC,3)
mInlet = vac.inlet
ELSE
SELECT mag&& Calculate mag flow rate
SET FILTER TO sitecode=logs.site AND channel="C1" AND DATE <= logs.samdat AND
NOT DELETED()
GO BOTTOM
* mTempc = IIF(logs.samdat>={^2004-01-01}, 293., mag.tempc+273.)
mTempc = 293.    && Changed 12/12/2005 to properly calculate flow rates
mFLOWC = IIF(INLIST(logs.cflnm, &ValidStat),;

      ROUND(10^(mag.A_Davis+mag.B_Davis*LOG10((logs.cflmi+logs.cflmf)/2+0.00001)
) *;
      mag.Elevfact*;
      SQRT((logs.tempcur+273.)/(mTempc)),1),0)
IF logs.samdat>{9/1/2004}
  IF logs.uselc = "V" && Calculate vac flow rate
    SELECT vac
    SET FILTER TO sitecode=logs.site AND channel="C1" AND DATE <=
logs.samdat AND NOT DELETED()
    GO BOTTOM
    mFLOWC = IIF(INLIST(logs.cflnm, &ValidStat),;

      ROUND((vac.A_Davis+vac.B_Davis*(logs.cflgi+logs.cflgf)/2+0.00001)*;
      vac.Elevfact*;
      SQRT((logs.tempcur+273.)/(mTempc)),1),0)

    ENDIF
  ENDIF
mETC = logs.cflet
mAreaC = 3.53
* IF INLIST(logs.af1nm, &ValidStat)
* IF INLIST(logs.af1nm,'NM','QD','LF','RF') AND
INLIST(logs.cflnm,'NM','CG','LF','RF')
IF INLIST(logs.cflnm, &NMLFRF)
  DO CASE
    CASE mFLOWC < 15          && Added by Nicole 10/14/05
      mFlagC = 'CL'
    CASE mFLOWC < 18          && Added by Nicole 10/14/05
      mFlagC = 'CG'
    CASE (BETWEEN(mFLOWC,19.0,21.3) .OR. BETWEEN(mFLOWC,24.3,27.0));
      .AND. .NOT. INLIST(logs.site,'BLNK1')
      mFlagC = 'LF'
    CASE (mFLOWC < 19.0 .OR. mFLOWC > 27.0);
      .AND. .NOT. INLIST(logs.site,'BLNK1')
      mFlagC = 'RF'
  
```

```

                OTHERWISE
                    mFlagC = logs.cf1nm
            ENDCASE
        ELSE
            mFlagC = logs.cf1nm
        ENDIF

* Apply flow flags from Nicole's file for years 2000-2004
SELECT logs
IF BETWEEN(logs.samdat, {1/1/2000}, {12/31/2004})
    IF NOT EOF("FlowFlags")
        DO CASE
            CASE mFlagC = "CG" AND FlowFlags.c_flag = "CL"
                mFlagC = FlowFlags.c_flag
            CASE INLIST(mFlagC,"NM", "QD", "UN", "SW", "SP", "AA", "LF",
"RF", "NR");
                AND FlowFlags.c_flag <> "NM"
                mFlagC = FlowFlags.c_flag
            ENDCASE
        ENDIF
    ENDIF
* mFlagC = logs.cf1nm                &&deleted by Nicole 10/14/05

mVOLC = IIF(INLIST(logs.cf1nm, &ValidStat),;
mFLOWC*mETC*60/1000,0)
mCONVC = IIF(INLIST(logs.cf1nm, &ValidStat),;
mVOLC/mAreaC,3)
ENDIF
* Module D *****
IF logs.sampler="VER2"
    SELECT vac
    SET FILTER TO sitecode=logs.site AND channel="D1" AND DATE <= logs.samdat AND
    NOT DELETED()
    GO BOTTOM
* mTempc = IIF(logs.samdat>={^2004-01-01}, 293., vac.tempc+273.)
mTempc = 293.    && Changed 12/12/2005 to properly calculate flow rates
mFLOWD = IIF(INLIST(logs.df1nm, &ValidStat),;
    ROUND((vac.A_Davis+vac.B_Davis*(logs.df1gi+logs.df1gf)/2)*;
    vac.Elevfact*;
    SQRT((logs.tempcur+273.)/(mTempc)),1),0)
ELSE    && VER1 sampler
    SELECT mag&& Calculate mag flow rate
    SET FILTER TO sitecode=logs.site AND channel="D1" AND DATE <= logs.samdat AND
    NOT DELETED()
    GO BOTTOM
    mTempc = mag.tempc+273
    mFLOWD = IIF(INLIST(logs.df1nm, &ValidStat),;
        ROUND(10^(mag.A_Davis+mag.B_Davis*LOG10((logs.df1mi+logs.df1mf)/2+0.00001)
        )*;
        mag.Elevfact*;
        SQRT((logs.tempcur+273.)/(mTempc)),1),0)
ENDIF
mETD = logs.df1et
mAreaD = 3.53
*IF INLIST(logs.bf1nm, &ValidStat)
*IF INLIST(logs.bf1nm, 'NM', 'QD', 'LF', 'RF') AND
    INLIST(logs.df1nm, 'NM', 'CG', 'LF', 'RF')
IF INLIST(logs.df1nm, &NMLFRF)
    DO CASE
        CASE vac.inlet = 'WED' .AND.;
            ( BETWEEN(mFLOWD,15.0,17.0) .OR. ;
            BETWEEN(mFLOWD,21.0,23.0) ).AND. ;
            .NOT. INLIST(logs.site, 'BLNK1')
            mFlagD = 'LF'
        CASE vac.inlet = 'WED' .AND. ;
            ( mFLOWD < 15.0 .OR. ;
            mFLOWD > 23.0) .AND. ;
            .NOT. INLIST(logs.site, 'BLNK1')
            mFlagD = 'RF'
        CASE vac.inlet <> 'WED' .AND.;
            ( BETWEEN(mFLOWD,13.0,15.0) .OR. ;

```



```

        BETWEEN(mFLOWD,19.0,21.0) ).AND. ;
        .NOT. INLIST(logs.site,'BLNK1')
CASE    mFlagD = 'LF'
        vac.inlet <> 'WED' .AND. ;
        (      mFLOWD < 13.0 .OR. ;
          mFLOWD > 21.0) .AND. ;
        .NOT. INLIST(logs.site,'BLNK1')
        mFlagD = 'RF'
OTHERWISE
        mFlagD = logs.df1nm
ENDCASE
ELSE
        mFlagD = logs.df1nm
ENDIF

mVOLD = IIF(INLIST(logs.df1nm, &ValidStat),;
            mFLOWD*mETD*60/1000,0)
mCONVD = IIF(INLIST(logs.df1nm, &ValidStat),;
            mVOLD/mAreaD,3)
mInlet = vac.inlet
RETURN

```

C.10. Mass

PROCEDURE mass

```
*****
* Calculate mass concentration, err, and mdl
*
* Modified 9/22/05 by Lowell to exclude FB records from calculation.
* The LOCATE command stops at the first record that matches the criteria.
* If the FB record precedes the NM record, the mass was being calculated
* from the FB record. Now it selects the correct record
*
*****

PUBLIC mMF, mMF_err, mMF_mdl, mMT, mMT_err, mMT_mdl

SELECT wts

***** PM10 mass *****
LOCATE FOR wts.site=logs.site AND wts.samdat=logs.samdat AND
wts.strtim=logs.strtim AND wts.chan="D1" AND wts.status<>"FB"
If INLIST(mflagd, &ValidStat)
IF .NOT. EOF('wts') .AND. wts.postwght <> 0 .AND. mvold <>0 .AND. not
inlist(wts.status,"UN","XX")
    mMT = 1000*(wts.postwght - wts.prewght)*1000/mvold
    mMT_err = 1000*SQRT((5/mvold)^2 + (mMT/1000*mmfvol)^2)
    mMT_mdl = 1000*10/mvold
ELSE
    mMT = -999
    mMT_err = -999
    mMT_mdl = -999
ENDIF
IF inlist(wts.status,"SW","UN","XX")
    mflagd = wts.status
ENDIF
ELSE
    mMT = -999
    mMT_err = -999
    mMT_mdl = -999
ENDIF

***** Fine mass *****
LOCATE FOR wts.site=logs.site AND wts.samdat=logs.samdat AND
wts.strtim=logs.strtim AND wts.chan="A1" AND wts.status<>"FB"
If INLIST(mflaga, &ValidStat)
IF .NOT. EOF('wts') .AND. wts.postwght <> 0 .AND. mvola <>0 .AND. not
inlist(wts.status,"UN","XX")
    mMF = 1000*(wts.postwght - wts.prewght)*1000/mvola
    mMF_err = 1000*SQRT((5/mvola)^2 + (mMF/1000*mmfvol)^2)
    mMF_mdl = 1000*10/mvola
ELSE
    mMF = -999
    mMF_err = -999
    mMF_mdl = -999
ENDIF
IF inlist(wts.status,"SW","UN","XX")
    mflaga = wts.status
ENDIF
ELSE
    mMF = -999
    mMF_err = -999
    mMF_mdl = -999
ENDIF

RETURN
```

C.11. Ions

```
PROCEDURE ions
* Calculate ions concentrations
*
*****
*
* Modified 11/17/2005 to add analytical mdl for Cl, NO2, NO3, SO4, and NH4
* to conterr.dbf and use them if field blanks stdev is zero.
*
*****

PUBLIC mCl, mCl_Err, mCl_MDL
PUBLIC mNO2, mNO2_Err, mNO2_MDL
PUBLIC mNO3, mNO3_Err, mNO3_MDL
PUBLIC mSO4, mSO4_Err, mSO4_MDL
PUBLIC mNH4, mNH4_Err, mNH4_MDL

IF mSeason="B00"
  DO CASE
    CASE logs.bdiam="37"
      mCLStd = mCLStd37
      mNO2Std = mNO2Std37
      mNO3Std = mNO3Std37
      mSO4Std = mSO4Std37
      mNH4Std = mNH4Std37
      mCLArt = mCLArt37
      mNO2Art = mNO2Art37
      mNO3Art = mNO3Art37
      mSO4Art = mSO4Art37
      mNH4Art = mNH4Art37
    OTHERWISE
      mCLStd = mCLStd25
      mNO2Std = mNO2Std25
      mNO3Std = mNO3Std25
      mSO4Std = mSO4Std25
      mNH4Std = mNH4Std25
      mCLArt = mCLArt25
      mNO2Art = mNO2Art25
      mNO3Art = mNO3Art25
      mSO4Art = mSO4Art25
      mNH4Art = mNH4Art25
  ENDCASE
ENDIF
lCLStd = mCLStd
lNO2Std = mNO2Std
lNO3Std = mNO3Std
lSO4Std = mSO4Std
lNH4Std = mNH4Std
lCLArt = mCLArt
lNO2Art = mNO2Art
lNO3Art = mNO3Art
lSO4Art = mSO4Art
lNH4Art = mNH4Art
* Test for use of old nylon filter lot on 1/1/04 or 1/4/04 for selected sites
IF INLIST(samdat, {^2004-01-01}, {^2004-01-04})
  IF INLIST(site, "ADPI1", "AREN1", "BAND1", "BIBE1", "BIBEX", "BLIS1", "BOAP1",
    "BRCA1", ;
    "BRET1", "BRID1", "BRID9", "BRMA1", "CABA1", "CABI1", "CACO1",
    "CANY1", ;
    "CAPI1", "CEBL1", "CHAS1", "CHER1", "CHIC1", "CHIR1", "CLPE1",
    "COHI1") .OR. ;
    INLIST(site, "CORI1", "CRLA1", "DENA1", "DETR1", "DEVA1", "DOME1",
    "ELDO1", "ELLI1", ;
    "FLAT1", "FOPE1", "GICL1", "GRBA1", "GRSA1", "GUMO1", "HALE1",
    "HAVO1", ;
    "HEGL1", "KAIS1", "KALM1", "LASU1", "MAVI1", "MEVE1", "MEVEX",
    "MKGO1") .OR. ;
    INLIST(site, "MOHO1", "MONT1", "MORA1", "MORA9", "NOAB1", "NOCH1",
    "OLTO1", "ORPI1", ;
```

```

"PASA1", "PEFO1", "PMRF1", "PMRFX", "PORE1", "QUCI1", "QURE1",
"QUREX",;
"REDW1", "SACR1", "SAFO1", "SAFOX", "SAMA1", "SAPE1", "SAWE1",
"SAWT1") .OR. ;
INLIST(site,"SEQU1", "SEQU9", "SIAN1", "SIKE1", "SNPA1", "SYCA1",
"TALL1", "THBA1",;
"THSI1", "TONT1", "TRIN1", "VILA1", "WARI1", "WEMI1", "WHPA1",
"WHRI1",;
"WIMO1", "YELL1", "YOSE1", "ZION1")
lCLStd = mCLoStd
lNO2Std = mNO2oStd
lNO3Std = mNO3oStd
lSO4Std = mSO4oStd
lNH4Std = mNH4oStd
lCLArt = mCLoArt
lNO2Art = mNO2oArt
lNO3Art = mNO3oArt
lSO4Art = mSO4oArt
lNH4Art = mNH4oArt
ELSE
lCLStd = mCLStd
lNO2Std = mNO2Std
lNO3Std = mNO3Std
lSO4Std = mSO4Std
lNH4Std = mNH4Std
lCLArt = mCLArt
lNO2Art = mNO2Art
lNO3Art = mNO3Art
lSO4Art = mSO4Art
lNH4Art = mNH4Art
ENDIF
ENDIF

*BOWA used the old lot for 1/7/04 through 1/19/04 by mistake. 1/7/04 and 1/10/04
were EP samples - not run
IF INLIST(samdat, {^2004-01-13}, {^2004-01-16}, {^2004-01-19}) AND site="BOWA1"
lCLStd = mCLoStd
lNO2Std = mNO2oStd
lNO3Std = mNO3oStd
lSO4Std = mSO4oStd
lNH4Std = mNH4oStd
lCLArt = mCLoArt
lNO2Art = mNO2oArt
lNO3Art = mNO3oArt
lSO4Art = mSO4oArt
lNH4Art = mNH4oArt
ENDIF

SELECT ions
SET ORDER TO sitdatim
SET FILTER TO LEFT(STATUS,1) <> "F" AND LEFT(STATUS,1) <> "R"
SELECT logs
SELECT ions
IF INLIST(mflagb, &ValidStat)
  IF .NOT. EOF('ions') .AND. mvolb <> 0
    IF INLIST(ions.STATUS, &ValidStat)
      IF ions.STATUS="SW"
        mflagb=ions.STATUS
      ENDIF
***** Chloride *****
      mCl = 1000*(ions.cl - lCLArt)/mvolb
      * mCl Err = 1000*SQRT((mMFVOL*mcl/1000)^2 + (mcocl*mcl/1000)^2 +
      (lCLSTD/mvolb)^2)
      mCl_Err = 1000*SQRT(lCLStd^2 + 2*(lCLArt)*(mCOCL^2)*MAX(0,ions.cl-
      lCLArt);
      + mCOCL^2*(ions.cl-lCLArt)^2+ mMFVOL^2*(ions.cl-
      lCLArt)^2)/mvolb
      mCl_MDL = 1000*2*MAX(lCLStd,mdlcl)/mvolb
***** Nitrite *****
      mNO2 = 1000*(ions.NO2 - lNO2Art)/mvolb
      IF mNO2 > 0

```

```

*          mNO2_Err = 1000*SQRT((mMFVOL*mNO2/1000)^2 + (mcoNO2*mNO2/1000)^2 +
(LNO2STD/mvolb)^2)
          mNO2_Err = 1000*SQRT(LNO2Std^2 +
2*(LNO2Art)*(mCONO2^2)*(ions.NO2-LNO2Art);
          + (mCONO2*(ions.NO2-LNO2Art))^2+ (mMFVOL*(ions.NO2-
LNO2Art))^2)/mvolb
          ELSE
            mNO2_Err = 0
          ENDIF
          mNO2_MDL = 1000*2*MAX(LNO2Std,mdlno2)/mvolb

***** Nitrate *****
          mNO3      = 1000*(ions.NO3 - LNO3Art)/mvolb
*          mNO3_Err = 1000*SQRT((mMFVOL*mNO3/1000)^2 + (mcoNO3*mNO3/1000)^2 +
(LNO3STD/mvolb)^2)
          mNO3_Err = 1000*SQRT(LNO3Std^2 + 2*(LNO3Art)*(mCONO3^2)*(ions.NO3-
LNO3Art);
          + mCONO3^2*(ions.NO3-LNO3Art)^2+ mMFVOL^2*(ions.NO3-
LNO3Art)^2)/mvolb
          mNO3_MDL = 1000*2*MAX(LNO3Std,mdlno3)/mvolb

***** Sulfate *****
          mSO4      = 1000*(ions.SO4 - lSO4Art)/mvolb
*          mSO4_Err = 1000*SQRT((mMFVOL*mSO4/1000)^2 + (mcoSO4*mSO4/1000)^2 +
(lSO4STD/mvolb)^2)
          mSO4_Err = 1000*SQRT(lSO4Std^2 + 2*(lSO4Art)*(mCOSO4^2)*(ions.SO4-
lSO4Art);
          + mCOSO4^2*(ions.SO4-lSO4Art)^2+ mMFVOL^2*(ions.SO4-
lSO4Art)^2)/mvolb
          mSO4_MDL = 1000*2*MAX(lSO4Std,mdlso4)/mvolb

***** Ammonium *****
* Must account for records with NH4 values.
          IF EOF('nh4sites')
            REPLACE ions.nh4 WITH -999
          ELSE
            IF logs.samdat<nh4sites.startdate OR
logs.samdat>nh4sites.enddate
              REPLACE ions.nh4 WITH -999
            ENDIF
          ENDIF
          IF ions.nh4=0 AND NOT INLIST(ions.site,
"COHU1", "DOSO1", "EVER1", "GRSM1", "GRSM9", "JARI1", "LIGO1", "MACA1", "ROMA1", "SAMA1
", "SHEN1", "SHRO1", "SWAN1", "MACAX", "SIPS1")
*          REPLACE ions.nh4 WITH -999
*          ENDIF
          IF ions.nh4 >= 0
            mNH4      = 1000*(ions.nh4- lNH4Art)/mvolb
            mNH4_Err = 1000*SQRT((mMFVOL*mNH4/1000)^2 +
(mcoNH4*mNH4/1000)^2 + (lNH4Std/mvolb)^2)
            mNH4_MDL = 1000*2*MAX(lNH4Std,mdl nh4)/mvolb
          ELSE
            mNH4      = -999
            mNH4_Err = -999
            mNH4_MDL = -999
          ENDIF
        ELSE
          mCl      = -999
          mCl_Err  = -999
          mCl_MDL  = -999
          mNO2     = -999
          mNO2_Err = -999
          mNO2_MDL = -999
          mNO3     = -999
          mNO3_Err = -999
          mNO3_MDL = -999
          mSO4     = -999
          mSO4_Err = -999
          mSO4_MDL = -999
          mNH4     = -999
          mNH4_Err = -999
        
```

```

                mNH4_MDL = -999
                mflagb = ions.STATUS
            ENDIF
        ELSE
            mCl      = -999
            mCl_Err  = -999
            mCl_MDL  = -999
            mNO2     = -999
            mNO2_Err = -999
            mNO2_MDL = -999
            mNO3     = -999
            mNO3_Err = -999
            mNO3_MDL = -999
            mSO4     = -999
            mSO4_Err = -999
            mSO4_MDL = -999
            mNH4     = -999
            mNH4_Err = -999
            mNH4_MDL = -999
        ENDIF
    ELSE
        mCl      = -999
        mCl_Err  = -999
        mCl_MDL  = -999
        mNO2     = -999
        mNO2_Err = -999
        mNO2_MDL = -999
        mNO3     = -999
        mNO3_Err = -999
        mNO3_MDL = -999
        mSO4     = -999
        mSO4_Err = -999
        mSO4_MDL = -999
        mNH4     = -999
        mNH4_Err = -999
        mNH4_MDL = -999
    ENDIF
RETURN

```

C.12. Carbon

PROCEDURE carbon

```
* Calculate carbon concentrations
* Modified 1/4/2006 to include new carbon pyrolysis fraction in DRI data.
  Beginning with January 2004
* data, DRI is including OP calculated with both reflectance and transmittance
  levels. Beginning
* with D04, then, the carbon files contain OPR and OPT values.
*
* DRI reports micrograms/filter. This program converts them to nanograms/cubic
  meter of air sampled.
*
```

```
PUBLIC mO1, mO1_Err, mO1_MDL
PUBLIC mO2, mO2_Err, mO2_MDL
PUBLIC mO3, mO3_Err, mO3_MDL
PUBLIC mO4, mO4_Err, mO4_MDL
PUBLIC mOP, mOP_Err, mOP_MDL
PUBLIC mOPT, mOPT_Err, mOPT_MDL
PUBLIC mE1, mE1_Err, mE1_MDL
PUBLIC mE2, mE2_Err, mE2_MDL
PUBLIC mE3, mE3_Err, mE3_MDL
PUBLIC mOC, mOC_Err, mOC_Mdl
PUBLIC mEC, mEC_Err, mEC_Mdl
```

```
IF INLIST(mflagc, &ValidStat)
  IF .NOT. EOF('carbon') .AND. mvolc <>0
    IF carbon.STATUS="SW"
      mflagc=carbon.STATUS
    ENDIF
    mO1 = 1000*(carbon.O1tc - mO1Art)/mvolc
    mO1_Err = 1000*SQRT(mO1Std^2 + 2*(mO1Art)*(mCOO1^2)*(carbon.O1tc-mO1Art);
      + (mCOO1*(carbon.O1tc-mO1Art))^2 + (mMFVOL*(carbon.O1tc-
mO1Art))^2)/mvolc
    mO1_MDL = 1000*2*mO1Std/mvolc

    mO2 = 1000*(carbon.O2tc - mO2Art)/mvolc
    mO2_Err = 1000*SQRT(mO2Std^2 + 2*(mO2Art)*(mCOO2^2)*(carbon.O2tc-mO2Art);
      + (mCOO2*(carbon.O2tc-mO2Art))^2 + (mMFVOL*(carbon.O2tc-
mO2Art))^2)/mvolc
    mO2_MDL = 1000*2*mO2Std/mvolc

    mO3 = 1000*(carbon.O3tc - mO3Art)/mvolc
    mO3_Err = 1000*SQRT(mO3Std^2 + 2*(mO3Art)*(mCOO3^2)*(carbon.O3tc-mO3Art);
      + (mCOO3*(carbon.O3tc-mO3Art))^2 + (mMFVOL*(carbon.O3tc-
mO3Art))^2)/mvolc
    mO3_MDL = 1000*2*mO3Std/mvolc

    mO4 = 1000*(carbon.O4tc - mO4Art)/mvolc
    mO4_Err = 1000*SQRT(mO4Std^2 + 2*(mO4Art)*(mCOO4^2)*(carbon.O4tc-mO4Art);
      + (mCOO4*(carbon.O4tc-mO4Art))^2 + (mMFVOL*(carbon.O4tc-
mO4Art))^2)/mvolc
    mO4_MDL = 1000*2*mO4Std/mvolc

    mE1 = 1000*(carbon.E1tc - mE1Art)/mvolc
    mE1_Err = 1000*SQRT(mE1Std^2 + 2*(mE1Art)*(mCOE1^2)*(carbon.E1tc-mE1Art);
      + (mCOE1*(carbon.E1tc-mE1Art))^2 + (mMFVOL*(carbon.E1tc-
mE1Art))^2)/mvolc
    mE1_MDL = 1000*2*mE1Std/mvolc

    mE2 = 1000*(carbon.E2tc - mE2Art)/mvolc
    mE2_Err = 1000*SQRT(mE2Std^2 + 2*(mE2Art)*(mCOE2^2)*(carbon.E2tc-mE2Art);
      + (mCOE2*(carbon.E2tc-mE2Art))^2 + (mMFVOL*(carbon.E2tc-
mE2Art))^2)/mvolc
    mE2_MDL = 1000*2*mE2Std/mvolc
    mE3 = 1000*(carbon.E3tc - mE3Art)/mvolc
    mE3_Err = 1000*SQRT(mE3Std^2 + 2*(mE3Art)*(mCOE3^2)*(carbon.E3tc-mE3Art);
```

```

+ (mCOE3*(carbon.E3tc-mE3Art))^2 + (mMFVOL*(carbon.E3tc-
mE3Art))^2)/mvolc
mE3_MDL = 1000*2*mE3Std/mvolc

** Starting with D04 use both OPR and OPT data. Prior to D04, use only OP - Not
implemented yet
* IF samdat<{12/1/2004}
mOP = 1000*(carbon.OPtc - mOPArt)/mvolc
mOP_Err = 1000*SQRT(mOPStd^2 + 2*(mOPArt)*(mCOOP^2)*(carbon.OPtc-mOPArt);
+ (mCOOP*(carbon.OPtc-mOPArt))^2 + (mMFVOL*(carbon.OPtc-
mOPArt))^2)/mvolc
mOP_MDL = 1000*2*mOPStd/mvolc
* ELSE
mOP = 1000*(carbon.OPtrc - mOPArt)/mvolc
mOP_Err = 1000*SQRT(mOPStd^2 + 2*(mOPArt)*(mCOOP^2)*(carbon.OPtrc-
mOPArt);
+ (mCOOP*(carbon.OPtrc-mOPArt))^2 + (mMFVOL*(carbon.OPtrc-
mOPArt))^2)/mvolc
mOP_MDL = 1000*2*mOPStd/mvolc
*
mOPT = 1000*(carbon.OPttc - mOPTArt)/mvolc
mOPT_Err = 1000*SQRT(mOPTStd^2 +
2*(mOPTArt)*(mCOOP^2)*(carbon.OPttc-mOPTArt);
+ (mCOOP*(carbon.OPttc-mOPTArt))^2 + (mMFVOL*(carbon.OPttc-
mOPTArt))^2)/mvolc
mOPT_MDL = 1000*2*mOPTStd/mvolc
*
ENDIF

mOC = mO1 + mO2 + mO3 + mO4 + mOP
mOC_Err = SQRT((120^2) + (0.05*mOC)^2)
mEC = mE1 + mE2 + mE3 - mOP
mEC_Err = SQRT((34^2) + (0.07*mEC)^2)
ELSE
mO1 = -999
mO1_Err = -999
mO1_MDL = -999
mO2 = -999
mO2_Err = -999
mO2_MDL = -999
mO3 = -999
mO3_Err = -999
mO3_MDL = -999
mO4 = -999
mO4_Err = -999
mO4_MDL = -999
mOP = -999
mOP_Err = -999
mOP_MDL = -999
mOPT = -999
mOPT_Err = -999
mOPT_MDL = -999
mE1 = -999
mE1_Err = -999
mE1_MDL = -999
mE2 = -999
mE2_Err = -999
mE2_MDL = -999
mE3 = -999
mE3_Err = -999
mE3_MDL = -999
mOC = -999
mOC_Err = -999
mOC_Mdl = -999
mEC = -999
mEC_Err = -999
mEC_Mdl = -999
ENDIF
ELSE
mO1 = -999
mO1_Err = -999
mO1_MDL = -999
mO2 = -999

```



```
mO2_Err = -999
mO2_MDL = -999
mO3      = -999
mO3_Err = -999
mO3_MDL = -999
mO4      = -999
mO4_Err = -999
mO4_MDL = -999
mOP      = -999
mOP_Err = -999
mOP_MDL = -999
mOPT     = -999
mOPT_Err = -999
mOPT_MDL = -999
mE1      = -999
mE1_Err = -999
mE1_MDL = -999
mE2      = -999
mE2_Err = -999
mE2_MDL = -999
mE3      = -999
mE3_Err = -999
mE3_MDL = -999
mOC      = -999
mOC_Err = -999
mOC_Mdl = -999
mEC      = -999
mEC_Err = -999
mEC_Mdl = -999
ENDIF
RETURN
```

C.13. SO₂

```
PROCEDURE SO2
* Calculate SO2 concentrations

PUBLIC mSO2, mSO2_err, mSO2_mdl

*SELECT so2
*SET ORDER TO sitdatim
*SET FILTER TO left(status,1)<>"F" and left(status,1)<>"R"
*SELECT logs
*SELECT so2
*If INLIST(mflagb, &ValidStat)
* IF .NOT. EOF('ions') .AND. mvolb <>0

mSO2 = -999
mSO2_err = -999
mSO2_mdl = -999

RETURN
```

C.14. Elems

PROCEDURE elems

```
*****
* Calculate elemental concentrations.
* Originally used Fe from Mo-XRF. Changed to YFe from Cu-XRF in ???
*
* Modified 10/11/2005 by Lowell to add spectral interference adjustment
* for XRF data. Prior to sole use of XRF, the S and Al concentrations
* were adjusted for interference from PB and BR within the PIXE analysis
* routines.
*
* Modified 10/12/2005 by Lowell to disallow the spectral correction from
* making S or Al negative
*****
```

```
PUBLIC elements(75)
PUBLIC mH, mH_err, mH_mdl, mNa, mNa_err, mNa_mdl, mAl, mAl_err, mAl_mdl
PUBLIC mSi, mSi_err, mSi_mdl, mS, mS_err, mS_mdl, mK, mK_err, mK_mdl
PUBLIC mCa, mCa_err, mCa_mdl, mFe, mFe_err, mFe_mdl, mTi, mTi_err, mTi_mdl
```

SELECT elements

```
IF INLIST(mflaga, 'UN', &ValidStat);
  .AND. .NOT. DELETED('elements');
  .AND. .NOT. EOF('elements') .AND. wts.postwght <> 0 .AND. mvola <>0
  IF samdat >= {12/1/2001} && Started Cu-XRF reporting on 12/1/2001
```

```
  SCATTER TO elements FIELDS;
    H, H_ERR, H_MDL,;
    YNA, YNA_ERR, YNA_MDL,;
    YMG, YMG_ERR, YMG_MDL,;
    YAL, YAL_ERR, YAL_MDL,;
    YSI, YSI_ERR, YSI_MDL,;
    YP, YP_ERR, YP_MDL,;
    YS, YS_ERR, YS_MDL,;
    YCL, YCL_ERR, YCL_MDL,;
    YK, YK_ERR, YK_MDL,;
    YCA, YCA_ERR, YCA_MDL,;
    YTI, YTI_ERR, YTI_MDL,;
    YV, YV_ERR, YV_MDL,;
    YCR, YCR_ERR, YCR_MDL,;
    YMN, YMN_ERR, YMN_MDL,;
    YFE, YFE_ERR, YFE_MDL,;
    NI, NI_ERR, NI_MDL,;
    CU, CU_ERR, CU_MDL,;
    ZN, ZN_ERR, ZN_MDL,;
    AS, AS_ERR, AS_MDL,;
    PB, PB_ERR, PB_MDL,;
    SE, SE_ERR, SE_MDL,;
    BR, BR_ERR, BR_MDL,;
    RB, RB_ERR, RB_MDL,;
    SR, SR_ERR, SR_MDL,;
    ZR, ZR_ERR, ZR_MDL
```

```
*
* Correct Pb/S and Br/Al interference
* Modified 10/12/2005 to disallow negative numbers
*
```

```
  Elements(19) = MAX(0, Elements(19) - 0.74 * Elements(58))
  Elements(10) = MAX(0, Elements(10) - 0.62 * Elements(64))
```

```
ELSE && PIXE data prior to
12/1/2001
  SCATTER TO elements FIELDS;
    H, H_ERR, H_MDL,;
    PNA, PNA_ERR, PNA_MDL,;
    PMG, PMG_ERR, PMG_MDL,;
    PAL, PAL_ERR, PAL_MDL,;
    PSI, PSI_ERR, PSI_MDL,;
    PP, PP_ERR, PP_MDL,;
```

```

        PS, PS_ERR, PS_MDL,;
        PCL, PCL_ERR, PCL_MDL,;
        PK, PK_ERR, PK_MDL,;
        PCA, PCA_ERR, PCA_MDL,;
        PTI, PTI_ERR, PTI_MDL,;
        PV, PV_ERR, PV_MDL,;
        PCR, PCR_ERR, PCR_MDL,;
        PMN, PMN_ERR, PMN_MDL,;
        FE, FE_ERR, FE_MDL,; && Changed this to Mo-XRF FE data-LLA 7/14/04
        NI, NI_ERR, NI_MDL,;
        CU, CU_ERR, CU_MDL,;
        ZN, ZN_ERR, ZN_MDL,;
        AS, AS_ERR, AS_MDL,;
        PB, PB_ERR, PB_MDL,;
        SE, SE_ERR, SE_MDL,;
        BR, BR_ERR, BR_MDL,;
        RB, RB_ERR, RB_MDL,;
        SR, SR_ERR, SR_MDL,;
        ZR, ZR_ERR, ZR_MDL,;
    ENDIF

* Convert from ng/cm2 to ng/m3
    FOR N=1 TO 75
        elements(N) = elements(N)*logs.area/mvola
    ENDFOR

ELSE
    FOR N=1 TO 75
        elements(N) = -999
    ENDFOR
ENDIF

IF NOT INLIST(elements.pesstat, &ValidStat)
    FOR N=1 TO 3
        elements(N) = -999
    ENDFOR
ENDIF

IF logs.samdat>{12/1/2001} && Use status flags for Cu-XRF
    IF NOT INLIST(elements.xrcstat, &ValidStat)
        FOR N=4 TO 45
            elements(N) = -999
        ENDFOR
    ENDIF
    IF NOT INLIST(elements.xrmstat, &ValidStat)
        FOR N=46 TO 75
            elements(N) = -999
        ENDFOR
    ENDIF
ELSE && Use status flags for PIXE prior to
    12/1/2001
    IF NOT INLIST(elements.pixstat, &ValidStat)
        FOR N=4 TO 42
            elements(N) = -999
        ENDFOR
    ENDIF
    IF NOT INLIST(elements.xrmstat, &ValidStat)
        FOR N=43 TO 75
            elements(N) = -999
        ENDFOR
    ENDIF
ENDIF

*Store some elemens in memory for later calculations
mH = elements(1)
mH_err = elements(2)
mH_mdl = elements(3)
mNa = elements(4)
mNa_err = elements(5)
mNa_mdl = elements(6)
mAl = elements(10)
mAl_err = elements(11)

```

```
mAl_mdl = elements(12)
mSi     = elements(13)
mSi_err = elements(14)
mSi_mdl = elements(15)
mS      = elements(19)
mS_err  = elements(20)
mS_mdl  = elements(21)
mK      = elements(25)
mK_err  = elements(26)
mK_mdl  = elements(27)
mCa     = elements(28)
mCa_err = elements(29)
mCa_mdl = elements(30)
mTi     = elements(31)
mTi_err = elements(32)
mTi_mdl = elements(33)
mFe     = elements(43)
mFe_err = elements(44)
mFe_mdl = elements(45)
```

```
RETURN
```

C.15. Laser

```
PROCEDURE laser
* Calculate LRNC from laser data

PUBLIC mLRNC, mLRNC_err, mLRNC_mdl

IF INLIST(mFlagA, &ValidStat)
  IF .NOT. EOF('laser') .AND. mvola <> 0 .AND. laser.status <> "XX" ;
  .AND. .NOT. DELETED('laser') .AND. laser.T1>0
* From BRAVO      REPLACE LRNC_err WITH SQRT((mav*225)^2 + (.03*lrnc)^2)/100
* From BRAVO      REPLACE LRNC_mdl WITH mav*4.50
* From Brian      mLRNC_err = ((0.1/(1000-laser.R1))*SQRT(1+((1000-
  laser.R1)/laser.T1)^2))/mConvA*10000
* Code below is from VAUXDATA
  mLRNC = 10000 * LOG((1000-laser.R1)/laser.T1) / mConvA
  mLRNC_err = SQRT((10000*0.045/(2*mCONVA))^2 + (mMFVol*mLRNC)^2)
  mLRNC_mDL = 10000. * 0.045/mCONVA
  mLRNC_mDL = IIF(mLRNC < 0.0 .and. mLRNC_mDL = 0.0, ABS(mLRNC), mLRNC_mDL)
  mLRNC      = IIF(mLRNC < mLRNC_mDL, 0.0, mLRNC)
  mLRNC_err  = IIF(mLRNC < mLRNC_mDL, 0.0, mLRNC_err)
  ELSE
  mLRNC = -999
  mLRNC_err = -999
  mLRNC_mdl = -999
  ENDIF
ELSE
  mLRNC = -999
  mLRNC_err = -999
  mLRNC_mdl = -999
ENDIF

RETURN
```

C.16. Composites

PROCEDURE Composites

* Calculate composite species
* 6/17/04 Nicole changed OMC calculation. OMC calculated with
* negative carbon fractions. Calculation used to convert negative
* carbon fractions into zeros.
* 4/6/05 Lowell changed the uncertainty calculation for OMC and LAC.
* Does not affect data sent to CIRA.

```
PUBLIC mOMH, mOMH_err, mOMH_mdl
PUBLIC mOMCN, mOMCN_err, mOMCN_mdl
PUBLIC mLACN, mLACN_err, mLACN_mdl
PUBLIC mSOIL, mSOIL_err, mSOIL_mdl
PUBLIC mNHSO, mNHSO_err, mNHSO_mdl
PUBLIC mNHNO, mNHNO_err, mNHNO_mdl
PUBLIC mKNON, mKNON_err, mKNON_mdl
PUBLIC mRCMA, mRCMA_err, mRCMA_mdl
PUBLIC mRCMC, mRCMC_err, mRCMC_mdl
PUBLIC mRCMI, mRCMI_err, mRCMI_mdl
PUBLIC mS3, mS3_err, mS3_mdl
PUBLIC mRCNH, mRCNH_err, mRCNH_mdl
PUBLIC mOMHC, mOMHC_err, mOMHC_mdl
```

*****OMH*****

```
If mH > 0.and.mS > mS_mDL
    mOMH = 13.75*(mH - (0.25*mS))
    mOMH_err = 13.75 * Sqrt((0.25*mS_err)^2 + mH_err^2)
    mOMH_mdl = 0
Else
    mOMH = -999
    mOMH_err = -999
    mOMH_mdl = -999
Endif
```

*****OMCN*****

```
If mO1 > 0 .or. mO1_err > 0
    mOMCN = 1.4*(mO1 + mO2 + mO3 + mO4 + mOP)
    * Uncertainty calculation changed 4/6/05. Does not affect data sent to CIRA.
    * mOMCN_err = 1.4*Sqrt((MAX(0,mO1_err))^2 + (MAX(0,mO2_err))^2 +
    (MAX(0,mO3_err))^2 + (MAX(0,mO4_err))^2 + (MAX(0,mOP_err))^2)
    mOMCN_err = Sqrt((168)^2 + (0.05*mOMCN)^2)
    mOMCN_mdl = 0
Else
    mOMCN = -999
    mOMCN_err = -999
    mOMCN_mdl = -999
Endif
```

*****LACN*****

```
If mE1 > 0 .or. mE1_err > 0
    mLACN = (mE1 + mE2 + mE3 - mOP)
    * Uncertainty calculation changed 4/6/05. Does not affect data sent to CIRA.
    * mLACN_err = Sqrt((MAX(0,mE1_err))^2 + (MAX(0,mE2_err))^2 + (MAX(0,mE3_err))^2
    + (MAX(0,mOP_err))^2)
    mLACN_err = Sqrt((34)^2 + (0.07*mLACN)^2)
    mLACN_mdl = 0
Else
    mLACN = -999
    mLACN_err = -999
    mLACN_mdl = -999
Endif
```

*****SOIL*****

```
If mSI_mDL > 0 .and. mFE_mDL > 0
    mSOIL = 3.48*MAX(mSI,mSI_mDL/2,0) + ;
    1.63*MAX(mCA,mCA_mDL/2,0) + ;
    1.94*MAX(mTI,mTI_mDL/2,0) + ;
    2.42*MAX(mFE,mFE_mDL/2,0)
    mSOIL_err = Sqrt((3.48*MAX(mSI_err,mSI_mDL/2,0))^2 + ;
```

```

(1.63*MAX(mCA_err,mCA_mDL/2,0))^2 + ;
(1.94*MAX(mTI_err,mTI_mDL/2,0))^2 + ;
(2.42*MAX(mFE_err,mFE_mDL/2,0))^2)

mSOIL_mDL = 0
Else
mSOIL = -999
mSOIL_err = -999
mSOIL_mdl = -999
Endif

*****NHSO*****
mNHSO = MAX(4.125 * MAX(mS, mS_mdl/2), -999)
mNHSO_err = MAX(4.125 * MAX(mS_err, mS_mdl/2), -999)
mNHSO_mdl = MAX(4.125 * mS_mdl, -999)

*****NHNO*****
mNHNO = MAX(-999, 1.29 * mNO3)
mNHNO_err = MAX(-999, 1.29 * mNO3_err)
mNHNO_mdl = MAX(-999, 1.29 * mNO3_mdl)

*****KNON*****
If mFE > 0 .and. mK > 0
mKNON = MAX(mK, mK_mDL) - 0.6*MAX(mFE, mFE_mDL)
mKNON_err = SQRT((0.6*mFE_err)^2+mK_err^2)
mKNON_mDL = 0
Else
mKNON = -999
mKNON_err = -999
mKNON_mdl = -999
Endif

*****RCMA*****
If mLRNC_mDL > 0
mRCMA = MAX(mNHSO, 0.0) + ;
MAX(mOMH, 0.0) + ;
MAX(mSOIL, 0.0) + ;
1.4*MAX(mKNON, 0.0) + ;
MAX(mLRNC, mLRNC_mDL/2, 0) + ;
2.5*MAX(mNA, mNA_mDL/2)
mRCMA_err = SQRT( MAX(0, mNHSO_err)^2 + ;
MAX(0, mOMH_err)^2 + ;
MAX(0, mSOIL_err)^2 + ;
(1.4*MAX(0, mKNON_err))^2 + ;
MAX(0, mLRNC_err)^2 + ;
(2.5*MAX(mNA_err, mNA_mDL/2, 0))^2)

mRCMA_mDL = 0
Else
mRCMA = -999
mRCMA_err = -999
mRCMA_mdl = -999
Endif

*****RCMC*****
If mOMCN_err > 0 .and. mLACN_err > 0
mRCMC = MAX(mNHSO, 0.0) + ;
MAX(mOMCN, 0.0) + ;
MAX(mSOIL, 0.0) + ;
1.4*MAX(mKNON, 0.0) + ;
MAX(mLACN, 0.0) + ;
2.5*MAX(mNA, mNA_mDL/2, 0)
mRCMC_err = SQRT( MAX(0, mNHSO_err)^2 + ;
MAX(0, mOMCN_err)^2 + ;
MAX(0, mSOIL_err)^2 + ;
(1.4*MAX(0, mKNON_err))^2 + ;
MAX(0, mLACN_err)^2 + ;
(2.5*MAX(mNA_err, mNA_mDL/2, 0))^2)

mRCMC_mDL = 0
Else
mRCMC = -999
mRCMC_err = -999
mRCMC_mdl = -999
Endif

```



```

*****RCMI*****
If mSO4_eRR > 0 .and. mLRNC_mDL > 0
  mRCMI = 1.375*MAX(mSO4,0.0) + ;
          MAX(mOMH, 0.0) + ;
          MAX(mSOIL,0.0) + ;
          1.4*MAX(mKNON,0.0) + ;
          MAX(mLRNC,mLRNC_mDL/2,0) + ;
          2.5*MAX(mNA,mNA_mDL/2,0)
  mRCMI_eRR = SQRT((1.375*MAX(0, mSO4_eRR))^2 + ;
                  MAX(0, mOMH_eRR)^2 + ;
                  MAX(0, mSOIL_eRR)^2 + ;
                  (1.4*MAX(0, mKNON_eRR))^2 + ;
                  MAX(0, mLRNC_eRR)^2 + ;
                  (2.5*MAX(mNA_eRR,mNA_mDL/2,0))^2)

  mRCMI_mDL = 0
Else
  mRCMI = -999
  mRCMI_err = -999
  mRCMI_mdl = -999
Endif

*****S3*****
IF mS > 0
  mS3 = 3*mS
  mS3_err = 3*mS_err
  mS3_mdl = 3*mS_mdl
ELSE
  mS3 = -999
  mS3_err = -999
  mS3_mdl = -999
ENDIF

*****MC*****
IF mMF > 0
  mMC = MAX(mMT - mMF,0.0)
  mMC_eRR = SQRT(mMT_eRR^2 + mMF_eRR^2)
  mMC_mDL = 0
ELSE
  mMC = -999
  mMC_eRR = -999
  mMC_mDL = -999
ENDIF

*****RCNH*****
IF mSO4 > 0 AND mNO3 > 0
  mRCNH = 0.375*mSO4 + 0.29*mNO3
  mRCNH_eRR = SQRT((0.375*mSO4_eRR)^2 + (0.29*mNO3_eRR)^2)
  mRCNH_mDL = 0
ELSE
  mRCNH = -999
  mRCNH_eRR = -999
  mRCNH_mDL = -999
ENDIF

*****OMHC*****
If mOMH_eRR > 0 .and. mNO3_eRR > 0
  mOMHC = mOMH - 2*0.447*mNO3
  mOMHC_eRR = SQRT(mOMH_eRR^2 + mNO3_eRR^2)
  mOMHC_mDL = 0
Else
  mOMHC = -999
  mOMHC_eRR = -999
  mOMHC_mDL = -999
Endif

RETURN

```

C.17. Site9

- * Process the "SITE9" files to add mass data to "SITE1" as MT and move MF to MT in
- * "SITE9". The MF listed for SITE9 is really PM10 mass, so should be in the MT field.

```
SELECT site, samdat, mf, mf_err, mf_mdl, mt, mt_err, mt_mdl,;
       flow_a, time_a, flag_a;
FROM out;
INTO TABLE SITE9FILE;
WHERE AT("9",site) > 0;
ORDER BY site, samdat
```

```
GO TOP
SELECT out
GO TOP
SELECT site9file
DO WHILE NOT EOF("SITE9FILE")
  msitename = LEFT(site9file.site,4)
  mdate = samdat
  msite = msitename + "1"
  SELECT out
  LOCATE FOR out.site = msite AND samdat = mdate
  IF site9file.mf>0
    REPLACE out.mt WITH site9file.mf
    REPLACE out.mt_err WITH site9file.mf_err
    REPLACE out.mt_mdl WITH site9file.mf_mdl
    REPLACE out.flow_d WITH site9file.flow_a
    REPLACE out.time_d WITH site9file.time_a
    REPLACE out.flag_d WITH site9file.flag_a
    LOCATE FOR site = site9file.site AND samdat = mdate
    REPLACE out.mt WITH site9file.mf
    REPLACE out.mt_err WITH site9file.mf_err
    REPLACE out.mt_mdl WITH site9file.mf_mdl
    REPLACE out.time_d with 0
    REPLACE out.mf WITH -999
    REPLACE out.mf_err WITH -999
    REPLACE out.mf_mdl WITH -999
  ENDIF
  SELECT site9file
  SKIP
ENDDO

RETURN
```

C.18. CleanUp

```
PROCEDURE cleanup
*****
* 10/13/2005 - Added MALO1 and MALO2 to the list of sites to remove from TOTFILE
*****

* Delete temporary files

mDefPath = SYS(5)+SYS(2003)
SET DEFAULT TO &mPathTemp

*DO Totfile
* Create the output file to send to CIRA

* This SQL code will select only the sites in SITEFILE.DBF
* to include in the file sent to CIRA. It then compares the resulting list
* to exclude sites held back for this quarter.

* Select only the network sites to include in the file to CIRA
* This processing takes place prior to adding in the corresponding "1" modules to
* the SITEX data
SELECT site;
FROM U:\Improve\QAPlots\Holds\Holdsite;
HAVING holdsite.qtr=mSeason AND holdsite.mon=VAL(mMonth);
AND NOT holdsite.cleared;
INTO ARRAY Holdsites

*SUSPEND
SELECT out.*;
FROM out LEFT JOIN sites ;
ON out.site = Sites.site;
WHERE out.site = sites.site;
INTO TABLE totfile

SELECT totfile
SET FILTER TO ASCAN(holdsites, site)>0
DELETE ALL
SET FILTER TO INLIST(site, "BLNK1", "HANCS", "INGAS", "MALO1", "MALO2")
DELETE ALL
SET FILTER TO
PACK

USE &mspcl IN 0 ALIAS spl
SELECT spl
APPEND FROM totfile FOR INLIST(RIGHT(site,1), "5", "9")
SELECT totfile
SET FILTER TO INLIST(RIGHT(site,1), "5", "9")
DELETE ALL
PACK
SET FILTER TO
USE &mfml IN 0 ALIAS fml
SELECT fml
APPEND FROM totfile
*COPY TO &mfml FOX2X FOR NOT (INLIST(site, "BLNK1", "HANCS", "INGAS");
OR INLIST(RIGHT(site,1), "5", "9"))
*COPY TO &mspcl FOX2X FOR INLIST(RIGHT(site,1), "5", "9")

* Process X module data to include in QA file
DO sitexfile

* Copy the output file to the QA file
SELECT out
COPY TO &mout FOX2X

*Include next line to stop processing before deleting temporary files.
*RETURN

CLOSE DATABASES ALL
DELETE FILE tlogs.dbf
```

```
DELETE FILE tcarbon.dbf
DELETE FILE tions.dbf
DELETE FILE tlaser.dbf
DELETE FILE tele.dbf
DELETE FILE tmag.dbf
DELETE FILE tvac.dbf
DELETE FILE twts.dbf
DELETE FILE tout.dbf
DELETE FILE tfinl.dbf
DELETE FILE tsites.dbf
DELETE FILE tso2.dbf
DELETE FILE totfile.dbf
DELETE FILE tlogs.cdx
DELETE FILE tsites.cdx
DELETE FILE tso2.cdx
DELETE FILE tcarbon.cdx
DELETE FILE tions.cdx
DELETE FILE tlaser.cdx
DELETE FILE tele.cdx
DELETE FILE tmag.cdx
DELETE FILE tvac.cdx
DELETE FILE twts.cdx
DELETE FILE site9file.dbf
DELETE FILE sitexfile.dbf
```

```
*WAIT CLEAR
```

```
SET DEFAULT TO &mDefPath
RETURN
```

C.19. SiteXFile

```
PROCEDURE sitexfile
* Processing for the "X Modules." Adds the complimentary module data to the X
  module data to
* complete the data suite for validation purposes.
* 6/16/04 Nicole
```

```
* copy only X site data records into a temporary file
```

```
wait window at 10,40 "Processing X files" nowait
SELECT *;
FROM out;
WHERE right( Out.site,1) = "X";
ORDER BY Out.site, Out.samdat;
INTO TABLE sitexfile
```

```
*SELECT ALL;
* FROM out;
* INTO TABLE sitexfile;
* WHERE RIGHT(out.site,1)="X";
* ORDER BY site, samdat
GO TOP
SELECT sites
GO TOP
SELECT out
GO TOP
SELECT sitexfile
GO TOP
DO WHILE NOT EOF("sitexfile")  && loops thru entire sitexfile
  sitexname = sitexfile.site
  sitename = LEFT(sitexfile.site,4)
  sitelname = sitename + "1"
  SELECT sites
  LOCATE FOR sites.site = sitexname
  configuration = sites.config
  SELECT out
  DO CASE  && determine if the X module is A, B, or C type and process
  appropriately
    CASE configuration = "A"  && X module is an A-type module
      DO WHILE sitexfile.site = sitexname AND NOT EOF("sitexfile")
        mdate = sitexfile.samdat
        SELECT out
        LOCATE FOR out.site = sitelname AND out.samdat = mdate
        DO xmodule_add_B
        DO xmodule_add_C
        DO xmodule_add_D
        DO Xcomposites
        SKIP in sitexfile
      ENDDO
    CASE configuration = "B"  && X module is a B-type module
      DO WHILE sitexfile.site = sitexname AND NOT EOF("sitexfile")
        mdate = sitexfile.samdat
        SELECT out
        LOCATE FOR out.site = sitelname AND out.samdat = mdate
        DO xmodule_add_A
        DO xmodule_add_C
        DO xmodule_add_D
        DO Xcomposites
        SKIP in sitexfile
      ENDDO
    CASE configuration = "C"  && X module is a C-type module
      DO WHILE sitexfile.site = sitexname AND NOT EOF("sitexfile")
        mdate = sitexfile.samdat
        SELECT out
        LOCATE FOR out.site = sitelname AND out.samdat = mdate
        DO xmodule_add_A
        DO xmodule_add_B
        DO xmodule_add_D
        DO Xcomposites
```

```

                SKIP in sitexfile
            ENDDO
        CASE configuration = "D"    && X module is a D-type module
            DO WHILE sitexfile.site = sitexname AND NOT EOF("sitexfile")
                mdate = sitexfile.samdat
                SELECT out
                LOCATE FOR out.site = sitelname AND out.samdat = mdate
                DO xmodule_add_A
                DO xmodule_add_B
                DO xmodule_add_C
                DO Xcomposites
                SKIP in sitexfile
            ENDDO
        ENDCASE

ENDDO

* Delete X records from out file
SELECT out
GO TOP
DELETE FOR right(site,1)="X"    && marks records for deletion
PACK                            && deletes marked records
* Append X records from sitexfiles
APPEND FROM sitexfile

RETURN

```

C.20. Xmodule_Add_A

PROCEDURE xmodule_add_a
* When the X module is not an A-type module, this procedure
* pastes the A-module routine data into the X site data set
* for validation purposes.
* Created 6/17/04 by Nicole
* Added LRNC to replacements on 6/24/04 Nicole

```
REPLACE sitexfile.flow_a WITH out.flow_a
REPLACE sitexfile.time_a WITH out.time_a
REPLACE sitexfile.flag_a WITH out.flag_a
REPLACE sitexfile.mf WITH out.mf
REPLACE sitexfile.mf_err WITH out.mf_err
REPLACE sitexfile.mf_mdl WITH out.mf_mdl
REPLACE sitexfile.H WITH out.H
REPLACE sitexfile.H_err WITH out.H_err
REPLACE sitexfile.H_mdl WITH out.H_mdl
REPLACE sitexfile.Na WITH out.Na
REPLACE sitexfile.Na_err WITH out.Na_err
REPLACE sitexfile.Na_mdl WITH out.Na_mdl
REPLACE sitexfile.Mg WITH out.Mg
REPLACE sitexfile.Mg_err WITH out.Mg_err
REPLACE sitexfile.Mg_mdl WITH out.Mg_mdl
REPLACE sitexfile.Al WITH out.Al
REPLACE sitexfile.Al_err WITH out.Al_err
REPLACE sitexfile.Al_mdl WITH out.Al_mdl
REPLACE sitexfile.Si WITH out.Si
REPLACE sitexfile.Si_err WITH out.Si_err
REPLACE sitexfile.Si_mdl WITH out.Si_mdl
REPLACE sitexfile.P WITH out.P
REPLACE sitexfile.P_err WITH out.P_err
REPLACE sitexfile.P_mdl WITH out.P_mdl
REPLACE sitexfile.S WITH out.S
REPLACE sitexfile.S_err WITH out.S_err
REPLACE sitexfile.S_mdl WITH out.S_mdl
REPLACE sitexfile.Cl WITH out.Cl
REPLACE sitexfile.Cl_err WITH out.Cl_err
REPLACE sitexfile.Cl_mdl WITH out.Cl_mdl
REPLACE sitexfile.K WITH out.K
REPLACE sitexfile.K_err WITH out.K_err
REPLACE sitexfile.K_mdl WITH out.K_mdl
REPLACE sitexfile.Ca WITH out.Ca
REPLACE sitexfile.Ca_err WITH out.Ca_err
REPLACE sitexfile.Ca_mdl WITH out.Ca_mdl
REPLACE sitexfile.Ti WITH out.Ti
REPLACE sitexfile.Ti_err WITH out.Ti_err
REPLACE sitexfile.Ti_mdl WITH out.Ti_mdl
REPLACE sitexfile.V WITH out.V
REPLACE sitexfile.V_err WITH out.V_err
REPLACE sitexfile.V_mdl WITH out.V_mdl
REPLACE sitexfile.Cr WITH out.Cr
REPLACE sitexfile.Cr_err WITH out.Cr_err
REPLACE sitexfile.Cr_mdl WITH out.Cr_mdl
REPLACE sitexfile.Mn WITH out.Mn
REPLACE sitexfile.Mn_err WITH out.Mn_err
REPLACE sitexfile.Mn_mdl WITH out.Mn_mdl
REPLACE sitexfile.Fe WITH out.Fe
REPLACE sitexfile.Fe_err WITH out.Fe_err
REPLACE sitexfile.Fe_mdl WITH out.Fe_mdl
REPLACE sitexfile.Ni WITH out.Ni
REPLACE sitexfile.Ni_err WITH out.Ni_err
REPLACE sitexfile.Ni_mdl WITH out.Ni_mdl
REPLACE sitexfile.Cu WITH out.Cu
REPLACE sitexfile.Cu_err WITH out.Cu_err
REPLACE sitexfile.Cu_mdl WITH out.Cu_mdl
REPLACE sitexfile.Zn WITH out.Zn
REPLACE sitexfile.Zn_err WITH out.Zn_err
REPLACE sitexfile.Zn_mdl WITH out.Zn_mdl
REPLACE sitexfile.AS WITH out.AS
```

```
REPLACE sitexfile.As_err WITH out.As_err
REPLACE sitexfile.As_mdl WITH out.As_mdl
REPLACE sitexfile.Pb WITH out.Pb
REPLACE sitexfile.Pb_err WITH out.Pb_err
REPLACE sitexfile.Pb_mdl WITH out.Pb_mdl
REPLACE sitexfile.Se WITH out.Se
REPLACE sitexfile.Se_err WITH out.Se_err
REPLACE sitexfile.Se_mdl WITH out.Se_mdl
REPLACE sitexfile.Br WITH out.Br
REPLACE sitexfile.Br_err WITH out.Br_err
REPLACE sitexfile.Br_mdl WITH out.Br_mdl
REPLACE sitexfile.Rb WITH out.Rb
REPLACE sitexfile.Rb_err WITH out.Rb_err
REPLACE sitexfile.Rb_mdl WITH out.Rb_mdl
REPLACE sitexfile.Sr WITH out.Sr
REPLACE sitexfile.Sr_err WITH out.Sr_err
REPLACE sitexfile.Sr_mdl WITH out.Sr_mdl
REPLACE sitexfile.Zr WITH out.Zr
REPLACE sitexfile.Zr_err WITH out.Zr_err
REPLACE sitexfile.Zr_mdl WITH out.Zr_mdl
REPLACE sitexfile.LRNC WITH out.LRNC
REPLACE sitexfile.LRNC_err WITH out.LRNC_err
REPLACE sitexfile.LRNC_mdl WITH out.LRNC_mdl
RETURN
```


C.21. Xmodule_Add_B

```
PROCEDURE xmodule_add_b
* When the X module is not a B-type module, this procedure
* pastes the B-module routine data into the X site data set
* for validation purposes.
* Created 6/17/04 by Nicole
* Changed Cl to Cld on 6/24/04 Nicole
```

```
SELECT sitexfile
REPLACE sitexfile.flow_b WITH out.flow_b
REPLACE sitexfile.time_b WITH out.time_b
REPLACE sitexfile.flag_b WITH out.flag_b
REPLACE sitexfile.Cld WITH out.Cld
REPLACE sitexfile.Cld_Err WITH out.Cld_Err
REPLACE sitexfile.Cld_MDL WITH out.Cld_MDL
REPLACE sitexfile.NO2 WITH out.NO2
REPLACE sitexfile.NO2_Err WITH out.NO2_Err
REPLACE sitexfile.NO2_MDL WITH out.NO2_MDL
REPLACE sitexfile.NO3 WITH out.NO3
REPLACE sitexfile.NO3_Err WITH out.NO3_Err
REPLACE sitexfile.NO3_MDL WITH out.NO3_MDL
REPLACE sitexfile.SO4 WITH out.SO4
REPLACE sitexfile.SO4_Err WITH out.SO4_Err
REPLACE sitexfile.SO4_MDL WITH out.SO4_MDL
REPLACE sitexfile.NH4 WITH out.NH4
REPLACE sitexfile.NH4_Err WITH out.NH4_Err
REPLACE sitexfile.NH4_MDL WITH out.NH4_MDL
RETURN
```

C.22. Xmodule_Add_C

```
PROCEDURE xmodule_add_c
* When the X module is not a C-type module, this procedure
* pastes the C-module routine data into the X site data set
* for validation purposes.
* Created 6/17/04 by Nicole
* Fixed typo on 6/24/04 Nicole

REPLACE sitexfile.flow_c WITH out.flow_c
REPLACE sitexfile.time_c WITH out.time_c
REPLACE sitexfile.flag_c WITH out.flag_c
REPLACE sitexfile.O1 WITH out.O1
REPLACE sitexfile.O1_ERR WITH out.O1_ERR
REPLACE sitexfile.O1_MDL WITH out.O1_MDL
REPLACE sitexfile.O2 WITH out.O2
REPLACE sitexfile.O2_ERR WITH out.O2_ERR
REPLACE sitexfile.O2_MDL WITH out.O2_MDL
REPLACE sitexfile.O3 WITH out.O3
REPLACE sitexfile.O3_ERR WITH out.O3_ERR
REPLACE sitexfile.O3_MDL WITH out.O3_MDL
REPLACE sitexfile.O4 WITH out.O4
REPLACE sitexfile.O4_ERR WITH out.O4_ERR
REPLACE sitexfile.O4_MDL WITH out.O4_MDL
REPLACE sitexfile.OP WITH out.OP
REPLACE sitexfile.OP_ERR WITH out.OP_ERR
REPLACE sitexfile.OP_MDL WITH out.OP_MDL
REPLACE sitexfile.E1 WITH out.E1
REPLACE sitexfile.E1_ERR WITH out.E1_ERR
REPLACE sitexfile.E1_MDL WITH out.E1_MDL
REPLACE sitexfile.E2 WITH out.E2
REPLACE sitexfile.E2_ERR WITH out.E2_ERR
REPLACE sitexfile.E2_MDL WITH out.E2_MDL
REPLACE sitexfile.E3 WITH out.E3
REPLACE sitexfile.E3_ERR WITH out.E3_ERR
REPLACE sitexfile.E3_MDL WITH out.E3_MDL
RETURN
```

C.23. Xmodule_Add_D

PROCEDURE xmodule_add_d

- * When the X module is not a D-type module, this procedure
- * pastes the D-module routine data into the X site data set
- * for validation purposes.
- * Created 6/17/04 by Nicole

```
REPLACE sitexfile.flow_d WITH out.flow_d
REPLACE sitexfile.time_d WITH out.time_d
REPLACE sitexfile.flag_d WITH out.flag_d
REPLACE sitexfile.mt WITH out.mt
REPLACE sitexfile.mt_err WITH out.mt_err
REPLACE sitexfile.mt_mdl WITH out.mt_mdl
REPLACE sitexfile.SO2 WITH out.SO2
REPLACE sitexfile.SO2_err WITH out.SO2_err
REPLACE sitexfile.SO2_mdl WITH out.SO2_mdl
RETURN
```

C.24. Xcomposites

```
PROCEDURE Xcomposites
* Calculate composites for X module data records, which are
* compiled after the composites for the routine data are calculated.
* 6/17/04 Nicole created

*****OMH*****
IF sitexfile.H > 0 AND sitexfile.S > -999
  REPLACE sitexfile.Omh WITH 13.75*(sitexfile.H - (0.25 * sitexfile.S))
  REPLACE sitexfile.Omh_err WITH 13.75 * SQRT((0.25 * sitexfile.S_err)^2 +
  sitexfile.H_err^2)
  REPLACE sitexfile.Omh_md1 WITH 0
ELSE
  REPLACE sitexfile.Omh WITH -999
  REPLACE sitexfile.Omh_err WITH -999
  REPLACE sitexfile.Omh_md1 WITH -999
ENDIF

*****OMCN*****
IF sitexfile.O1 <> -999
  REPLACE sitexfile.OMCN WITH 1.4*(sitexfile.O1 + sitexfile.O2 +
  sitexfile.O3 + sitexfile.O4 + sitexfile.OP)
  REPLACE sitexfile.OMCN_err WITH 1.4*SQRT((MAX(0,sitexfile.O1_err))^2 +
  (MAX(0,sitexfile.O2_err))^2 + (MAX(0,sitexfile.O3_err))^2 +
  (MAX(0,sitexfile.O4_err))^2 + (MAX(0,sitexfile.OP_err))^2)
  REPLACE sitexfile.OMCN_md1 WITH 0
ELSE
  REPLACE sitexfile.OMCN WITH -999
  REPLACE sitexfile.OMCN_err WITH -999
  REPLACE sitexfile.OMCN_md1 WITH -999
ENDIF

*****LACN*****
IF sitexfile.E1 <> -999
  REPLACE sitexfile.LACN WITH (sitexfile.E1 + sitexfile.E2 + sitexfile.E3 -
  sitexfile.OP)
  REPLACE sitexfile.LACN_err WITH SQRT((MAX(0,sitexfile.E1_err))^2 +
  (MAX(0,sitexfile.E2_err))^2 + (MAX(0,sitexfile.E3_err))^2 +
  (MAX(0,sitexfile.OP_err))^2)
  REPLACE sitexfile.LACN_md1 WITH 0
ELSE
  REPLACE sitexfile.LACN WITH -999
  REPLACE sitexfile.LACN_err WITH -999
  REPLACE sitexfile.LACN_md1 WITH -999
ENDIF

*****SOIL*****
IF sitexfile.SI_md1 > 0 AND sitexfile.FE_md1 > 0
  REPLACE sitexfile.SOIL WITH 3.48*MAX(sitexfile.SI,sitexfile.SI_md1/2,0) + ;
  1.63*MAX(sitexfile.CA,sitexfile.CA_md1/2,0) + ;
  1.94*MAX(sitexfile.TI,sitexfile.TI_md1/2,0) + ;
  2.42*MAX(sitexfile.FE,sitexfile.FE_md1/2,0)
  REPLACE sitexfile.SOIL_err WITH SQRT(
  (3.48*MAX(sitexfile.SI_err,sitexfile.SI_md1/2,0))^2 + ;
  (1.63*MAX(sitexfile.CA_err,sitexfile.CA_md1/2,0))^2 + ;
  (1.94*MAX(sitexfile.TI_err,sitexfile.TI_md1/2,0))^2 + ;
  (2.42*MAX(sitexfile.FE_err,sitexfile.FE_md1/2,0))^2)
  REPLACE sitexfile.SOIL_md1 WITH 0
ELSE
  REPLACE sitexfile.SOIL WITH -999
  REPLACE sitexfile.SOIL_err WITH -999
  REPLACE sitexfile.SOIL_md1 WITH -999
ENDIF

*****NHISO*****
REPLACE sitexfile.NHISO WITH MAX(4.125 * MAX(sitexfile.S, sitexfile.S_md1/2), -
999)
REPLACE sitexfile.NHISO_err WITH MAX(4.125 * MAX(sitexfile.S_err,
sitexfile.S_md1/2), -999)
```

```

REPLACE sitexfile.NHSO_MDL WITH MAX(4.125 * sitexfile.S_MDL,-999)

*****NHNO*****
REPLACE sitexfile.NHNO WITH MAX(-999,1.29 * sitexfile.NO3)
REPLACE sitexfile.NHNO_err WITH MAX(-999,1.29 * sitexfile.NO3_err)
REPLACE sitexfile.NHNO_MDL WITH MAX(-999,1.29 * sitexfile.NO3_MDL)

*****KNON*****
IF sitexfile.FE > 0 AND sitexfile.K > 0
  REPLACE sitexfile.KNON WITH MAX(sitexfile.K,sitexfile.K_MDL) -
  0.6*MAX(sitexfile.FE,sitexfile.FE_MDL)
  REPLACE sitexfile.KNON_err WITH
  SQRT((0.6*sitexfile.FE_err)^2+sitexfile.K_err^2)
  REPLACE sitexfile.KNON_MDL WITH 0
ELSE
  REPLACE sitexfile.KNON WITH -999
  REPLACE sitexfile.KNON_err WITH -999
  REPLACE sitexfile.KNON_MDL WITH -999
ENDIF

*****RCMA*****
IF sitexfile.LRNC_MDL > 0 AND sitexfile.Omh <> -999 AND sitexfile.SOIL <> -999
  AND sitexfile.NHSO <> -999
  REPLACE sitexfile.RCMA WITH MAX(sitexfile.NHSO,0.0) + ;
  MAX(sitexfile.Omh, 0.0) + ;
  MAX(sitexfile.SOIL,0.0) + ;
  1.4*MAX(sitexfile.KNON,0.0) + ;
  MAX(sitexfile.LRNC,sitexfile.LRNC_MDL/2,0)+ ;
  2.5*MAX(sitexfile.NA,sitexfile.NA_MDL/2)
  REPLACE sitexfile.RCMA_err WITH SQRT(MAX(0, sitexfile.NHSO_err)^2 + ;
  MAX(0, sitexfile.Omh_err)^2 + ;
  MAX(0, sitexfile.SOIL_err)^2 + ;
  (1.4*MAX(0, sitexfile.KNON_err))^2 + ;
  MAX(0, sitexfile.LRNC_err)^2 + ;
  (2.5*MAX(sitexfile.NA_err,sitexfile.NA_MDL/2,0))^2)
  REPLACE sitexfile.RCMA_MDL WITH 0
ELSE
  REPLACE sitexfile.RCMA WITH -999
  REPLACE sitexfile.RCMA_err WITH -999
  REPLACE sitexfile.RCMA_MDL WITH -999
ENDIF

*****RCMC*****
IF sitexfile.OMCN_err > 0 AND sitexfile.LACN_err > 0 AND sitexfile.SOIL <> -999
  AND sitexfile.NHSO <> -999
  REPLACE sitexfile.RCMC WITH MAX(sitexfile.NHSO,0.0) + ;
  MAX(sitexfile.OMCN,0.0) + ;
  MAX(sitexfile.SOIL,0.0) + ;
  1.4*MAX(sitexfile.KNON,0.0) + ;
  MAX(sitexfile.LACN,0.0) + ;
  2.5*MAX(sitexfile.NA,sitexfile.NA_MDL/2,0)
  REPLACE sitexfile.RCMC_err WITH SQRT( MAX(0, sitexfile.NHSO_err)^2 + ;
  MAX(0, sitexfile.OMCN_err)^2 + ;
  MAX(0, sitexfile.SOIL_err)^2 + ;
  (1.4*MAX(0, sitexfile.KNON_err))^2 + ;
  MAX(0, sitexfile.LACN_err)^2 + ;
  (2.5*MAX(sitexfile.NA_err,sitexfile.NA_MDL/2,0))^2)
  REPLACE sitexfile.RCMC_MDL WITH 0
ELSE
  REPLACE sitexfile.RCMC WITH -999
  REPLACE sitexfile.RCMC_err WITH -999
  REPLACE sitexfile.RCMC_MDL WITH -999
ENDIF

*****RCMI*****
IF sitexfile.SO4_err > 0 AND sitexfile.LRNC_MDL > 0 AND sitexfile.Omh <> -999 AND
  sitexfile.SOIL <> -999
  REPLACE sitexfile.RCMI WITH 1.375*MAX(sitexfile.SO4,0.0) + ;
  MAX(sitexfile.Omh, 0.0) + ;
  MAX(sitexfile.SOIL,0.0) + ;
  1.4*MAX(sitexfile.KNON,0.0) + ;
  MAX(sitexfile.LRNC,sitexfile.LRNC_MDL/2,0) + ;

```

```

2.5*MAX(sitexfile.NA,sitexfile.NA_MDL/2,0)
REPLACE sitexfile.RCMI_eRR WITH SQRT((1.375*MAX(0, sitexfile.SO4_eRR))^2 + ;
MAX(0, sitexfile.Omh_err)^2 + ;
MAX(0, sitexfile.SOIL_eRR)^2 + ;
(1.4*MAX(0, sitexfile.KNON_eRR))^2 + ;
MAX(0, sitexfile.LRNC_eRR)^2 + ;
(2.5*MAX(sitexfile.NA_eRR,sitexfile.NA_MDL/2,0))^2)
REPLACE sitexfile.RCMI_MDL WITH 0
ELSE
REPLACE sitexfile.RCMI WITH -999
REPLACE sitexfile.RCMI_eRR WITH -999
REPLACE sitexfile.RCMI_MDL WITH -999
ENDIF

*****S3*****
IF sitexfile.S > 0
REPLACE sitexfile.S3 WITH 3*sitexfile.S
REPLACE sitexfile.S3_err WITH 3*sitexfile.S_eRR
REPLACE sitexfile.S3_MDL WITH 3*sitexfile.S_MDL
ELSE
REPLACE sitexfile.S3 WITH -999
REPLACE sitexfile.S3_err WITH -999
REPLACE sitexfile.S3_MDL WITH -999
ENDIF

*****MC*****
*IF sitexfile.MF <> -999 AND sitexfile.MT <> -999
* REPLACE sitexfile.MC WITH MAX(sitexfile.MT - sitexfile.MF,0.0)
* REPLACE sitexfile.MC_eRR WITH SQRT(sitexfile.MT_eRR^2 + sitexfile.MF_eRR^2)
* REPLACE sitexfile.MC_MDL WITH 0
*ELSE
* REPLACE sitexfile.MC WITH -999
* REPLACE sitexfile.MC_eRR WITH -999
* REPLACE sitexfile.MC_MDL WITH -999
*ENDIF

*****RCNH*****
IF sitexfile.SO4 > 0 AND sitexfile.NO3 > 0
REPLACE sitexfile.RCNH WITH 0.375*sitexfile.SO4 + 0.29*sitexfile.NO3
REPLACE sitexfile.RCNH_eRR WITH SQRT((0.375*sitexfile.SO4_eRR)^2 +
(0.29*sitexfile.NO3_err)^2)
REPLACE sitexfile.RCNH_MDL WITH 0
ELSE
REPLACE sitexfile.RCNH WITH -999
REPLACE sitexfile.RCNH_eRR WITH -999
REPLACE sitexfile.RCNH_MDL WITH -999
ENDIF

*****OMHC*****
IF sitexfile.Omh_err > 0 AND sitexfile.NO3_err > 0
REPLACE sitexfile.OMHC WITH sitexfile.Omh - 2*0.447*sitexfile.NO3
REPLACE sitexfile.OMHC_eRR WITH SQRT(sitexfile.Omh_err^2 +
sitexfile.NO3_err^2)
REPLACE sitexfile.OMHC_MDL WITH 0
ELSE
REPLACE sitexfile.OMHC WITH -999
REPLACE sitexfile.OMHC_eRR WITH -999
REPLACE sitexfile.OMHC_MDL WITH -999
ENDIF

RETURN

```

C.25. QlogsErr

```
PROCEDURE QLOGSERR
PARAMETER mFile, mERROR, mMessage
SET ESCAPE OFF
WAIT WINDOW ;
  "Error on "+mFile + ". May be not on Disk or already open" ;
  + CHR(13) + CHR(10) + mMessage + ;
  + CHR(13) + "Hit ESC or ENTER to return to program" TIMEOUT 2
* Wait &mLogFile window Timeout 1
DO CASE
CASE LASTKEY() = 138
  DEACTIVATE WINDOW ALL
  CLOSE DATABASES ALL
  CANCEL
* QUIT
CASE LASTKEY()=13 OR LASTKEY()=32 OR LASTKEY()=27
  mBadLog = .T.
  RETURN
OTHERWISE
  mBadLog = .T.
  RETRY          &&
ENDCASE

IF SELECT('logs')<>0
  SELECT logs
  USE
ENDIF
SET ESCAPE ON
RETURN
```

C.26. Errhand

```
PROCEDURE errhand
PARAMETER merror, mess, mess1, mprog, mlineno
CLEAR
? 'Error number: ' + LTRIM(STR(merror))
? 'Error message: ' + mess
? 'Line of code with error: ' + mess1
? 'Line number of error: ' + LTRIM(STR(mlineno))
? 'Program with error: ' + mprog
*CLOSE DATABASES ALL
WAIT
ON ERROR && restore system error handler
CANCEL
```


APPENDIX D. SWAP CODE

Table of Contents

| | | |
|-------|-------------------|-------|
| D.1. | Swap..... | D-2 |
| D.2. | ErrHand..... | D-19 |
| D.3. | EditForm..... | D-20 |
| D.4. | QLogsErr..... | D-28 |
| D.5. | Errhand..... | D-29 |
| D.6. | EditLogs..... | D-30 |
| D.7. | EditFlows..... | D-31 |
| D.8. | EditVFlows..... | D-32 |
| D.9. | EditWeights..... | D-33 |
| D.10. | CalcWts..... | D-35 |
| D.11. | FBLevel2..... | D-39 |
| D.12. | GenXTab..... | D-40 |
| D.13. | EditIons..... | D-56 |
| D.14. | EditCarbon..... | D-57 |
| D.15. | FBions..... | D-58 |
| D.16. | CarbonCS..... | D-60 |
| D.17. | EditLaser..... | D-64 |
| D.18. | EditSO2..... | D-65 |
| D.19. | EditProblems..... | D-66 |
| D.20. | EditHolds..... | D-67 |
| D.21. | EditFlash..... | D-68 |
| D.22. | EditMag..... | D-69 |
| D.23. | EditVac..... | D-70 |
| D.24. | EditBal..... | D-71 |
| D.25. | FlowRSD..... | D-72 |
| D.26. | AllSites..... | D-79 |
| D.27. | SQLSel..... | D-81 |
| D.28. | FlowRate..... | D-83 |
| D.29. | AllFlows..... | D-90 |
| D.30. | SQLFlow..... | D-92 |
| D.31. | FlowCalc..... | D-93 |
| D.32. | SwapCheck..... | D-96 |
| D.33. | GetMonth..... | D-98 |
| D.34. | ArtHist..... | D-101 |
| D.35. | ClearQD..... | D-102 |
| D.36. | OutCheck..... | D-103 |

D.1. Swap

PROCEDURE SWAP

```
PUBLIC mSite, mSeason, mVolsummary, mFile, mThisSeason, mLogFile, medit, nofile
PUBLIC mBadLog, CurrSeas, mMonth, mYear
PUBLIC mGETENV, mVolsummary, mRootPath, mCarbonPath, mIonsPath, mXRFPPath, mLaserPath
PUBLIC mWtsPath, mLogsPath, mCalPath, mSiteFile, mSuppPath, mHoldPath, mLogFile
PUBLIC mFlashPath, EditFile, mUSER, mPathTemp, mQtr
```

```
DIMENSION QTR(4)
mBadLog = .F.
```

```
ON KEY LABEL CTRL+F12 SUSPEND
*ON ERROR DO errhand WITH ;
ERROR( ), MESSAGE( ), MESSAGE(1), PROGRAM( ), LINENO( )
*ON ERROR DO Qlogserr
```

```
ON KEY LABEL CTRL+F11 DO EDITFORM
```

```
SET TALK OFF
SET SAFETY OFF  && Make this active to remove dialog box on overwriting site and
sites.
```

```
QTR(1) = "A"
QTR(2) = "B"
QTR(3) = "C"
QTR(4) = "D"
mUSER = GETENV('USERNAME')
```

```
CLEAR
```

```
CLOSE DATABASES ALL
STORE SYS(5)+SYS(2003) TO DefaultDir
IF DIRECTORY("M:\")
    mPathTemp = "m:\improve\Temp\"
    IF NOT DIRECTORY("&mPathTemp")
        RUN MD &mPathTemp
    ENDIF
    IF NOT DIRECTORY("m:\IMPROVE\PROGRAMS")
        RUN MD "m:\IMPROVE\PROGRAMS"
    ENDIF
    SET PATH TO "m:\IMPROVE\PROGRAMS", mdisk + "IMPROVE\PROGRAMS"
```

```
ELSE
    SET PATH TO mdisk + "IMPROVE\PROGRAMS"
    IF NOT DIRECTORY(mdisk + "improve\Temp\")
        tdisk = &mdisk + "m:\improve\Temp\"
        RUN MD &tdisk
    ENDIF
    mPathTemp = mdisk + "improve\Temp\"
ENDIF
```

```
mGETENV      = 'M:\zDEFAULT\FOXPRO\'
mVolsummary  = mGETENV + "volsummary.dbf"
mRootPath    = "U:\IMPROVE\"
mCarbonPath  = mRootPath + "Carbon\"
mIonsPath    = mRootPath + "Ions\"
mXRFPPath    = mRootPath + "PIXEXRF\"
mLaserPath   = mRootPath + "Laser\"
mWtsPath     = mRootPath + "Weights\"
mLogsPath    = mRootPath + "Logs\"
mCalPath     = mRootPath + "SiteCal\"
mSiteFile    = mCalPath + "sitefile.dbf"
mSuppPath    = mRootPath + "Support\"
mHoldPath    = mRootPath + "QAPlots\Holds\"
*mLogFile    = mLogsPath + "logs.dbf"
mLogFile     = "u:\improve\logs\logs.dbf"
mso2path     = mrootpath + "so2\"
mFlashPath   = mRootPath + "FlashDBF\"
EditFile     = mLogFile
```

```
STORE SYS(5)+SYS(2003) TO mDefDir
&& Moved section from here
```

```

IF NOT DIRECTORY("M:\IMPROVE\Temp")
  MKDIR "M:\IMPROVE\Temp"
ENDIF
SET DEFAULT TO M:\IMPROVE\Temp
ON ERROR DO QLOGSERR WITH mLogFile, ERROR(), MESSAGE()
USE &mLogFile IN 0 ALIAS logs
IF mBadLog
  LOOP
ENDIF
ON ERROR DO errhand WITH ;
  ERROR( ), MESSAGE( ), MESSAGE(1), PROGRAM( ), LINENO( )
GO top
mSite = logs.site
mDate = logs.samdat
SELECT DISTINCT logs.site;
  FROM logs;
  ORDER BY logs.site;
  INTO TABLE LogSites
*INSERT INTO Sites (site) values (" ALL")
*INSERT INTO Sites (site) values (" QUIT")
INDEX ON site TAG site
GO TOP
IF LogSites.site="  "
  DELETE
  PACK
ENDIF
GO TOP
mSite=LogSites.site

*GO TOP IN Site
SELECT logs
USE

SET DEFAULT TO U:\IMPROVE\SUPPORT

mMonth = MONTH(mDate)
mYear = YEAR(mDate)
mQtr = QTR(mMonth/3) + RIGHT(STR(mYear),2)
mSeason = mQtr
CurrSeas = mSeason
*STORE "U:\IMPROVE\FLASHDBF\" to mFlashDir
*IF SELECT("volsummary")>0
* SELECT volsummary
* USE
*ENDIF
*IF FILE('&mvolsummary')
* DELETE FILE &mvolsummary.dbf
*ENDIF

DO WHILE .T.
  medit=" "
* DO FormInput
  mBadLog=.F.
  frmInput = CREATEOBJECT('Form')
  frmInput.CLOSABLE = .F.
  frmInput.MOVABLE = .T.
  frmInput.BORDERSTYLE = 3
  IF mUSER = "ashbaugh"
    frmInput.MOVE(0,0,424,494)
  ELSE
    frmInput.MOVE(0,0,424,396)
  ENDIF
  frmInput.CAPTION = "Edit data records for selected site/quarter"
  frmInput.NAME = "Form1"
  frmInput.ADDOBJECT('Label1', 'SiteLabel')
  frmInput.ADDOBJECT('Label2', 'SeasonLabel')
  frmInput.ADDOBJECT('Label3', 'LogsLabel')
  frmInput.ADDOBJECT('Label4', 'BlanksLabel')
  frmInput.ADDOBJECT('Label5', 'DataLabel')
  frmInput.ADDOBJECT('Label6', 'CalLabel')
  frmInput.ADDOBJECT('Label7', 'BalLabel')
  frmInput.ADDOBJECT('mSite', 'SiteBox')

```

```

frmInput.ADDOBJECT('mSeason', 'SeasonBox')
frmInput.ADDOBJECT('LogStat', 'EditLogs')
frmInput.ADDOBJECT('LogFlow', 'EditFlow')
frmInput.ADDOBJECT('LogVFlow', 'EditVFlow')
frmInput.ADDOBJECT('QuitBtn', 'QuitBtn1')
frmInput.ADDOBJECT('FBs', 'FBLevel')
frmInput.ADDOBJECT('IonsFB', 'EditIonsFB')
frmInput.ADDOBJECT('CarbonCS', 'CarbonCs')
frmInput.ADDOBJECT('Ions', 'EditIons')
frmInput.ADDOBJECT('XRF', 'EditXRF')
frmInput.ADDOBJECT('Weights', 'EditWeights')
frmInput.ADDOBJECT('Problems', 'EditProb')
frmInput.ADDOBJECT('Carbon', 'EditCarbon')
frmInput.ADDOBJECT('Laser', 'EditLaser')
frmInput.ADDOBJECT('SO2', 'EditSO2')
frmInput.ADDOBJECT('Flash', 'EditFlash')
frmInput.ADDOBJECT('MagCal', 'EditMag')
frmInput.ADDOBJECT('VacCal', 'EditVac')
frmInput.ADDOBJECT('Bal25', 'Bal25')
frmInput.ADDOBJECT('Bal30', 'Bal30')
frmInput.ADDOBJECT('Bal30a', 'Bal30a')
frmInput.ADDOBJECT('Bal31a', 'Bal31a')
IF INLIST(mUSER, "ashbaugh")
    frmInput.ADDOBJECT('Label8', 'ProcLabel')
    frmInput.ADDOBJECT('ArtHist', 'ArtHist')
    frmInput.ADDOBJECT('FlowRSD', 'FlowRSD')
    frmInput.ADDOBJECT('BuildDBF', 'BuildDBF')
    frmInput.ADDOBJECT('Holds', 'EditHold')
    frmInput.ADDOBJECT('ClearQD', 'ClearQD')
    frmInput.ADDOBJECT('FlowRate', 'FlowRate')
    frmInput.ADDOBJECT('EditForm', 'EditForm')
    frmInput.ADDOBJECT('SwapCheck', 'SwapCheck')
    frmInput.ADDOBJECT('FinalCheck', 'FinalCheck')
    frmInput.Label8.VISIBLE = .T.
    frmInput.ArtHist.VISIBLE = .T.
    frmInput.FlowRSD.VISIBLE = .T.
    frmInput.BuildDBF.VISIBLE = .T.
    frmInput.Holds.VISIBLE = .T.
    frmInput.ClearQD.VISIBLE = .T.
    frmInput.FlowRate.VISIBLE = .T.
    frmInput.EditForm.VISIBLE = .T.
    frmInput.SwapCheck.VISIBLE = .T.
    frmInput.FinalCheck.VISIBLE = .T.
ENDIF
frmInput.Label11.VISIBLE = .T.
frmInput.Label12.VISIBLE = .T.
frmInput.Label13.VISIBLE = .T.
frmInput.Label14.VISIBLE = .T.
frmInput.Label15.VISIBLE = .T.
frmInput.Label16.VISIBLE = .T.
frmInput.Label17.VISIBLE = .T.
frmInput.mSite.VISIBLE = .T.
frmInput.mSeason.VISIBLE = .T.
frmInput.LogStat.VISIBLE = .T.
frmInput.LogFlow.VISIBLE = .T.
frmInput.LogVFlow.VISIBLE = .T.
frmInput.QuitBtn.VISIBLE = .T.
frmInput.FBs.VISIBLE = .T.
frmInput.IonsFB.VISIBLE = .T.
frmInput.CarbonCS.VISIBLE = .T.
frmInput.Ions.VISIBLE = .T.
frmInput.XRF.VISIBLE = .T.
frmInput.weights.VISIBLE = .T.
frmInput.Problems.VISIBLE = .T.
frmInput.Carbon.VISIBLE = .T.
frmInput.Laser.VISIBLE = .T.
frmInput.SO2.VISIBLE = .T.
frmInput.Flash.VISIBLE = .T.
frmInput.MagCal.VISIBLE = .T.
frmInput.VacCal.VISIBLE = .T.
frmInput.Bal25.VISIBLE = .T.
frmInput.Bal30.VISIBLE = .T.

```

```

frmInput.Bal30a.VISIBLE = .T.
frmInput.Bal31a.VISIBLE = .T.

frmInput.mSite.VISIBLE = .T.
SELECT LogSites
LOCATE FOR LogSites.site=mSite
frmInput.mSite.VALUE = mSite  && Comment this out to return to same site
frmInput.SHOW
READ EVENTS
*? msite, mLogfile, mFile
   nofile = .F.

* Object definitions for display form follow
* ON ERROR DO errhand WITH ;
ERROR( ), MESSAGE( ), MESSAGE(1), PROGRAM( ), LINENO( )
ON ERROR DO QLOGSERR WITH EditFile, ERROR(), MESSAGE()

DO CASE
CASE medit="Logs"
  DO EditLogs
CASE medit="Flows"
  DO EditFlows
CASE medit="VFlows"
  DO EditVFlows
CASE medit="Weights"
  DO EditWeights
CASE medit="FBs"
  DO FBLevel2
CASE medit="Ions"
  DO EditIons
CASE medit="Carbon"
  DO EditCarbon
CASE medit="FBions"
  DO FBions
CASE medit="CSCarbon"
  DO CarbonCS
CASE medit="XRF"
  DO EditXRF
CASE medit="Laser"
  DO EditLaser
CASE medit="SO2"
  DO EditSO2
CASE medit="Problems"
  DO EditProblems
CASE medit="Holds"
  DO EditHolds
CASE medit="Flash"
  DO EditFlash
CASE medit="Mag"
  DO EditMag
CASE medit="Vac"
  DO EditVac
CASE medit="Edit Form"
  mSite=LogSites.site
  DO EditForm
CASE medit="25"
  EditFile = mSuppPath + "BalEq25.DBF"
  cTitle = "Cahn " + medit + " Balance Equation"
  DO EditBal
CASE medit="30 "
  EditFile = mSuppPath + "BalEq30.DBF"
  cTitle = "Cahn " + medit + " Balance Equation"
  DO EditBal
CASE medit="30a"
  EditFile = mSuppPath + "BalEq30a.DBF"
  cTitle = "Cahn " + medit + " Balance Equation"
  DO EditBal
CASE medit="31a"
  EditFile = mSuppPath + "BalEq31a.DBF"
  cTitle = "Cahn " + medit + " Balance Equation"
  DO EditBal
CASE medit="FlowRSD"

```

```

        mSite = LogSites.site
        DO FlowRSD.prg WITH .T.
        SET safety off
*       SELECT site
*       LOCATE FOR site.site=sites.site
        CASE medit="FlowRate"
            mSite = LogSites.site
            DO flowrate.prg WITH .T.
            LOOP
        CASE medit="BuildDBF"
            HoldSeason = mSeason
            DO BuildDBF.prg WITH .T.
            IF SELECT("logsites")=0
                USE m:\improve\temp\logsites IN 0
            ENDIF
            mSeason = HoldSeason
            LOOP
        CASE medit="SwapCheck"
            DO SwapCheck.prg WITH .T.
            LOOP
        CASE medit="ArtHist"
            DO ArtHist.prg
            LOOP
        CASE medit="ClearQD"
            DO ClearQD.prg
            LOOP
        CASE medit="Check"
            DO OutCheck.prg
            LOOP
        OTHERWISE
            DO EditLogs
    ENDCASE

*frmInput.Grid1.height = reccount("volsummary")*20
    SET DEFAULT TO M:\IMPROVE\Temp
    ON ERROR DO QLOGSERR WITH mLogFile, ERROR(), MESSAGE()
    IF SELECT('logs')<>0
        SELECT logs
        USE
    ENDIF
    USE &mLogFile IN 0 ALIAS logs
    IF NOT mBadLog
        SELECT logs
        ON ERROR DO errhand WITH ;
            ERROR( ), MESSAGE( ), MESSAGE(1), PROGRAM( ), LINENO( )
        GO TOP
*       mSite = logs.site
*       mDate = logs.samdat
        SELECT DISTINCT logs.site;
            FROM logs;
            ORDER BY logs.site;
            INTO TABLE LogSites
*INSERT INTO Sites (site) values (" ALL")
*INSERT INTO Sites (site) values (" QUIT")
        INDEX ON site TAG site
        GO TOP
        IF LogSites.site=" "
            DELETE
            PACK
        ENDIF
    ENDIF

*GO TOP IN Site
    SELECT logs
    USE

*SET DEFAULT TO U:\IMPROVE\SUPPORT

ENDDO
DEFINE CLASS SiteLabel AS LABEL
    AUTOSIZE = .F.
    FONTBOLD = .T.

```

```

    FONTSIZE = 9
    ALIGNMENT = 0
    CAPTION = "Site Name:"
    HEIGHT = 20
    LEFT = 35
    TOP = 24
    WIDTH = 75
    NAME = "Label1"
ENDDDEFINE

DEFINE CLASS SeasonLabel AS LABEL
    AUTOSIZE = .F.
    FONTBOLD = .T.
    FONTSIZE = 9
    ALIGNMENT = 1
    CAPTION = "Season:"
    TOP = 28
    LEFT = 220
    HEIGHT = 20
    WIDTH = 78
    NAME = "Label2"
ENDDDEFINE

DEFINE CLASS LogsLabel AS LABEL
    AUTOSIZE = .F.
    FONTBOLD = .T.
    FONTSIZE = 10
    ALIGNMENT = 0
    CAPTION = "Logs:"
    HEIGHT = 20
    LEFT = 35
    TOP = 72
    WIDTH = 75
    NAME = "Label3"
ENDDDEFINE

DEFINE CLASS BlanksLabel AS LABEL
    AUTOSIZE = .F.
    FONTBOLD = .T.
    FONTSIZE = 10
    ALIGNMENT = 0
    CAPTION = "Blanks & Artifacts:"
    HEIGHT = 20
    LEFT = 35
    TOP = 132
    WIDTH = 120
    NAME = "Label4"
ENDDDEFINE

DEFINE CLASS DataLabel AS LABEL
    AUTOSIZE = .F.
    FONTBOLD = .T.
    FONTSIZE = 10
    ALIGNMENT = 0
    CAPTION = "Data Files:"
    HEIGHT = 20
    LEFT = 35
    TOP = 192
    WIDTH = 65
    NAME = "Label5"
ENDDDEFINE

DEFINE CLASS CalLabel AS LABEL
    AUTOSIZE = .F.
    FONTBOLD = .T.
    FONTSIZE = 10
    ALIGNMENT = 0
    CAPTION = "Calibrations:"
    HEIGHT = 20
    LEFT = 35
    TOP = 288
    WIDTH = 80

```

```

NAME = "Label6"
ENDDDEFINE

DEFINE CLASS BallLabel AS LABEL
AUTOSIZE = .F.
FONTBOLD = .T.
FONTSIZE = 10
ALIGNMENT = 0
CAPTION = "Balance Equations:"
HEIGHT = 20
LEFT = 220
TOP = 288
WIDTH = 130
NAME = "Label7"
ENDDDEFINE

DEFINE CLASS ProcLabel AS LABEL
AUTOSIZE = .F.
FONTBOLD = .T.
FONTSIZE = 10
ALIGNMENT = 0
CAPTION = "Data Processing:"
HEIGHT = 20
LEFT = 35
TOP = 384
WIDTH = 110
NAME = "Label8"
ENDDDEFINE

DEFINE CLASS SiteBox AS LISTBOX
FONTBOLD = .T.
FONTSIZE = 8
VALUE = LogSites.site
CONTROLSOURCE = LogSites.site
TOP = 12
LEFT = 108
WIDTH = 75
HEIGHT = 50
MAXLENGTH = 5
SPECIALEFFECT = 0
COLUMNCOUNT = 1
NAME = "mSite"
SELSTART = 1
SELLENGTH = 5
SELECTONENTRY = .T.
ROWSOURCETYPE = 3
* ROWSOURCE = "select site from LogSites into cursor site ORDER BY site"
ROWSOURCETYPE = 2
ROWSOURCE = "LogSites"
* DisplayValue = 1
NUMBEROFELEMENTS = 180
INPUTMASK = "!!!!!"
READONLY = .F.
INCREMENTALSEARCH = .T.
PROCEDURE VALID
* m.mSite = site.site
m.mSite = LogSites.site
CLEAR EVENTS
* THISFORM.RELEASE
THISFORM.HIDE
WAIT CLEAR
IF SELECT("Flash")>0
SELECT Flash
USE
ENDIF
ENDPROC
ENDDDEFINE

DEFINE CLASS SeasonBox AS TEXTBOX
FONTBOLD = .T.
FONTSIZE = 9
ALIGNMENT = 1

```



```

VALUE = m.mSeason
SELSTART = 0
SELENGTH = 3
SELECTONENTRY = .T.
CONTROLSOURCE = "m.mSeason"
FORMAT = "!"
TOP = 24
LEFT = 310
HEIGHT = 25
WIDTH = 48
MAXLENGTH = 3
SPECIALEFFECT = 0
NAME = "mSeason"
PROCEDURE VALID
IF mSeason = CurrSeas
    mLogFile = mLogsPath + "logs.dbf"
ELSE
    mLogFile = mLogsPath + mSeason + "logs.dbf"
ENDIF
ENDPROC
ENDDDEFINE

*DEFINE CLASS FileLabel AS LABEL
* AUTOSIZE = .F.
* FONTBOLD = .T.
* FONTSIZE = 9
* ALIGNMENT = 1
* CAPTION = "File to Edit"
* HEIGHT = 90
* LEFT = 3
* TOP = 80
* WIDTH = 93
* NAME = "Label3"
*ENDDDEFINE

DEFINE CLASS EditLogs AS COMMANDBUTTON
TOP = 96
LEFT = 24
HEIGHT = 25
WIDTH = 85
FONTBOLD = .T.
FONTSIZE = 10
CANCEL = .F.
CAPTION = "Log\<Status"
DEFAULT = .T.
TERMINATEREAD = .F.
NAME = "Logs"
PROCEDURE CLICK
* IIF(len(msite)=0, m.msite=" ", m.msite = site.site)
m.mSite = LogSites.site
CLEAR EVENTS
THISFORM.RELEASE
WAIT CLEAR
medit="Logs"
ENDPROC
ENDDDEFINE

DEFINE CLASS EditFlow AS COMMANDBUTTON
TOP = 96
LEFT = 120
HEIGHT = 25
WIDTH = 85
FONTBOLD = .T.
FONTSIZE = 10
CANCEL = .F.
CAPTION = "Mag\<Flow"
DEFAULT = .F.
TERMINATEREAD = .F.
NAME = "Flows"
PROCEDURE CLICK
* IIF(len(msite)=0, m.msite=" ", m.msite = site.site)
m.mSite = LogSites.site

```

```

CLEAR EVENTS
THISFORM.RELEASE
WAIT CLEAR
medit="Flows"
ENDPROC
ENDDDEFINE

DEFINE CLASS EditVFlow AS COMMANDBUTTON
TOP = 96
LEFT = 216
HEIGHT = 25
WIDTH = 85
FONTBOLD = .T.
FONTSIZE = 10
CANCEL = .F.
CAPTION = "Vac Flow"
DEFAULT = .F.
TERMINATEREAD = .F.
NAME = "VFlows"
PROCEDURE CLICK
*       IIF(len(msite)=0, m.msite=" ", m.msite = site.site)
m.mSite = LogSites.site
CLEAR EVENTS
THISFORM.RELEASE
WAIT CLEAR
medit="VFlows"
ENDPROC
ENDDDEFINE

DEFINE CLASS QuitBtn1 AS COMMANDBUTTON
TOP = 96
LEFT = 312
HEIGHT = 25
WIDTH = 85
FONTBOLD = .T.
FONTSIZE = 10
CANCEL = .T.
CAPTION = "\<Quit"
TERMINATEREAD = .F.
NAME = "QuitBtn"
PROCEDURE CLICK
ON ERROR && restore system error handler
* SET DEFAULT TO &DefaultDir
* CLOSE DATABASES ALL
* SET SAFETY ON
POP KEY ALL
IF INLIST(mUser, "ashbaugh", "sengupta")
    CANCEL
ENDIF
QUIT
ENDPROC
ENDDDEFINE

DEFINE CLASS FBLevel AS COMMANDBUTTON
TOP = 156
LEFT = 24
HEIGHT = 25
WIDTH = 85
FONTBOLD = .T.
FONTSIZE = 10
CANCEL = .F.
CAPTION = "FB Level"
DEFAULT = .F.
TERMINATEREAD = .F.
NAME = "FBs"
PROCEDURE CLICK
m.mSite = LogSites.site
CLEAR EVENTS
THISFORM.RELEASE
WAIT CLEAR
medit="FBs"
ENDPROC

```

```

ENDDDEFINE

DEFINE CLASS EditIonsFB AS COMMANDBUTTON
    TOP = 156
    LEFT = 120
    HEIGHT = 25
    WIDTH = 85
    FONTBOLD = .T.
    FONTSIZE = 10
    CANCEL = .F.
    CAPTION = "IonsF\<B"
    DEFAULT = .F.
    TERMINATEREAD = .F.
    NAME = "IonsFB"
    PROCEDURE CLICK
*       IIF(len(msite)=0, m.msite=" ", m.msite = site.site)
    m.mSite = LogSites.site
    CLEAR EVENTS
    THISFORM.RELEASE
    WAIT CLEAR
    medit="FBions"
    ENDPROC
ENDDDEFINE

DEFINE CLASS CarbonCS AS COMMANDBUTTON
    TOP = 156
    LEFT = 216
    HEIGHT = 25
    WIDTH = 85
    FONTBOLD = .T.
    FONTSIZE = 10
    CANCEL = .F.
    CAPTION = "CarbonCS"
    DEFAULT = .F.
    TERMINATEREAD = .F.
    NAME = "CarbonCS"
    PROCEDURE CLICK
*       IIF(len(msite)=0, m.msite=" ", m.msite = site.site)
    m.mSite = LogSites.site
    CLEAR EVENTS
    THISFORM.RELEASE
    WAIT CLEAR
    medit="CSCarbon"
    ENDPROC
ENDDDEFINE

DEFINE CLASS ArtHist AS COMMANDBUTTON
    TOP = 156
    LEFT = 312
    HEIGHT = 25
    WIDTH = 85
    FONTBOLD = .T.
    FONTSIZE = 10
    CANCEL = .F.
    CAPTION = "ArtHist"
    TERMINATEREAD = .F.
    NAME = "ArtHist"
    PROCEDURE CLICK
*       IIF(len(msite)=0, m.msite=" ", m.msite = site.site)
    m.mSite = LogSites.site
    CLEAR EVENTS
    THISFORM.RELEASE
    WAIT CLEAR
    medit="ArtHist"
    ENDPROC
ENDDDEFINE

DEFINE CLASS EditIons AS COMMANDBUTTON
    TOP = 216
    LEFT = 24
    HEIGHT = 25
    WIDTH = 85

```

```

FONTBOLD = .T.
FONTSIZE = 10
CANCEL = .F.
CAPTION = "\<Ions"
DEFAULT = .F.
TERMINATEREAD = .F.
NAME = "Ions"
PROCEDURE CLICK
*   IIF(len(msite)=0, m.msite=" ", m.msite = site.site)
m.mSite = LogSites.site
CLEAR EVENTS
THISFORM.RELEASE
WAIT CLEAR
medit="Ions"
ENDPROC
ENDDDEFINE

DEFINE CLASS EditXRF AS COMMANDBUTTON
TOP = 216
LEFT = 120
HEIGHT = 25
WIDTH = 85
FONTBOLD = .T.
FONTSIZE = 10
CANCEL = .F.
CAPTION = "\<XRF"
DEFAULT = .F.
TERMINATEREAD = .F.
NAME = "Carbon"
PROCEDURE CLICK
*   IIF(len(msite)=0, m.msite=" ", m.msite = site.site)
m.mSite = LogSites.site
CLEAR EVENTS
THISFORM.RELEASE
WAIT CLEAR
medit="XRF"
ENDPROC
ENDDDEFINE

DEFINE CLASS EditWeights AS COMMANDBUTTON
TOP = 216
LEFT = 216
HEIGHT = 25
WIDTH = 85
FONTBOLD = .T.
FONTSIZE = 10
CANCEL = .F.
CAPTION = "\<Weights"
DEFAULT = .F.
TERMINATEREAD = .F.
NAME = "Weights"
PROCEDURE CLICK
*   IIF(len(msite)=0, m.msite=" ", m.msite = site.site)
m.mSite = LogSites.site
CLEAR EVENTS
THISFORM.RELEASE
WAIT CLEAR
medit="Weights"
ENDPROC
ENDDDEFINE

DEFINE CLASS EditProb AS COMMANDBUTTON
TOP = 216
LEFT = 312
HEIGHT = 25
WIDTH = 85
FONTBOLD = .T.
FONTSIZE = 10
CANCEL = .F.
CAPTION = "\<Problems"
DEFAULT = .F.
TERMINATEREAD = .F.

```

```

NAME = "Problems"
PROCEDURE CLICK
*   IIF(len(msite)=0, m.msite=" ", m.msite = site.site)
m.mSite = LogSites.site
CLEAR EVENTS
THISFORM.RELEASE
WAIT CLEAR
medit="Problems"
ENDPROC
ENDDDEFINE

DEFINE CLASS EditCarbon AS COMMANDBUTTON
TOP = 252
LEFT = 24
HEIGHT = 25
WIDTH = 85
FONTBOLD = .T.
FONTSIZE = 10
CANCEL = .F.
CAPTION = "\<Carbon"
DEFAULT = .F.
TERMINATEREAD = .F.
NAME = "Carbon"
PROCEDURE CLICK
*   IIF(len(msite)=0, m.msite=" ", m.msite = site.site)
m.mSite = LogSites.site
CLEAR EVENTS
THISFORM.RELEASE
WAIT CLEAR
medit="Carbon"
ENDPROC
ENDDDEFINE

DEFINE CLASS EditLaser AS COMMANDBUTTON
TOP = 252
LEFT = 120
HEIGHT = 25
WIDTH = 85
FONTBOLD = .T.
FONTSIZE = 10
CANCEL = .F.
CAPTION = "\<Laser"
DEFAULT = .F.
TERMINATEREAD = .F.
NAME = "Ions"
PROCEDURE CLICK
*   IIF(len(msite)=0, m.msite=" ", m.msite = site.site)
m.mSite = LogSites.site
CLEAR EVENTS
THISFORM.RELEASE
WAIT CLEAR
medit="Laser"
ENDPROC
ENDDDEFINE

DEFINE CLASS EditSO2 AS COMMANDBUTTON
TOP = 252
LEFT = 216
HEIGHT = 25
WIDTH = 85
FONTBOLD = .T.
FONTSIZE = 10
CANCEL = .F.
CAPTION = "SO\<2"
DEFAULT = .F.
TERMINATEREAD = .F.
NAME = "SO2"
PROCEDURE CLICK
*   IIF(len(msite)=0, m.msite=" ", m.msite = site.site)
m.mSite = LogSites.site
CLEAR EVENTS
THISFORM.RELEASE

```

```

        WAIT CLEAR
        medit="SO2"
    ENDPROC
ENDEDFINE

DEFINE CLASS EditFlash AS COMMANDBUTTON
    TOP = 252
    LEFT = 312
    HEIGHT = 25
    WIDTH = 85
    FONTBOLD = .T.
    FONTSIZE = 10
    CANCEL = .F.
    CAPTION = "Flash \<Data"
    DEFAULT = .F.
    TERMINATEREAD = .F.
    NAME = "Flash"
    PROCEDURE CLICK
*       IIF(len(msite)=0, m.msite=" ", m.msite = site.site)
    m.mSite = LogSites.site
    CLEAR EVENTS
    THISFORM.RELEASE
    WAIT CLEAR
    medit="Flash"
    ENDPROC
ENDEDFINE

DEFINE CLASS EditMag AS COMMANDBUTTON
    TOP = 312
    LEFT = 24
    HEIGHT = 25
    WIDTH = 85
    FONTBOLD = .T.
    FONTSIZE = 10
    CANCEL = .F.
    CAPTION = "\<Mag Cal"
    DEFAULT = .F.
    TERMINATEREAD = .F.
    NAME = "Mag"
    PROCEDURE CLICK
*       IIF(len(msite)=0, m.msite=" ", m.msite = site.site)
    m.mSite = LogSites.site
    CLEAR EVENTS
    THISFORM.RELEASE
    WAIT CLEAR
    medit="Mag"
    ENDPROC
ENDEDFINE

DEFINE CLASS EditVac AS COMMANDBUTTON
    TOP = 348
    LEFT = 24
    HEIGHT = 25
    WIDTH = 85
    FONTBOLD = .T.
    FONTSIZE = 10
    CANCEL = .F.
    CAPTION = "\<Vac Cal"
    TERMINATEREAD = .F.
    NAME = "Vac"
    PROCEDURE CLICK
*       IIF(len(msite)=0, m.msite=" ", m.msite = site.site)
    m.mSite = LogSites.site
    CLEAR EVENTS
    THISFORM.RELEASE
    WAIT CLEAR
    medit="Vac"
    ENDPROC
ENDEDFINE

DEFINE CLASS Bal25 AS COMMANDBUTTON
    TOP = 312

```

```

LEFT = 216
HEIGHT = 25
WIDTH = 85
FONTBOLD = .T.
FONTSIZE = 10
CANCEL = .F.
CAPTION = "C25 Eqn"
TERMINATEREAD = .F.
NAME = "Bal25"
PROCEDURE CLICK
*   IIF(len(msite)=0, m.msite=" ", m.msite = site.site)
m.mSite = LogSites.site
CLEAR EVENTS
THISFORM.RELEASE
WAIT CLEAR
medit="25"
ENDPROC
ENDEDEFINE

DEFINE CLASS Bal30 AS COMMANDBUTTON
TOP = 312
LEFT = 312
HEIGHT = 25
WIDTH = 85
FONTBOLD = .T.
FONTSIZE = 10
CANCEL = .F.
CAPTION = "C30 Eqn"
TERMINATEREAD = .F.
NAME = "Bal30"
PROCEDURE CLICK
*   IIF(len(msite)=0, m.msite=" ", m.msite = site.site)
m.mSite = LogSites.site
CLEAR EVENTS
THISFORM.RELEASE
WAIT CLEAR
medit="30 "
ENDPROC
ENDEDEFINE

DEFINE CLASS Bal30a AS COMMANDBUTTON
TOP = 348
LEFT = 216
HEIGHT = 25
WIDTH = 85
FONTBOLD = .T.
FONTSIZE = 10
CANCEL = .F.
CAPTION = "C30a Eqn"
TERMINATEREAD = .F.
NAME = "Bal30a"
PROCEDURE CLICK
*   IIF(len(msite)=0, m.msite=" ", m.msite = site.site)
m.mSite = LogSites.site
CLEAR EVENTS
THISFORM.RELEASE
WAIT CLEAR
medit="30a"
ENDPROC
ENDEDEFINE

DEFINE CLASS Bal31a AS COMMANDBUTTON
TOP = 348
LEFT = 312
HEIGHT = 25
WIDTH = 85
FONTBOLD = .T.
FONTSIZE = 10
CANCEL = .F.
CAPTION = "C31a Eqn"
TERMINATEREAD = .F.
NAME = "Bal31a"

```

```

PROCEDURE CLICK
*   IIF(len(msite)=0, m.msite=" ", m.msite = site.site)
  m.mSite = LogSites.site
  CLEAR EVENTS
  THISFORM.RELEASE
  WAIT CLEAR
  medit="31a"
ENDPROC
ENDDDEFINE

DEFINE CLASS FlowRSD AS COMMANDBUTTON
  TOP = 408
  LEFT = 24
  HEIGHT = 25
  WIDTH = 85
  FONTBOLD = .T.
  FONTSIZE = 10
  CANCEL = .F.
  CAPTION = "FlowRSD"
  DEFAULT = .F.
  TERMINATEREAD = .F.
  NAME = "FlowRSD"
  PROCEDURE CLICK
*   IIF(len(msite)=0, m.msite=" ", m.msite = site.site)
    m.mSite = LogSites.site
    CLEAR EVENTS
    THISFORM.RELEASE
    WAIT CLEAR
    medit="FlowRSD"
  ENDPROC
ENDDDEFINE

DEFINE CLASS BuildDBF AS COMMANDBUTTON
  TOP = 408
  LEFT = 120
  HEIGHT = 25
  WIDTH = 85
  FONTBOLD = .T.
  FONTSIZE = 10
  CANCEL = .F.
  CAPTION = "BuildDBF"
  TERMINATEREAD = .F.
  NAME = "BuildDBF"
  PROCEDURE CLICK
*   IIF(len(msite)=0, m.msite=" ", m.msite = site.site)
    m.mSite = LogSites.site
    CLEAR EVENTS
    THISFORM.RELEASE
    WAIT CLEAR
    medit="BuildDBF"
  ENDPROC
ENDDDEFINE

DEFINE CLASS EditHold AS COMMANDBUTTON
  TOP = 408
  LEFT = 216
  HEIGHT = 25
  WIDTH = 85
  FONTBOLD = .T.
  FONTSIZE = 10
  CANCEL = .F.
  CAPTION = "\<Hold Sites"
  DEFAULT = .F.
  TERMINATEREAD = .F.
  NAME = "Holds"
  PROCEDURE CLICK
*   IIF(len(msite)=0, m.msite=" ", m.msite = site.site)
    m.mSite = LogSites.site
    CLEAR EVENTS
    THISFORM.RELEASE
    WAIT CLEAR
    medit="Holds"

```



```

ENDPROC
ENDDDEFINE

DEFINE CLASS ClearQD AS COMMANDBUTTON
TOP = 408
LEFT = 312
HEIGHT = 25
WIDTH = 85
FONTBOLD = .T.
FONTSIZE = 10
CANCEL = .F.
CAPTION = "Clear QDs"
TERMINATEREAD = .F.
NAME = "ClearQD"
PROCEDURE CLICK
*       IIF(len(msite)=0, m.msite=" ", m.msite = site.site)
m.mSite = LogSites.site
CLEAR EVENTS
THISFORM.RELEASE
WAIT CLEAR
medit="ClearQD"
ENDPROC
ENDDDEFINE

DEFINE CLASS FlowRate AS COMMANDBUTTON
TOP = 444
LEFT = 24
HEIGHT = 25
WIDTH = 85
FONTBOLD = .T.
FONTSIZE = 10
CANCEL = .F.
CAPTION = "FlowRate"
DEFAULT = .F.
TERMINATEREAD = .F.
NAME = "FlowRate"
PROCEDURE CLICK
*       IIF(len(msite)=0, m.msite=" ", m.msite = site.site)
m.mSite = LogSites.site
CLEAR EVENTS
THISFORM.RELEASE
WAIT CLEAR
medit="FlowRate"
ENDPROC
ENDDDEFINE

DEFINE CLASS EditForm AS COMMANDBUTTON
TOP = 444
LEFT = 120
HEIGHT = 25
WIDTH = 85
FONTBOLD = .T.
FONTSIZE = 10
CANCEL = .F.
CAPTION = "\<Edit Form"
DEFAULT = .F.
TERMINATEREAD = .F.
NAME = "EditForm"
PROCEDURE CLICK
m.mSite = logsites.site
CLEAR EVENTS
THISFORM.RELEASE
WAIT CLEAR
medit = "Edit Form"
ENDPROC
ENDDDEFINE

DEFINE CLASS SwapCheck AS COMMANDBUTTON
TOP = 444
LEFT = 216
HEIGHT = 25
WIDTH = 85

```

```

FONTBOLD = .T.
FONTSIZE = 10
CANCEL = .F.
CAPTION = "SwapCheck"
TERMINATEREAD = .F.
NAME = "SwapCheck"
PROCEDURE CLICK
*      IIF(len(msite)=0, m.msite=" ", m.msite = site.site)
m.mSite = LogSites.site
CLEAR EVENTS
THISFORM.RELEASE
WAIT CLEAR
medit="SwapCheck"
ENDPROC
ENDDDEFINE

DEFINE CLASS FinalCheck AS COMMANDBUTTON
TOP = 444
LEFT = 312
HEIGHT = 25
WIDTH = 85
FONTBOLD = .T.
FONTSIZE = 10
CANCEL = .F.
CAPTION = "Final Check"
TERMINATEREAD = .F.
NAME = "FinalCheck"
PROCEDURE CLICK
*      IIF(len(msite)=0, m.msite=" ", m.msite = site.site)
m.mSite = LogSites.site
CLEAR EVENTS
THISFORM.RELEASE
WAIT CLEAR
medit="Check"
ENDPROC
ENDDDEFINE

```

D.2. ErrHand

```
PROCEDURE errhand
PARAMETER merror, MESS, mess1, mprog, mlineno
CLEAR
CLEAR WINDOWS

WAIT WINDOW TIMEOUT 4 'Error number: ' + LTRIM(STR(merror));
  + CHR(13)+ 'Error message: ' + MESS;
  + CHR(13)+ 'Line of code with error: ' + mess1;
  + CHR(13)+ 'Line number of error: ' + LTRIM(STR(mlineno));
  + CHR(13)+ 'Program with error: ' + mprog
IF merror=1
  nofile=.T.
ENDIF
*CLOSE DATABASES ALL
*ON ERROR  && restore system error handler
*CANCEL
*EXIT
RETURN
```

D.3. EditForm

PROCEDURE EditForm

PUBLIC notesite, notedate, mAffectedModule, mTDate, mrecnum, mSDate, mEDate, mOther,
mProblemDescription, mSwapDateChangedTo, mFlag

msite=logsites.site

STORE SYS(5)+SYS(2003) TO DefaultDir

* used for the list of problem descriptions
*USE J:\user\annexfront\sengupta\swap\problems.dbf IN 0
ProbDesc = mGETENV + "problems.dbf"
USE &ProbDesc IN 0 ALIAS problems

mproblem = problems.COMMENT
SELECT problems
INDEX ON COMMENT TAG COMMENT
GO TOP

*used for the list of flags
*USE J:\user\annexfront\sengupta\swap\flag.dbf IN 0
FlagFile = mGETENV + "flag.dbf"
USE &FlagFile IN 0 ALIAS FLAG

mFlag = FLAG.bi
SELECT FLAG
INDEX ON bi TAG bi
GO TOP

*used for the actual validation notes table where everything is inputed
*USE J:\user\annexfront\sengupta\swap\ValidationNotes.dbf IN 0
NotesFile = mGETENV + "ValidationNotes.dbf"
USE &NotesFile IN 0 ALIAS ValidationNotes

SELECT ValidationNotes
INDEX ON recnum TAG recnum
SET ORDER TO recnum
GO TOP

* creates all the variables where the input data is stored
mAffectedModule = ValidationNotes.MODULE
mTDate = DATE()
mSDate = ValidationNotes.startdate
mEDate = ValidationNotes.enddate
mSwapDateChangedTo = ValidationNotes.swapdate

mBadLog=.F.
frmInput = CREATEOBJECT('Form')
frmInput.CLOSABLE = .F.
frmInput.MOVABLE = .T.
frmInput.BORDERSTYLE = 3

frmInput.MOVE(0,0,425, 325)
frmInput.CAPTION = "Validate records for selected site/quarter"
frmInput.NAME = "EditForm"

*create all the objects on the form
frmInput.ADDOBJECT('Label1', 'SiteLabel')
frmInput.ADDOBJECT('Label11', 'SiteBox')

frmInput.ADDOBJECT('Label3', 'DateLabel')
frmInput.ADDOBJECT('EntryDate', 'DateBox')

frmInput.ADDOBJECT('Label4', 'SDateLabel')
frmInput.ADDOBJECT('StartDate', 'SDateBox')

frmInput.ADDOBJECT('Label5', 'EDateLabel')
frmInput.ADDOBJECT('EndDate', 'EDateBox')

```

frmInput.ADDOBJECT('Label6', 'ModuleLabel')
frmInput.ADDOBJECT('AffectedModule', 'AffectedModuleBox')

frmInput.ADDOBJECT('Label7', 'ProblemLabel')
frmInput.ADDOBJECT('ProblemDescription', 'ProblemDescriptionBox')

frmInput.ADDOBJECT('Label8', 'OtherLabel')
frmInput.ADDOBJECT('Other', 'OtherBox')

frmInput.ADDOBJECT('Label9', 'SwapLabel')
frmInput.ADDOBJECT('SwapDateChangedTo', 'SwapDateChangedToBox')

frmInput.ADDOBJECT('Label10', 'FlagLabel')
frmInput.ADDOBJECT('Flag', 'FlagBox')

frmInput.ADDOBJECT('Quitbtn', 'QuitBtn')
frmInput.ADDOBJECT('EnterBtn', 'EnterBtn')

frmInput.Label11.VISIBLE = .T.
frmInput.Label111.VISIBLE = .T.
frmInput.Label3.VISIBLE = .T.
frmInput.EntryDate.VISIBLE = .T.
frmInput.Label4.VISIBLE = .T.
frmInput.startdate.VISIBLE = .T.
frmInput.Label5.VISIBLE = .T.
frmInput.enddate.VISIBLE = .T.
frmInput.Label6.VISIBLE = .T.
frmInput.AffectedModule.VISIBLE = .T.
frmInput.Label7.VISIBLE = .T.
frmInput.ProblemDescription.VISIBLE = .T.
frmInput.Label8.VISIBLE = .T.
frmInput.OTHER.VISIBLE = .T.
frmInput.Label9.VISIBLE = .T.
frmInput.SwapDateChangedTo.VISIBLE = .T.
frmInput.Label10.VISIBLE = .T.
frmInput.FLAG.VISIBLE = .T.
frmInput.QuitBtn.VISIBLE = .T.
frmInput.EnterBtn.VISIBLE = .T.

frmInput.Label111.CAPTION = msite
frmInput.SHOW
*frmInput.REFRESH
READ EVENTS

*? msite, mLogfile, mFile
nofile = .F.

* Object definitions for display form follow
DEFINE CLASS SiteLabel AS LABEL
    AUTOSIZE = .F.
    FONTBOLD = .T.
    FONTSIZE = 9
    ALIGNMENT = 1
    CAPTION = "Site Name:  "&& + msite
    HEIGHT = 30
    LEFT = 35
    TOP = 24
    WIDTH = 93
    NAME = "Label11"
ENDDDEFINE

* Object definitions for display form follow
DEFINE CLASS SiteBox AS LABEL
    AUTOSIZE = .F.
    FONTBOLD = .T.
    FONTSIZE = 9
    ALIGNMENT = 1
    CAPTION = m.msites
    HEIGHT = 30
    LEFT = 130
    TOP = 24
    WIDTH = 60

```

```

NAME = "Label11"
ENDDDEFINE

*****
DEFINE CLASS DateLabel AS LABEL
AUTOSIZE = .F.
FONTBOLD = .T.
FONTSIZE = 9
ALIGNMENT = 1
CAPTION = "Today's Date:"
TOP = 70
LEFT = 18
HEIGHT = 25
WIDTH = 100
NAME = "Label3"
ENDDDEFINE

DEFINE CLASS DateBox AS TEXTBOX
FONTBOLD = .T.
FONTSIZE = 9
ALIGNMENT = 1
SELSTART = 0
SELLENGTH = 8
SELECTONENTRY = .T.
CONTROLSOURCE = "mTDate"
FORMAT = "!"
TOP = 70
LEFT = 144
HEIGHT = 25
WIDTH = 80
MAXLENGTH = 8
SPECIALEFFECT = 0
NAME = "mTDate"
ENDDDEFINE

*****
DEFINE CLASS ModuleLabel AS LABEL
AUTOSIZE = .F.
FONTBOLD = .T.
FONTSIZE = 9
ALIGNMENT = 1
CAPTION = "Affected Module:"
HEIGHT = 25
LEFT = 230
TOP = 70
WIDTH = 100
NAME = "Label6"
ENDDDEFINE

DEFINE CLASS AffectedModuleBox AS TEXTBOX
FONTBOLD = .T.
FONTSIZE = 9
ALIGNMENT = 1
VALUE = m.mAffectedModule
SELSTART = 0
SELLENGTH = 5
SELECTONENTRY = .T.
CONTROLSOURCE = "m.mAffectedModule"
FORMAT = "!"
TOP = 70
LEFT = 340
HEIGHT = 25
WIDTH = 60
MAXLENGTH = 5
SPECIALEFFECT = 0
NAME = "mAffectedModule"
ENDDDEFINE

*****
DEFINE CLASS SDateLabel AS LABEL
AUTOSIZE = .F.
FONTBOLD = .T.

```

```

    FONTSIZE = 9
    ALIGNMENT = 1
    CAPTION = "Start Date:"
    TOP = 110
    LEFT = 18
    HEIGHT = 25
    WIDTH = 100
    NAME = "Label4"
ENDDDEFINE

DEFINE CLASS SDateBox AS TEXTBOX
    FONTBOLD = .T.
    FONTSIZE = 9
    ALIGNMENT = 1
    SELSTART = 0
    SELLENGTH = 8
    SELECTONENTRY = .T.
    CONTROLSOURCE = "mSDate"
    FORMAT = "!"
    TOP = 110
    LEFT = 144
    HEIGHT = 25
    WIDTH = 80
    MAXLENGTH = 8
    SPECIALEFFECT = 0
    NAME = "mSDateBox"
ENDDDEFINE

*****
DEFINE CLASS ProblemLabel AS LABEL
    AUTOSIZE = .F.
    FONTBOLD = .T.
    FONTSIZE = 9
    ALIGNMENT = 1
    CAPTION = "Problem:"
    TOP = 150
    LEFT = 18
    HEIGHT = 25
    WIDTH = 100
    NAME = "Label7"
ENDDDEFINE

DEFINE CLASS ProblemDescriptionBox AS LISTBOX
    FONTBOLD = .T.
    FONTSIZE = 8
    VALUE = problems.COMMENT
    CONTROLSOURCE = problems.COMMENT
    TOP = 140
    LEFT = 144
    WIDTH = 250
    HEIGHT = 50
    MAXLENGTH = 60
    SPECIALEFFECT = 0
    COLUMNCOUNT = 1
    NAME = "ProblemDescription"
    SELSTART = 1
    SELLENGTH = 5
    SELECTONENTRY = .T.
    ROWSOURCETYPE = 3
    ROWSOURCE = "select comment from Problems into cursor comment ORDER BY comment"
    NUMBEROFELEMENTS = 62
    INPUTMASK = "!!!!!!!!!"
    READONLY = .F.
    INCREMENTALSEARCH = .T.
    PROCEDURE VALID
    m.ProblemDescription = problems.COMMENT
    CLEAR EVENTS
    THISFORM.RELEASE
    WAIT CLEAR
    IF SELECT("Flash")>0
        SELECT Flash
    USE

```

```

        ENDIF
    ENDPROC
ENDDDEFINE
*****

DEFINE CLASS EDateLabel AS LABEL
    AUTOSIZE = .F.
    FONTBOLD = .T.
    FONTSIZE = 9
    ALIGNMENT = 1
    CAPTION = "End Date:"
    TOP = 110
    LEFT = 230
    HEIGHT = 25
    WIDTH = 60
    NAME = "Label5"
ENDDDEFINE

DEFINE CLASS EDateBox AS TEXTBOX
    FONTBOLD = .T.
    FONTSIZE = 9
    ALIGNMENT = 1
    SELSTART = 0
    SELLENGTH = 8
    SELECTONENTRY = .T.
    CONTROLSOURCE = "mEDate"
    FORMAT = "!"
    TOP = 110
    LEFT = 300
    HEIGHT = 25
    WIDTH = 80
    MAXLENGTH = 8
    SPECIALEFFECT = 0
    NAME = "mEDateBox"
ENDDDEFINE

*****
DEFINE CLASS OtherLabel AS LABEL
    AUTOSIZE = .F.
    FONTBOLD = .T.
    FONTSIZE = 9
    ALIGNMENT = 1
    CAPTION = "Or Other:"
    TOP = 200
    LEFT = 18
    HEIGHT = 25
    WIDTH = 100
    NAME = "Label8"
ENDDDEFINE

DEFINE CLASS OtherBox AS TEXTBOX
    FONTBOLD = .T.
    FONTSIZE = 9
    ALIGNMENT = 1
    SELSTART = 0
    SELLENGTH = 60
    SELECTONENTRY = .T.
    CONTROLSOURCE = "mOther"
    TOP = 200
    LEFT = 144
    HEIGHT = 25
    WIDTH = 250
    MAXLENGTH = 60
    SPECIALEFFECT = 0
    NAME = "mOtherBox"
ENDDDEFINE

*****
DEFINE CLASS SwapLabel AS LABEL
    AUTOSIZE = .F.
    FONTBOLD = .T.
    FONTSIZE = 9

```



```

ALIGNMENT = 1
CAPTION = "Swap Date Changed To:"
TOP = 230
LEFT = 18
HEIGHT = 25
WIDTH = 150
NAME = "Label9"
ENDDDEFINE

DEFINE CLASS SwapDateChangedToBox AS TEXTBOX
    FONTBOLD = .T.
    FONTSIZE = 9
    ALIGNMENT = 1
    SELSTART = 0
    SELLENGTH = 8
    SELECTONENTRY = .T.
    CONTROLSOURCE = "mSwapDateChangedTo"
    FORMAT = "!"
    TOP = 230
    LEFT = 200
    HEIGHT = 25
    WIDTH = 70
    MAXLENGTH = 8
    SPECIALEFFECT = 0
    NAME = "mSwapDateChangedToBox"

ENDDDEFINE

*****
DEFINE CLASS FlagLabel AS LABEL
    AUTOSIZE = .F.
    FONTBOLD = .T.
    FONTSIZE = 9
    ALIGNMENT = 1
    CAPTION = "Flag:"
    TOP = 270
    LEFT = 18
    HEIGHT = 25
    WIDTH = 75
    NAME = "Label10"
ENDDDEFINE

DEFINE CLASS FlagBox AS LISTBOX
    FONTBOLD = .T.
    FONTSIZE = 8
    VALUE = FLAG.bi
    CONTROLSOURCE = FLAG.bi
    TOP = 270
    LEFT = 144
    WIDTH = 60
    HEIGHT = 50
    MAXLENGTH = 3
    SPECIALEFFECT = 0
    COLUMNCOUNT = 1
    NAME = "flag"
    SELSTART = 1
    SELLENGTH = 3
    SELECTONENTRY = .T.
    ROWSOURCETYPE = 3
    ROWSOURCE = "select bi from flag into cursor bi ORDER BY bi"
    NUMBEROFELEMENTS = 18
    INPUTMASK = "!!!"
    READONLY = .F.
    INCREMENTALSEARCH = .T.
    PROCEDURE VALID
    m.flag = FLAG.bi
    CLEAR EVENTS
    THISFORM.RELEASE
    WAIT CLEAR
    IF SELECT("Flash")>0
        SELECT Flash
        USE
    ENDIF

```

```

ENDPROC
ENDDDEFINE

DEFINE CLASS EnterBtn AS COMMANDBUTTON
    TOP = 270
    LEFT = 280
    HEIGHT = 25
    WIDTH = 50
    FONTBOLD = .T.
    FONTSIZE = 10
    CANCEL = .T.
    CAPTION = "\<Enter"
    TERMINATEREAD = .F.
    NAME = "Enterbtn"

    PROCEDURE CLICK
* the first 2 lines allows the clicked selection from the list to be saved
    mProblemDescription = COMMENT.COMMENT
    mFlag = bi.bi
*mOther = validationnotes.comment

    ON ERROR && restore system error handler
    SELECT ValidationNotes
* if there is an added comment, then do not add the clicked problem selection
*IF !EMPTY(TAG(mOther))
* INSERT INTO ValidationNotes (EntryDate, Site, Module, StartDate,;
EndDate, Comment, SwapDate, Flag);
VALUES( mTDate, m.mSite, mAffectedModule, mSDate, mEDate,;
mOther, mSwapDateChangedTo, mflag)
*ELSE
    INSERT INTO ValidationNotes ( EntryDate, site, MODULE, startdate,;
    enddate, COMMENT, swapdate, FLAG);
    VALUES(mTDate, m.msite, mAffectedModule, mSDate, mEDate,;
    mProblemDescription, mSwapDateChangedTo, mFlag)
*ENDIF
    SET SAFETY ON
*POP KEY ALL
*HIDE WINDOW output
*USE
ENDPROC
ENDDDEFINE

*****
DEFINE CLASS QuitBtn AS COMMANDBUTTON
    TOP = 270
    LEFT = 340
    HEIGHT = 25
    WIDTH = 50
    FONTBOLD = .T.
    FONTSIZE = 10
    CANCEL = .T.
    CAPTION = "\<Quit"
    TERMINATEREAD = .F.
    NAME = "Quitbtn"

    PROCEDURE CLICK
    ON ERROR && restore system error handler
    SET DEFAULT TO &DefaultDir
    SET SAFETY ON
    DEACTIVATE WINDOW ALL
    POP KEY ALL
*CANCEL
* RETURN TO MASTER
*RETURN
*CLEAR EVENTS
    THISFORM.RELEASE
    SET SAFETY OFF
    SELECT ValidationNotes
    USE
    SELECT FLAG
    USE

```

```
SELECT bi
USE
SELECT COMMENT
USE
SELECT problems
USE
RETURN TO MASTER
CANCEL
ENDPROC

ENDDFINE
```

D.4. QLogsErr

```
PROCEDURE QLOGSERR
PARAMETER mFile, mERROR, mMessage
SET ESCAPE OFF
WAIT WINDOW ;
  "Error on "+mFile + ". May be not on Disk or already open" ;
  + CHR(13) + CHR(10) + mMessage + ;
  + CHR(13) + "Hit ESC or ENTER to return to program" TIMEOUT 2
* Wait &mLogFile window Timeout 1
DO CASE
CASE LASTKEY() = 138
  DEACTIVATE WINDOW ALL
  CLOSE DATABASES ALL
  CANCEL
*   QUIT
CASE LASTKEY()=13 OR LASTKEY()=32 OR LASTKEY()=27
  mBadLog = .T.
  RETURN
OTHERWISE
  mBadLog = .T.
  RETRY          &&
ENDCASE

IF SELECT('logs')<>0
  SELECT logs
  USE
ENDIF
SET ESCAPE ON
RETURN
```

D.5. Errhand

```
PROCEDURE errhand
PARAMETER merror, mess, mess1, mprog, mlineno
CLEAR
? 'Error number: ' + LTRIM(STR(merror))
? 'Error message: ' + mess
? 'Line of code with error: ' + mess1
? 'Line number of error: ' + LTRIM(STR(mlineno))
? 'Program with error: ' + mprog
*CLOSE DATABASES ALL
WAIT
ON ERROR  && restore system error handler
CANCEL
```

D.6. EditLogs

```
PROCEDURE EditLogs
IF mSeason = CurrSeas
  EditFile = mLogsPath + "logs.dbf"
ELSE
  EditFile = mLogsPath + mSeason + "logs.dbf"
ENDIF
USE &EditFile IN 0 ALIAS logs
IF NOT mBadLog
  IF NOT nofile
    DEFINE WINDOW OUTPUT FROM 0,0 TO 40,165 TITLE 'Logs Status - Hit ESC to quit' ;
    FLOAT GROW ZOOM MDI FONT 'Arial', 8
    ACTIVATE WINDOW OUTPUT
    SELECT logs
    SET FILTER TO logs.site=m.mSite
    IF samdat>={^2004/09/01}
      BROWSE FIELDS site:W=.F., DAY:W=.F., ;
        BOX=RIGHT(DTOS(predate),2):W="F", area:W=.F., samdat:W=.F.,;
        aflnm:P="!!", bflnm:P="!!", cflnm:P="!!", dflnm:P="!!",
comments:90,;
        usela:P="!", uselb:P="!", uselc:P="!", useld:P="!", uselt:P="!";
        WINDOW OUTPUT
    ELSE
      BROWSE FIELDS site:W=.F., DAY:W=.F., ;
        BOX=RIGHT(DTOS(predate),2):W="F", area:W=.F., samdat:W=.F.,;
        aflnm:P="!!", bflnm:P="!!", cflnm:P="!!", dflnm:P="!!", area,
comments;
        WINDOW OUTPUT
    ENDIF
    HIDE WINDOW OUTPUT
  USE
ENDIF
ENDIF
```

D.7. EditFlows

```
PROCEDURE EditFlows
EditFile = mLogsPath + mSeason + "logs.dbf"
USE &EditFile IN 0 ALIAS logs
IF NOT mBadLog
  IF NOT nofile
    DEFINE WINDOW OUTPUT FROM 0,0 TO 40,165 TITLE 'Flows' ;
    CLOSE FLOAT GROW ZOOM MDI FONT 'Arial', 8
    ACTIVATE WINDOW OUTPUT
    SELECT logs
    SET FILTER TO logs.site=m.mSite
    IF samdat>{8/31/2004}
      IF RIGHT(mSite,1)="9"
        BROWSE FIELDS site:W=.F., DAY:W=.F., samdat:W=.F.,;
          tempcur:8, alet:8, blet:8, clet:8, dlet:8,;
          aghi:8, aghf:8, bghi:8, bghf:8, cghi:8, cghf:8,
df1gi:8, df1gf:8, comments:85,;
          usela:P="!", uselb:P="!", uselc:P="!", useld:P="!",
usel:~P="!";
        WINDOW OUTPUT
      ELSE
        BROWSE FIELDS site:W=.F., DAY:W=.F., samdat:W=.F.,;
          tempcur:8, alet:8, blet:8, clet:8, dlet:8,;
          aflmi:8, aflmf:8, bflmi:8, bflmf:8, cflmi:8, cflmf:8,
df1gi:8, df1gf:8, comments:85,;
          usela:P="!", uselb:P="!", uselc:P="!", useld:P="!",
usel:~P="!";
        WINDOW OUTPUT
      ENDIF
    ELSE
      IF RIGHT(mSite,1)="9"
        BROWSE FIELDS site:W=.F., DAY:W=.F., samdat:W=.F.,;
          tempcur:8, alet:8, blet:8, clet:8, dlet:8,;
          aghi:8, aghf:8, bghi:8, bghf:8, cghi:8, cghf:8,
df1gi:8, df1gf:8, comments:85;
        WINDOW OUTPUT
      ELSE
        BROWSE FIELDS site:W=.F., DAY:W=.F., samdat:W=.F.,;
          tempcur:8, alet:8, blet:8, clet:8, dlet:8,;
          aflmi:8, aflmf:8, bflmi:8, bflmf:8, cflmi:8, cflmf:8,
df1gi:8, df1gf:8, comments:85;
        WINDOW OUTPUT
      ENDIF
    ENDIF
  HIDE WINDOW OUTPUT
  USE
ENDIF
ENDIF
```

D.8. EditVFlows

```
PROCEDURE EditVFlows
EditFile = mLogsPath + mSeason + "logs.dbf"
USE &EditFile IN 0 ALIAS logs
IF NOT mBadLog
  IF NOT nofile
    DEFINE WINDOW OUTPUT FROM 0,0 TO 40,165 TITLE 'Flows' ;
    CLOSE FLOAT GROW ZOOM MDI FONT 'Arial', 9
    ACTIVATE WINDOW OUTPUT
    SELECT logs
    SET FILTER TO logs.site=m.mSite
    IF samdat>{8/31/2004}
      IF RIGHT(mSite,1)="9"
        BROWSE FIELDS site:W=.F., DAY:W=.F., samdat:W=.F.,;
          tempcur:8, alet:8, blet:8, clet:8, dlet:8,;
          aghi:8, aghf:8, bghi:8, bghf:8, cghi:8, cghf:8,
df1gi:8, df1gf:8, comments:85,;
          usela:P="!", uselb:P="!", uselc:P="!", useld:P="!",
usel:~P="!";
        WINDOW OUTPUT
      ELSE
        BROWSE FIELDS site:W=.F., DAY:W=.F., samdat:W=.F.,;
          tempcur:8, alet:8, blet:8, clet:8, dlet:8,;
          aghi:8, aghf:8, bghi:8, bghf:8, cghi:8, cghf:8,
df1gi:8, df1gf:8, comments:85,;
          usela:P="!", uselb:P="!", uselc:P="!", useld:P="!",
usel:~P="!";
        WINDOW OUTPUT
      ENDIF
    ELSE
      IF RIGHT(mSite,1)="9"
        BROWSE FIELDS site:W=.F., DAY:W=.F., samdat:W=.F.,;
          tempcur:8, alet:8, blet:8, clet:8, dlet:8,;
          aghi:8, aghf:8, bghi:8, bghf:8, cghi:8, cghf:8,
df1gi:8, df1gf:8, comments:85;
        WINDOW OUTPUT
      ELSE
        BROWSE FIELDS site:W=.F., DAY:W=.F., samdat:W=.F.,;
          tempcur:8, alet:8, blet:8, clet:8, dlet:8,;
          aghi:8, aghf:8, bghi:8, bghf:8, cghi:8, cghf:8,
df1mi:8, df1mf:8, comments:85;
        WINDOW OUTPUT
      ENDIF
    ENDIF
  HIDE WINDOW OUTPUT
  USE
ENDIF
ENDIF
```


D.9. EditWeights

```
PROCEDURE EditWeights
IF mSeason = CurrSeas
    EditFile = mWtsPath + "weights.dbf"
ELSE
    EditFile = mWtsPath + mSeason + "wts.dbf"
ENDIF

USE &EditFile IN 0 ALIAS weights
IF NOT mBadLog
    IF NOT nofile
        SELECT weights
        SET FILTER TO weights.site=m.mSite
        mTmpWt = mGETENV + "TmpWt.dbf"
        COPY TO &mTmpWt
        USE &mTmpWt IN 0 ALIAS Wts
        SELECT Wts
        DO calcwts
        SELECT Wts
        INDEX ON site+DTC(samdat)+CHAN TAG sitedate
        SELECT weights
        GO TOP
        SET RELATION TO site+DTC(samdat)+CHAN INTO Wts ADDITIVE
        DEFINE WINDOW OUTPUT FROM 0,0 TO 40,145 ;
            TITLE 'Weights - Press F1 to create XLS file in M:\IMPROVE\Temp' ;
            CLOSE FLOAT GROW ZOOM MDI FONT 'Arial', 9
        ACTIVATE WINDOW OUTPUT
        PUSH KEY
        ON KEY LABEL F1 DO WTout
        DO CASE
            CASE weights.samdat<{09/01/2001}
                BROWSE FIELDS site:W=.F., samdat, CHAN, STATUS:P="!!", ;
                    Cahn25wt, Prewght:W=.F., Postwght, DIFF=Postwght-
Prewght:W=.F., ;
                    prindate:W=.F., printime:W=.F., ;
                    prbalance:W=.F., prname:W=.F., pobalance:W=.F.,
poname:W=.F. ;
                    LOCK -4 ;
                    WINDOW OUTPUT
            CASE weights.samdat<{06/01/2002}
                BROWSE FIELDS site:W=.F., samdat, CHAN, STATUS:P="!!", ;
                    Cahn25wt, Post30a, Prewght:W=.F., Postwght:W=.F.,
DIFF=Postwght-Prewght:W=.F., ;
                    prindate:W=.F., printime:W=.F., ;
                    prbalance:W=.F., prname:W=.F., pobalance:W=.F.,
poname:W=.F. ;
                    LOCK -4 ;
                    WINDOW OUTPUT
            OTHERWISE
                BROWSE FIELDS site:W=.F., samdat, CHAN, STATUS:P="!!", ;
                    prebalwt, pstbalwt, Wts.Prewght:W=.F., Wts.Postwght:W=.F., ;
                    Wts.DIFF:W=.F., prindate:W=.F., printime:W=.F., ;
                    prbalance:W=.F., prname:W=.F., pobalance:W=.F.,
poname:W=.F. ;
                    LOCK -4 ;
                    WINDOW OUTPUT
        ENDCASE
        HIDE WINDOW OUTPUT
        SELECT weights
        USE
        SELECT Wts
        USE
        POP KEY
    ENDIF
ENDIF

PROCEDURE WTout
outname = mPathtemp + ALLTRIM(wts.site) + mSeason + "WT.xls"
DO CASE
```

```

CASE weights.samdat<{09/01/2001}
  COPY TO &outname XL5 FIELDS site, samdat, CHAN, STATUS, ;
    Cahn25wt, PSTBAL=postwght, Prewght, Postwght, DIFF=Postwght-Prewght, ;
    prindate, printime, ;
    prbalance, prname, pobalance, poname
CASE weights.samdat<{06/01/2002}
  COPY TO &outname XL5 FIELDS site, samdat, CHAN, STATUS, ;
    Cahn25wt, Post30a, Prewght, Postwght, DIFF=Postwght-Prewght, ;
    prindate, printime, ;
    prbalance, prname, pobalance, poname
OTHERWISE
  COPY TO &outname XL5 FIELDS site, samdat, CHAN, STATUS, ;
    prebalwt, pstbalwt, Wts.Prewght, Wts.Postwght, ;
    Wts.DIFF, prindate, printime, ;
    prbalance, prname, pobalance, poname
ENDCASE
RETURN

```

D.10. CalcWts

PROCEDURE CalcWts

* Recalculates prewgt, postwgt, and diff in the WTS file for selected quarter

PUBLIC mPrewght, mPstwght

mDefPath = SYS(5)+SYS(2003)

SET DEFAULT TO &mPathTemp

*Delete temporary files if they exist

IF FILE('tbal25.dbf')

DELETE FILE tbal25.dbf

ENDIF

IF FILE('tbal30.dbf')

DELETE FILE tbal30.dbf

ENDIF

IF FILE('tbal30a.dbf')

DELETE FILE tbal30a.dbf

ENDIF

IF FILE('tbal31a.dbf')

DELETE FILE tbal31a.dbf

ENDIF

USE "u:\improve\support\baleq25.dbf" IN 0 Alias bal

SELECT bal

COPY ALL TO tbal25

USE "u:\improve\support\baleq30.dbf"

COPY ALL TO tbal30

USE "u:\improve\support\baleq30a.dbf"

COPY ALL TO tbal30a

USE "u:\improve\support\baleq31a.dbf"

COPY ALL TO tbal31a

USE

USE tbal25 IN 0 ALIAS bal25

USE tbal30 IN 0 ALIAS bal30

USE tbal30a IN 0 ALIAS bal30a

USE tbal31a IN 0 ALIAS bal31a

SELECT wts

lPrewght = .F.

FOR n = 1 TO FCOUNT("wts")

IF FIELD(n) = "PREWGHT"

lPrewght = .T.

ENDIF

ENDFOR

IF NOT lPrewght

ALTER TABLE wts ADD COLUMN Prewght N(6,3)

ALTER TABLE wts ADD COLUMN Postwght N(6,3)

ALTER TABLE wts ADD COLUMN Diff N(6,3)

REPLACE ALL Prewght WITH Prebalwt

REPLACE ALL Postwght WITH Pstbalwt

REPLACE ALL Diff WITH Postwght-Prewght

ENDIF

GO TOP

DO WHILE NOT EOF('wts')

DO GETPREWT

DO GETPSTWT

IF (wts.prewght <> 0 .and. (wts.prewght <> mPrewght)) .or. ;

(wts.postwght <> 0 .and. (wts.postwght <> mPstwght)) .or. ;

wts.diff <> mPstwght - mPrewght .and. ;

wts.postwght<>0

REPLACE wts.prewght WITH mPrewght

REPLACE wts.postwght WITH mPstwght

REPLACE wts.diff WITH mPstwght - mPrewght

```

        ENDIF
        skip in wts
    ENDDO

* Clean up files
SELECT bal25
USE
SELECT bal30
USE
SELECT bal30a
USE
SELECT bal31a
USE
DELETE FILE tbal25.dbf
DELETE FILE tbal30.dbf
DELETE FILE tbal30a.dbf
DELETE FILE tbal31a.dbf

RETURN

*****
Procedure GETPREWT
    SELECT wts
    PUBLIC mprecaldate
    mPRINDATE = wts.prindate
    mPRINTIME = wts.printime
    SET DELETED ON
    IF FIELD(1,"wts") = "SITE"
        STORE "Prebalwt" to Fprebalwt    && Select the balance prewt regardless of
        field name
    ELSE
        STORE FIELD(8,"wts") to Fprebalwt
    ENDIF
    mprebalwt = &Fprebalwt
*   IF mseason="B02"
*       mprebalwt = wts.cahn25wt
*   ELSE
*       mprebalwt = wts.prebalwt
*   ENDIF
    Do Case
    Case wts.prbalance = '31' .or. mprebalwt = 0 .or. wts.SAMDAT < {^1996-03-01}
        && Use the PREWGHT for all before A96
        mPREWGHT = wts.prewght
            && This is because of the 20 mg addition stuff on the 25
        mprecaldate = wts.prindate
    Case wts.prbalance = '25'
        SELECT bal25
        Locate For BEGDAT > mPRINDATE
        If Found()
            Skip - 1
        Else
            Goto Bottom
        Endif
        mTAREINT = tareint
        mTARESLP = tareslp
        mPREWGHT=round(mTAREINT+(mprebalwt*mTARESLP),3)
        mprecaldate = begdat
    Case wts.prbalance = '30'
        Sele bal30
        Locate For BEGDAT > mPRINDATE
        If Found()
            Skip - 1
        Else
            Goto Bottom
        Endif
        mTAREINT = TAREINT
        mTARESLP = TARESLP
        mPREWGHT=round(mTAREINT+(mprebalwt*mTARESLP),3)
        mprecaldate = begdat
    Case wts.prbalance = '3A'
        Sele bal30a
        Locate For BEGDAT > mPRINDATE

```

```

    If Found()
        Skip - 1
    Else
        Goto Bottom
    Endif
    mTAREINT = TAREINT
    mTARESLP = TARESLP
    mPREWGHT=round(mTAREINT+(mprebalwt*mTARESLP),3)
    mprecaldate = begdat
Case wts.prbalance = '1A'
    Sele bal31a
    Locate For BEGDAT > mPRINDATE
    If Found()
        Skip - 1
    Else
        Goto Bottom
    Endif
    mTAREINT = TAREINT
    mTARESLP = TARESLP
    mPREWGHT=round(mTAREINT+(mprebalwt*mTARESLP),3)
    mprecaldate = begdat
OTHERWISE
    mPREWGHT=mprebalwt
    mprecaldate = begdat
Endcase
SELECT wts
Return
*****
Procedure GETPSTWT
SELECT wts
PUBLIC mpstcaldate
mPOINDATE = wts.poindate
mPOINTIME = wts.pointime
Set Dele on
IF FIELD(1,"wts") = "SITE"
    STORE "Pstbalwt" to Fpstbalwt    && Select the balance pstwt regardless of
field name
ELSE
    STORE FIELD(10,"wts") to Fpstbalwt
ENDIF
mpstbalwt = &Fpstbalwt
Do Case
Case wts.pobalance = '25'
    SELECT bal25
    Locate For BEGDAT > mPoINDATE
    If Found()
        Skip - 1
    Else
        Goto Bottom
    Endif
    mTAREINT = tareint
    mTARESLP = tareslp
    mPstwght =ROUND(mTAREINT+(mpstbalwt*mTARESLP),3)
    mpstcaldate = begdat
Case wts.pobalance = '30'
    Sele bal30
    Locate For BEGDAT > mPOINDATE
    If Found()
        Skip - 1
    Else
        Goto Bottom
    Endif
    mTAREINT = TAREINT
    mTARESLP = TARESLP
    mPstwght =ROUND(mTAREINT+(mpstbalwt*mTARESLP),3)
    mpstcaldate = begdat
Case wts.pobalance = '3A'
    Sele bal30a
    Locate For BEGDAT > mPOINDATE
    If Found()
        Skip - 1
    Else

```

```

        Goto Bottom
    Endif
    mTAREINT = TAREINT
    mTARESLP = TARESLP
    mPstwght =ROUND (mTAREINT+(mpstbalwt*mTARESLP),3)
    mpstcaldate = begdat
    Case wts.pobalance = '1A'
        Sele bal31a
        Locate For BEGDAT > mPOINDATE
        If Found()
            Skip - 1
        Else
            Goto Bottom
        Endif
        mTAREINT = TAREINT
        mTARESLP = TARESLP
        mPstwght =ROUND (mTAREINT+(mpstbalwt*mTARESLP),3)
        mpstcaldate = begdat
    Otherwise
        &&POBALANCE = 31 and POBALANCE =
        ' '
        mPstwght = wts.postwght
        && This is because of the 20 mg addition
        stuff on the 25
        mpstcaldate = wts.poindate
    Endcase
Return

```

D.11. FBLevel2

```
PUBLIC numrec

*CLOSE DATABASES all
IF FILE("fblist.dbf")
    DELETE FILE fblist.DBF
ENDIF
IF mSeason = CurrSeas
    EditFile = mLogsPath + "logs.dbf"
ELSE
    EditFile = mLogsPath + mSeason + "logs.dbf"
ENDIF
USE &EditFile IN 0 ALIAS logs
IF NOT mBadLog

    SELECT DISTINCT MONTH(logs.samdat) AS MONTH, COUNT(logs.fbchan), YEAR(logs.samdat)
    as YEAR;
    FROM logs;
    WHERE logs.fbchan = ( " " );
    GROUP BY 1, 3;
    INTO CURSOR Fb_total
    INDEX ON MONTH TAG MONTH
*BROWSE LAST
** IF SELECT("fbcount")<>0
**     SELECT fbcount
**     USE
** ENDIF
** DELETE FILE fbcount.DBF
    SELECT DISTINCT MONTH(logs.samdat) AS MONTH, logs.fbchan, ;
    COUNT(logs.fbchan), YEAR(logs.samdat) as year ;
    FROM ;
    logs;
    WHERE logs.fbchan <> ( " " );
    GROUP BY 1, logs.fbchan, 4;
    INTO TABLE fbcount
    SELECT fbcount
    ALTER TABLE fbcount ALTER COLUMN cnt_fbchan fieldtype(6,2)
    SET RELATION TO MONTH INTO Fb_total ADDITIVE
    REPLACE ALL cnt_fbchan WITH cnt_fbchan/Fb_total.cnt_fbchan*100
    SET RELATION TO
    REPLACE ALL month WITH 0 FOR month=12
    INDEX ON month TAG month
*BROWSE LAST
    DO GENXTAB WITH 'Fb',.F.,.T.,.T.,1,2,3,.F. && There should be exactly three columns
    in result, in order to execute this command.
    REPLACE ALL month WITH 12 FOR month=0
    SET RELATION TO MONTH INTO Fb_total ADDITIVE
*BROWSE LAST
** SELECT logs
** USE
    SELECT fb
    REPORT FORM fb PREVIEW
* DEFINE WINDOW OUTPUT FROM 0,0 TO 45,100 ;
    TITLE 'Field Blank Report - Press any key to continue' ;
    CLOSE FLOAT GROW ZOOM MDI FONT 'Arial', 9
* ACTIVATE WINDOW output
* REPORT FORM fb IN WINDOW output
* SET ESCAPE off
* WAIT ""
* SET ESCAPE on
* HIDE WINDOW output
ENDIF
RETURN
```

D.12. GenXTab

```
*:*****
*:
*: Procedure file: C:\FOXPRO2\GENXTAB\GENXTAB.PRG
*:
*:       System: GENXTAB
*:       Author: Microsoft Corp.
*:       Copyright (c) 1994, Microsoft Corp.
*:       Last modified: 6/30/94      10:17
*:
*: Procs & Fncts: APPERROR
*:                : ESC_PROC
*:                : JUSTFNAME()
*:                : JUSTSTEM()
*:                : BAILOUT
*:                : DEFAULTTEXT()
*:                : ALERT
*:                : ACTTHERM
*:                : UPDTHERM
*:                : MAPNAME()
*:                : DEACTTHERMO
*:                : FORCEEXT
*:                : JUSTPATH
*:                : ADDBS
*:                : MAKESTRG
*:
*:       Calls: APPERROR      (procedure in GENXTAB.PRG)
*:              : ESC_PROC    (procedure in GENXTAB.PRG)
*:              : JUSTFNAME() (function in GENXTAB.PRG)
*:              : JUSTSTEM()  (function in GENXTAB.PRG)
*:              : BAILOUT     (procedure in GENXTAB.PRG)
*:              : DEFAULTTEXT() (function in GENXTAB.PRG)
*:              : ALERT       (procedure in GENXTAB.PRG)
*:              : ACTTHERM    (procedure in GENXTAB.PRG)
*:              : UPDTHERM    (procedure in GENXTAB.PRG)
*:              : MAPNAME()   (function in GENXTAB.PRG)
*:              : DEACTTHERMO (procedure in GENXTAB.PRG)
*:
*:       Uses: XTABTEMP.DBF
*:
*:*****
* Notes: This program is intended to be called by RQBE or a program
*        generated by RQBE.  On entry, a table should be open in the
*        current work area, and it should contain at most one record
*        for each cell in a cross-tabulation.  This table *must* be in
*        row order, or you will receive an "unexpected end of file"
*        error when you run GENXTAB.
*
*        The rowfld field in each record becomes the y-axis (rows) for
*        a cross-tab and the colfld field becomes the x-axis (columns)
*        The actual cross-tab results are saved to the database name
*        specified by "outfname."
*
*        The basic strategy goes like this.  Produce an empty database
*        with one field/column for each unique value of input field
*        colfld, plus one additional field for input field rowfld values.
*        This process determines the column headings in the database.
*        Next fill in the rows, but only for the first field in the output
*        database--the one that contains values for input field rowfld.
*        At this point, we have column headings "across the top"
*        and row identifiers "down the side."  Finally, look up
*        the cell values for the row/column intersections and put
*        them into the output database.
*
*
* Calling example:
*       DO genxtab WITH 'XTAB.DBF',.T.,.T.,.T.,1,2,5,.T.
```



```

*
*      This command causes GENXTAB to write the output database to
*      'XTAB.DBF'.  However, XTAB.DBF will be deleted and the output
*      stored to a cursor called XTAB.  The input database will be closed
*      at the conclusion of the program.  The rows in XTAB.DBF will
*      contain the unique values of field 1 in the database that is
*      selected when GENXTAB is called, the columns will contain
*      unique values of field 2 in the input database, and the
*      cell values will come from field 5 in the input database.
*      The thermometer will be shown.  A total field will be created.
*
*****
PARAMETERS outfname,      ;
        cursonly,        ;
        closeinput,     ;
        showtherm,      ;
        rowfld,         ;
        colfld,         ;
        cellfld,        ;
        xfoot

PRIVATE ALL

m.g_dlgface   = IIF(_MAC,"Geneva","MS Sans Serif")
m.g_dlgsize   = IIF(_MAC,10,8.000)
m.g_dlgstyle  = IIF(_MAC,"","B")

EXTERNAL ARRAY coluniq
EXTERNAL ARRAY colcnt

* -----
* Do opening housekeeping
* -----
IF SET("TALK") = "ON"
    SET TALK OFF
    xtalk_stat = "ON"
ELSE
    xtalk_stat = "OFF"
ENDIF
xsafe_stat = SET("SAFETY")
SET SAFETY OFF
xesc_stat = SET("ESCAPE")
SET ESCAPE ON

m.mfieldsto = SET("FIELDS",1)
m.fields = SET("FIELDS")
SET FIELDS TO
SET FIELDS OFF
m.udfparms = SET("UDFPARMS")
SET UDFPARMS TO VALUE

#if "MAC" $ UPPER(VERSION(1))
    IF _MAC
        m.mmacdesk = SET("MACDESKTOP")
        SET MACDESKTOP ON
    ENDIF
#endif
in_esc = ON('ESCAPE')
in_err = ON('ERROR')

ON ERROR DO apperror WITH PROGRAM(),MESSAGE(),MESSAGE(1),LINENO(),ERROR()
ON ESCAPE DO esc_proc

* -----
* Set default values for parameters
* -----
IF PARAMETERS() < 1
    m.outfname = 'XTAB.DBF'
ENDIF
IF PARAMETERS() < 2
    * Default to creating the same kind of output as we got as input.

```

```

* If the input "database" is a cursor, make the output a cursor.
* If the input "database" is an actual database, make the output a table.
cname = justfname(DBF())
DO CASE
CASE EMPTY(cname)    && create a table if nothing is currently selected
    cursonly = .F.
CASE ISDIGIT(LEFT(cname,1))
    cursonly = .T.
OTHERWISE
    cursonly = .F.
ENDCASE
ENDIF
IF PARAMETERS() < 3
    * Close the input database
    closeinput = .T.
ENDIF
IF PARAMETERS() < 4
    * show the thermometer
    showtherm = .T.
ENDIF
IF PARAMETERS() < 5
    * the field position in the input database for the crosstab rows
    m.rowfld = 1
ENDIF
IF PARAMETERS() < 6
    * the field position in the input database for the crosstab columns
    m.colfld = 2
ENDIF
IF PARAMETERS() < 7
    * the field position in the input database for the crosstab cells
    m.cellfld = 3
ENDIF
IF PARAMETERS() < 8
    * Create a total field?
    m.xfoot = .F.
ENDIF

* Define characters that are not allowed in field names
m.badchars = '•, f, „... † ‡ ^ % &lt; € Ž • • „ ' ' \" \" • -- ~™ § | ¢ £ ¢ ¥ / \ , - = : ; { } [ ] ! @ # $ % ^ & * . < > ( ) ? ' + ;
' + | € > α • ž Ÿ | § " © * « - - ® - ° ± ² ³ ´ µ ¶ · ¸ ¹ º » ¼ ½ ¾ ¿ À Á Â Ã Ä Å Æ Ç È É Ê Ë Ì Í Î Ï ' + ;
' ð Ñ Ò Ó Ô Õ Ö × Ø Ù Ú Û Ü Ý Þ ß à á â ã ä å æ ç è é ê ë ì í î ï ð ñ ò ó ô õ ö ÷ ø ù ú û ü ý þ ' + CHR(39)
* Map European characters to these
m.stdascii = 'ueaaaaceeeiiiAaEaAooouuyouaiounN'

IF !showtherm
    m.recthresh = 100000000    && don't show the thermometer
ELSE
    m.recthresh = 1            && show it if more than this many input records
ENDIF
m.g_thermwidth = 0            && Thermometer width

m.outfname = removequotes(m.outfname)
m.outstem = juststem(m.outfname)

* -----
* Construct the output database structure
* -----

m.dbfname = ALIAS()
m.dbfstem = Juststem(m.dbfname)

therm_on = (RECCOUNT() >= recthresh)

* Select one, if no database is open in the current workarea
m.ok = .F.
DO WHILE NOT ok
    DO CASE
    CASE EMPTY(m.dbfname)
        m.dbfname = GETFILE('DBF','Please locate the input database')
        m.dbfstem = juststem(m.dbfname)
        IF EMPTY(m.dbfname)

```

```

        * User canceled out of dialog, so quit the program
        DO bailout WITH .T.
    ENDIF
CASE FULLPATH(defaulttext(m.dbfname,'DBF')) == ;
    FULLPATH(defaulttext(m.outfname,'DBF'))
    SET CURSOR OFF
    WAIT WINDOW "The input and output databases must be different."
    SET CURSOR ON
    m.dbfname = ''
OTHERWISE
    IF USED(m.dbfstem)
        SELECT (m.dbfstem)
    ELSE
        SELECT 0
        USE (m.dbfname) ALIAS (m.dbfstem)
    ENDIF
    IF FCOUNT() < 3
        DO alert WITH "Crosstab input databases require; at least three fields"
        m.dbfname = ''
    ELSE
        ok = .T.
    ENDIF
ENDCASE
ENDDO

IF RECCOUNT() = 0
    DO alert WITH "Cannot prepare crosstab on empty database"
    DO bailout WITH .T.
ENDIF

* Gather information on the currently selected database fields
DIMENSION inpfields[FCOUNT(),4]
m.numflds = AFIELDS(inpfields)

* Map the physical input database field to logical field positions
m.rowfldname = inpfields[m.rowfld,1]
m.colfldname = inpfields[m.colfld,1]
m.cellfldname = inpfields[m.cellfld,1]

* None of these fields are allowed to be memo fields
IF inpfields[1,2] $ 'MGP'
    DO alert WITH "The crosstab row field in the input; database cannot be a memo,
    general or picture field."
    DO bailout WITH .T.
ENDIF
IF inpfields[2,2] $ 'MGP'
    DO alert WITH "The crosstab column field in the input; database cannot be a memo,
    general or picture field."
    DO bailout WITH .T.
ENDIF
IF inpfields[3,2] $ 'MGP'
    DO alert WITH "The crosstab cell field in the input; database cannot be a memo,
    general or picture field."
    DO bailout WITH .T.
ENDIF

IF therm_on
    DO acttherm WITH "Generating cross-tabulation ..."
    DO updtherm WITH 5
ENDIF

* Set the mouse off to avoid flicker on some systems
SET MOUSE OFF

* Count the number of columns we need to create the cross tab.
* This step could be combined with the following one so that there
* would only be one SELECT operation performed. It is coded in this
* way to avoid running out of memory if there are an unexpectedly
* large number of unique values of field 2 in the input database.
SELECT COUNT(DISTINCT &colfldname) FROM (m.dbfname) INTO ARRAY colcnt
DO CASE

```

```

CASE colcnt[1] > 254
  DO alert WITH "Too many unique values of "+PROPER(m.colfldname);
  + ".; The maximum is 254."
  DO bailout WITH .T.
CASE colcnt[1] = 0
  DO alert WITH "No columns found."
  DO bailout WITH .T.
ENDCASE

* Get the number of decimal places in numeric fields
* and extract all the unique values of colfldname
IF inpflds[m.colfld,2] $ 'NF'   && numeric or floating field
  m.cdec = inpflds[m.colfld,4]
  * Handle numbers separately to preserve correct sort order
  SELECT DISTINCT &colfldname ;
  FROM (m.dbfname) INTO ARRAY colunig
  FOR i = 1 TO ALEN(colunig)
    colunig[i] = mapname(colunig[i],m.cdec)
  ENDFOR
ELSE   && non-numeric field
  m.cdec = 0
  * Create an array to hold the output database fields.
  SELECT DISTINCT mapname(&colfldname,m.cdec) ;
  FROM (m.dbfname) INTO ARRAY colunig
ENDIF

IF therm_on
  DO updttherm WITH 15
ENDIF

* The field type, length and decimals in the output array control the
* cross-tab cells
IF !m.xfoot
  DIMENSION outarray[ALEN(colunig)+1,4]
ELSE
  DIMENSION outarray[ALEN(colunig)+2,4]
ENDIF

* Field 1 in the output DBF holds the unique values of the row input field.
* It is handled separately from the other fields, which take their names
* from input database colfld and their parameters (e.g., length) from
* input database cellfld.

outarray[1,1] = mapname(inpflds[1,1])
outarray[1,2] = inpflds[1,2]
outarray[1,3] = inpflds[1,3]
outarray[1,4] = inpflds[1,4]

FOR i = 2 TO ALEN(colunig) + 1
  outarray[i,1] = mapname(colunig[i-1],m.cdec)
  outarray[i,2] = inpflds[3,2]           && field type
  outarray[i,3] = inpflds[3,3]           && field length
  outarray[i,4] = inpflds[3,4]           && decimals
ENDFOR

* Create a field for the cross-footing, if that option was selected
IF m.xfoot
  outarray[ALEN(colunig)+2,1] = 'XTOTALS'
  outarray[ALEN(colunig)+2,2] = inpflds[3,2]
  outarray[ALEN(colunig)+2,3] = inpflds[3,3]
  outarray[ALEN(colunig)+2,4] = inpflds[3,4]
ENDIF

* Make sure that the output file is not already in use somewhere
IF USED(m.outstem)
  SELECT (m.outstem)
  USE
ENDIF

IF !cursonly
  CREATE TABLE (outfname) FROM ARRAY outarray
ELSE

```

```

CREATE CURSOR (outfname) FROM ARRAY outarray
ENDIF

IF therm on
DO updtherm WITH 25
ENDIF

* Get rid of the temporary arrays
RELEASE outarray, coluniq, inpfields

* -----
* Add output database rows and replace the first field
* -----

* Select distinct rows into a table (instead of an array) so that
* there can be lots of rows.  If we select into an array, we may
* run out of RAM if there are many rows.

SELECT DISTINCT &rowfldname FROM (m.dbfname) INTO TABLE xtabtemp

IF therm on
DO updtherm WITH 30
ENDIF

SELECT (m.outstem)
APPEND FIELD (FIELD(1)) FROM xtabtemp

IF therm on
DO updtherm WITH 35
ENDIF
* -----
* Look up and replace the cell values
* -----
*
* This algorithm makes one pass through the input file, dropping its
* values into the output file.  It exploits the fact that the output
* file is known to be in row order.
*
* Start at the top of the output file
SELECT (m.outstem)
GOTO TOP
outfname = FIELD(1)

* Start at the top of the input file
SELECT (m.dbfstem)
GOTO TOP

SCAN
m.f1 = EVAL(m.rowfldname)          && get next row value from input
m.f2 = mapname(EVAL(m.colfldname),m.cdec) && get corresponding column value
m.f3 = EVAL(m.cellfldname)        && get cell value

* Find the right row in the output file
SELECT (m.outstem)
DO WHILE !(EVAL(outfname) == m.f1) AND !EOF()
SKIP
ENDDO

IF !EOF()
IF TYPE(m.f2) $ "NF"
REPLACE (m.f2) WITH &f2 + m.f3
ELSE
REPLACE (m.f2) WITH m.f3
ENDIF
ELSE
DO alert WITH "Unexpected end of output file.;" ;
+ "The input file may be out of sequence."
DO bailout WITH .T.
ENDIF

SELECT (m.dbfstem)

```

```

* Map thermometer to remaining portion of display
IF therm_on
DO CASE
CASE RECCOUNT() > 1000
IF RECNO() % 100 = 0
DO updtherm WITH INT(RECNO() / RECCOUNT() * 65)+ 35
ENDIF
OTHERWISE
IF RECNO() % 10 = 0
DO updtherm WITH INT(RECNO() / RECCOUNT() * 65)+ 35
ENDIF
ENDCASE
ENDIF
ENDSCAN

* Cross-foot the columns and put the results into the total field
IF m.xfoot
SELECT (m.outstem)
m.totfldname = FIELD(FCOUNT())
SCAN
* Sum the relevant fields
m.gtotal = 0
FOR i = 2 TO FCOUNT() - 1
m.gtotal = m.gtotal + EVAL(FIELD(i))
ENDFOR

REPLACE (m.totfldname) WITH m.gtotal
ENDSCAN
ENDIF

IF therm_on
DO updtherm WITH 100
DO deactthermo
ENDIF

IF USED("XTABTEMP")
SELECT xtabtemp
USE
ENDIF
IF FILE("xtabtemp.dbf")
DELETE FILE xtabtemp.dbf
ENDIF

* Close the input database
IF closeinput
SELECT (m.dbfstem)
USE
ENDIF

* Leave the output database/cursor selected
SELECT (m.outstem)
GOTO TOP

* Do closing housekeeping
DO bailout WITH .F.

RETURN

*!*****
*!
*!      Function: MAPNAME()
*!
*!      Called by: GENXTAB.PRG
*!
*!      Calls: ALERT          (procedure in GENXTAB.PRG)
*!             : BAILOUT      (procedure in GENXTAB.PRG)
*!
*!*****
FUNCTION mapname

```

* Translate a field value of any type into a string containing a valid
 * field name.

```

PARAMETER in_name, in_dec
IF PARAMETERS() = 1
  in_dec = 0
ENDIF
DO CASE
CASE TYPE("in_name") $ 'CM'
  DO CASE
  CASE EMPTY(m.in_name)
    m.retval = 'C_BLANK'
  OTHERWISE
    m.retval = SUBSTR(CHRTRAN(m.in_name,m.badchars,m.stdascii),1,10)
    IF !ISALPHA(LEFT(m.retval,1))
      m.retval = 'C_'+LEFT(m.retval,8)
    ENDIF
  ENDCASE
CASE TYPE("in_name") $ 'NF'
  m.retval = 'N_'+ALLTRIM(CHRTRAN(STR(m.in_name,8,in_dec),','))
CASE TYPE("in_name") = 'D'
  DO CASE
  CASE EMPTY(m.in_name)
    m.retval = 'D_BLANK'
  OTHERWISE
    m.retval = 'D_' + CHRTRAN(DTOS(m.in_name),m.badchars,m.stdascii)
  ENDCASE
CASE TYPE("in_name") = 'L'
  IF m.in_name = .T.
    m.retval = 'T'
  ELSE
    m.retval = 'F'
  ENDIF
CASE TYPE("in_name") = 'P'
  DO alert WITH "Picture type fields are not allowed here."
  DO bailout WITH .T.
OTHERWISE
  DO alert WITH "Unknown field type."
  DO bailout WITH .T.
ENDCASE
m.retval = PADR(UPPER(ALLTRIM(m.retval)),10)
RETURN m.retval

```

```

*!*****
*!
*!      Function: JUSTSTEM()
*!
*!      Called by: GENXTAB.PRG
*!
*!*****
FUNCTION juststem
* Return just the stem name from "filename"
PARAMETERS filename
PRIVATE ALL
IF RAT('\',m.filename) > 0
  m.filename = SUBSTR(m.filename,RAT('\',m.filename)+1,255)
ENDIF
IF RAT(':',m.filename) > 0
  m.filename = SUBSTR(m.filename,RAT(':',m.filename)+1,255)
ENDIF
IF RAT('.',m.filename) > 0
  m.filename = SUBSTR(m.filename,1,RAT('.',m.filename)-1)
ENDIF
RETURN ALLTRIM(UPPER(m.filename))

*!*****
*!
*!      Procedure: FORCEEXT
*!
*!      Calls: JUSTPATH      (procedure in GENXTAB.PRG)
*!             JUSTFNAME()  (function in GENXTAB.PRG)
*!             ADDBS        (procedure in GENXTAB.PRG)

```

```

*!
*!*****
FUNCTION forceext
* Force the extension of "filename" to be whatever ext is.
PARAMETERS filename,ext
PRIVATE ALL
IF SUBSTR(m.ext,1,1) = "."
    m.ext = SUBSTR(m.ext,2,3)
ENDIF

m.pname = justpath(m.filename)
m.filename = justfname(UPPER(ALLTRIM(m.filename)))
IF RAT('.',m.filename) > 0
    m.filename = SUBSTR(m.filename,1,RAT('.',m.filename)-1) + '.' + m.ext
ELSE
    m.filename = m.filename + '.' + m.ext
ENDIF
RETURN addbs(m.pname) + m.filename

*!*****
*!
*!      Function: DEFAULTTEXT()
*!
*!      Called by: GENXTAB.PRG
*!
*!      Calls: JUSTPATH      (procedure in GENXTAB.PRG)
*!             JUSTFNAME()   (function in GENXTAB.PRG)
*!             ADDBS         (procedure in GENXTAB.PRG)
*!
*!*****
FUNCTION defaulttext
* Add a default extension to "filename" if it doesn't have one already
PARAMETERS filename,ext
PRIVATE ALL
IF SUBSTR(ext,1,1) = "."
    m.ext = SUBSTR(m.ext,2,3)
ENDIF

m.pname = justpath(m.filename)
m.filename = justfname(UPPER(ALLTRIM(m.filename)))
IF !EMPTY(m.filename) AND AT('.',m.filename) = 0
    m.filename = m.filename + '.' + m.ext
RETURN addbs(m.pname) + m.filename
ELSE
RETURN filename
ENDIF

*!*****
*!
*!      Function: JUSTFNAME()
*!
*!      Called by: GENXTAB.PRG
*!             DEFAULTTEXT() (function in GENXTAB.PRG)
*!             FORCEEXT      (procedure in GENXTAB.PRG)
*!
*!*****
FUNCTION justfname
* Return just the filename (i.e., no path) from "filename"
PARAMETERS filename
PRIVATE ALL
IF RAT('\',m.filename) > 0
    m.filename = SUBSTR(m.filename,RAT('\',m.filename)+1,255)
ENDIF
IF RAT(':',m.filename) > 0
    m.filename = SUBSTR(m.filename,RAT(':',m.filename)+1,255)
ENDIF
RETURN ALLTRIM(UPPER(m.filename))

*!*****
*!
*!      Procedure: JUSTPATH
*!
*!      Called by: DEFAULTTEXT() (function in GENXTAB.PRG)

```



```

*!          : FORCEEXT          (procedure in GENXTAB.PRG)
*!
*!*****
FUNCTION justpath
* Return just the path name from "filename"
PARAMETERS m.filename
PRIVATE ALL
m.filename = ALLTRIM(UPPER(m.filename))
m.pathsep = IIF(_MAC,":", "\\")
IF _MAC
  m_found_it = .F.
  m_maxchar = max(RAT("\\", m.filename), RAT(":", m.filename))
  IF m_maxchar > 0
    m.filename = SUBSTR(m.filename,1,m_maxchar)
    IF RIGHT(m.filename,1) $ ":\ " AND LEN(m.filename) > 1 ;
      AND !(SUBSTR(m.filename,LEN(m.filename)-1,1) $ ":\ ")
      m.filename = SUBSTR(m.filename,1,LEN(m.filename)-1)
    ENDIF
  RETURN m.filename
ENDIF
ELSE
  IF m.pathsep $ filename
    m.filename = SUBSTR(m.filename,1,RAT(m.pathsep,m.filename))
    IF RIGHT(m.filename,1) = m.pathsep AND LEN(m.filename) > 1 ;
      AND SUBSTR(m.filename,LEN(m.filename)-1,1) <> m.pathsep
      m.filename = SUBSTR(m.filename,1,LEN(m.filename)-1)
    ENDIF
  RETURN m.filename
ENDIF
RETURN ''

*!*****
*!
*!      Procedure: ADDBS
*!
*!      Called by: DEFAULTTEXT()      (function in GENXTAB.PRG)
*!                  : FORCEEXT          (procedure in GENXTAB.PRG)
*!
*!*****
FUNCTION addbs
* Add a backslash to a path name, if there isn't already one there
PARAMETER pathname
PRIVATE ALL
m.pathname = ALLTRIM(UPPER(m.pathname))
IF !(RIGHT(m.pathname,1) $ '\:') AND !EMPTY(m.pathname)
  m.pathname = m.pathname + IIF(_MAC,":",'\\')
ENDIF
RETURN m.pathname

*!*****
*!
*!      Procedure: APPERROR
*!
*!      Called by: GENXTAB.PRG
*!
*!      Calls: ALERT          (procedure in GENXTAB.PRG)
*!              : BAILOUT      (procedure in GENXTAB.PRG)
*!
*!*****
PROCEDURE apperror
* Simple ON ERROR routine

PARAMETERS e_program,e_message,e_source,e_lineno,e_error
ON ERROR
SET MOUSE ON
m_e_source = ALLTRIM(m_e_source)
DO alert WITH 'Line No.:' +ALLTRIM(STR(m_e_lineno,5))+';' ;
  +'Program: '+m_e_program+';' ;
  +' Error: '+m_e_message+';' ;
  +' Source: '+IIF(LEN(m_e_source)<50,;

```

```

    m.e_source,SUBSTR(m.e_source,1,50)+'...')
DO bailout WITH .T.

*!*****
*!
*!      Procedure: ALERT
*!
*!      Called by: GENXTAB.PRG
*!                : APPERROR      (procedure in GENXTAB.PRG)
*!                : MAPNAME()     (function in GENXTAB.PRG)
*!
*!*****
PROCEDURE alert
* Display an error message, automatically sizing the message window
* as necessary. Semicolons in "strg" mean "new line".
PARAMETERS strg
PRIVATE ALL

SET MOUSE ON
in_talk = SET('TALK')
SET TALK OFF
in_cons = SET('CONSOLE')

m.numlines = OCCURS('; ',m.strg) + 1

DIMENSION alert_array[m.numlines]
m.remain = m.strg
m.maxlen = 0
FOR i = 1 TO m.numlines
    IF AT('; ',m.remain) > 0
        alert_array[i] = SUBSTR(m.remain,1,AT('; ',m.remain)-1)
        alert_array[i] = CHRTRAN(alert_array[i],'; ','')
        m.remain = SUBSTR(m.remain,AT('; ',m.remain)+1)
    ELSE
        alert_array[i] = m.remain
        m.remain = ''
    ENDIF
    IF LEN(alert_array[i]) > SCOLS() - 6
        alert_array[i] = SUBSTR(alert_array[i],1,SCOLS()-6)
    ENDIF
    IF LEN(alert_array[i]) > m.maxlen
        m.maxlen = LEN(alert_array[i])
    ENDIF
ENDIFOR

m.top_row = INT( (SROWS() - 4 - m.numlines) / 2)
m.bot_row = m.top_row + 3 + m.numlines

m.top_col = INT((SCOLS() - m.maxlen - 6) / 2)
m.bot_col = m.top_col + m.maxlen + 6

IF _MAC
    DEFINE WINDOW alert FROM m.top_row,m.top_col TO m.bot_row,m.bot_col;
    DOUBLE COLOR SCHEME 7
ELSE
    DEFINE WINDOW alert FROM m.top_row,m.top_col TO m.bot_row,m.bot_col;
    DOUBLE COLOR SCHEME 7
ENDIF
ACTIVATE WINDOW alert

IF _WINDOWS
    FOR i = 1 TO m.numlines
        @ i,3 SAY PADC(alert_array[i],m.maxlen) FONT "MS Sans Serif",8
    ENDFOR
ELSE
    FOR i = 1 TO m.numlines
        @ i,3 SAY PADC(alert_array[i],m.maxlen)
    ENDFOR
ENDIF
SET CONSOLE OFF
keycode = INKEY(0,'HM')
SET CONSOLE ON

```

```

RELEASE WINDOW alert
IF m.in_talk = "ON"
  SET TALK ON
ENDIF
IF m.in_cons = "OFF"
  SET CONSOLE OFF
ENDIF

RETURN

*!*****
*!
*!      Procedure: MAKESTRG
*!
*!*****
FUNCTION makestrg
PARAMETER in_val
DO CASE
CASE TYPE("in_val") = "C"
  RETURN in_val
CASE TYPE("in_val") $ "NF"
  RETURN ALLTRIM(STR(in_val))
CASE TYPE("in_val") = "D"
  RETURN DTOC(in_val)
CASE TYPE("in_val") = "L"
  IF in_val
    RETURN ".T."
  ELSE
    RETURN ".F."
  ENDIF
OTHERWISE
  RETURN in_val
ENDCASE

*!*****
*!
*!      Procedure: ESC_PROC
*!
*!      Called by: GENXTAB.PRG
*!
*!      Calls: BAILOUT          (procedure in GENXTAB.PRG)
*!
*!*****
PROCEDURE esc_proc
WAIT WINDOW "Cross tabulation terminated." TIMEOUT 1
CLEAR TYPEAHEAD
DO bailout

*!*****
*!
*!      Procedure: BAILOUT
*!
*!      Called by: GENXTAB.PRG
*!                  : APPERROR          (procedure in GENXTAB.PRG)
*!                  : ESC_PROC          (procedure in GENXTAB.PRG)
*!                  : MAPNAME()        (function in GENXTAB.PRG)
*!
*!      Uses: XTABTEMP.DBF
*!
*!*****
PROCEDURE bailout
PARAMETER docancl
PRIVATE docancl
DO CASE
CASE PARAMETERS() = 0
  m.docancl = .T.
ENDCASE
IF WONTOP('THERMOMETE')
  RELEASE WINDOW thermomete
ENDIF

```

```

IF USED("XTABTEMP")
    SELECT xtabtemp
    USE
ENDIF
IF FILE("xtabtemp.dbf")
    DELETE FILE xtabtemp.dbf
ENDIF

SET FIELDS TO &mfieldsto
IF m.fields = "ON"
    SET FIELDS ON
ELSE
    SET FIELDS OFF
ENDIF

SET UDFPARMS TO &udfparms

IF m.xsafe_stat = "ON"
    SET SAFETY ON
ENDIF
IF m.xesc_stat = "ON"
    SET ESCAPE ON
ELSE
    SET ESCAPE OFF
ENDIF
IF m.xtalk_stat = "ON"
    SET TALK ON
ENDIF
#if "MAC" $ UPPER(VERSION(1))
    IF _MAC
        SET MACDESKTOP &mmacdesk
    ENDIF
#endif

ON ERROR &in_err
ON ESCAPE &in_esc

SET MOUSE ON
IF m.docancl
    m.outfname = ''
    CANCEL
ENDIF

*
* ACTTHERM(<text>) - Activate thermometer.
*
* Activates thermometer. Update the thermometer with UPDTHERM().
* Thermometer window is named "thermometer." Be sure to RELEASE
* this window when done with thermometer. Creates the global
* m.g_thermwidth.
*
*!*****
*!
*!           Procedure: ACTTHERM
*!
*!*****
PROCEDURE acttherm
PARAMETER m.prompt
PRIVATE m.text
m.text = ""
IF _MAC OR WINDOWS
    IF TXTWIDTH(m.prompt, m.g_dlgface, m.g_dlgsize, m.g_dlgstyle) > 43
        DO WHILE TXTWIDTH(m.prompt+"...", m.g_dlgface, m.g_dlgsize, m.g_dlgstyle) > 43
            m.prompt = LEFT(m.prompt, LEN(m.prompt)-1)
        ENDDO
        m.prompt = m.prompt + "..."
    ENDIF
DO CASE
CASE WINDOWS
    DEFINE WINDOW thermomete ;
        AT INT((SROW() - (( 5.615 * ;

```

```

FONTMETRIC(1, m.g_dlgface, m.g_dlgsize, m.g_dlgstyle)) / ;
FONTMETRIC(1, WFONT(1,""), WFONT( 2,""), WFONT(3,"")) / 2), ;
INT((SCOL() - (( 63.833 * ;
FONTMETRIC(6, m.g_dlgface, m.g_dlgsize, m.g_dlgstyle)) / ;
FONTMETRIC(6, WFONT(1,""), WFONT( 2,""), WFONT(3,"")) / 2) ;
SIZE 5.615,63.833 ;
FONT m.g_dlgface, m.g_dlgsize ;
STYLE m.g_dlgstyle ;
NOFLOAT ;
NOCLOSE ;
NONE ;
COLOR RGB(0, 0, 0, 192, 192, 192)
MOVE WINDOW thermomete CENTER
ACTIVATE WINDOW thermomete NOSHOW

@ 0.5,3 SAY m.text FONT m.g_dlgface, m.g_dlgsize STYLE m.g_dlgstyle
@ 1.5,3 SAY m.prompt FONT m.g_dlgface, m.g_dlgsize STYLE m.g_dlgstyle
@ 0.000,0.000 TO 0.000,63.833 ;
COLOR RGB(255, 255, 255, 255, 255, 255)
@ 0.000,0.000 TO 5.615,0.000 ;
COLOR RGB(255, 255, 255, 255, 255, 255)
@ 0.385,0.667 TO 5.231,0.667 ;
COLOR RGB(128, 128, 128, 128, 128, 128)
@ 0.308,0.667 TO 0.308,63.167 ;
COLOR RGB(128, 128, 128, 128, 128, 128)
@ 0.385,63.000 TO 5.308,63.000 ;
COLOR RGB(255, 255, 255, 255, 255, 255)
@ 5.231,0.667 TO 5.231,63.167 ;
COLOR RGB(255, 255, 255, 255, 255, 255)
@ 5.538,0.000 TO 5.538,63.833 ;
COLOR RGB(128, 128, 128, 128, 128, 128)
@ 0.000,63.667 TO 5.615,63.667 ;
COLOR RGB(128, 128, 128, 128, 128, 128)
@ 3.000,3.333 TO 4.231,3.333 ;
COLOR RGB(128, 128, 128, 128, 128, 128)
@ 3.000,60.333 TO 4.308,60.333 ;
COLOR RGB(255, 255, 255, 255, 255, 255)
@ 3.000,3.333 TO 3.000,60.333 ;
COLOR RGB(128, 128, 128, 128, 128, 128)
@ 4.231,3.333 TO 4.231,60.500 ;
COLOR RGB(255, 255, 255, 255, 255, 255)
m.g_thermwidth = 56.269
CASE _MAC
DEFINE WINDOW thermomete ;
AT INT((SROW() - (( 5.62 * ;
FONTMETRIC(1, m.g_dlgface, m.g_dlgsize, m.g_dlgstyle)) / ;
FONTMETRIC(1, WFONT(1,""), WFONT( 2,""), WFONT(3,"")) / 2), ;
INT((SCOL() - (( 63.83 * ;
FONTMETRIC(6, m.g_dlgface, m.g_dlgsize, m.g_dlgstyle)) / ;
FONTMETRIC(6, WFONT(1,""), WFONT( 2,""), WFONT(3,"")) / 2) ;
SIZE 5.62,63.83 ;
FONT m.g_dlgface, m.g_dlgsize ;
STYLE m.g_dlgstyle ;
NOFLOAT ;
NOCLOSE ;
NONE ;
COLOR RGB(0, 0, 0, 192, 192, 192)
MOVE WINDOW thermomete CENTER
ACTIVATE WINDOW thermomete NOSHOW

IF ISCOLOR()
@ 0.000,0.000 TO 5.62,63.83 PATTERN 1;
COLOR RGB(192, 192, 192, 192, 192, 192)
@ 0.000,0.000 TO 0.000,63.83 ;
COLOR RGB(255, 255, 255, 255, 255, 255)
@ 0.000,0.000 TO 5.62,0.000 ;
COLOR RGB(255, 255, 255, 255, 255, 255)
@ 0.385,0.67 TO 5.23,0.67 ;
COLOR RGB(128, 128, 128, 128, 128, 128)
@ 0.31,0.67 TO 0.31,63.17 ;
COLOR RGB(128, 128, 128, 128, 128, 128)
@ 0.385,63.000 TO 5.31,63.000 ;

```

```

        COLOR RGB(255, 255, 255, 255, 255, 255)
    @ 5.23,0.67 TO 5.23,63.17 ;
        COLOR RGB(255, 255, 255, 255, 255, 255)
    @ 5.54,0.000 TO 5.54,63.83 ;
        COLOR RGB(128, 128, 128, 128, 128, 128)
    @ 0.000,63.67 TO 5.62,63.67 ;
        COLOR RGB(128, 128, 128, 128, 128, 128)
    @ 3.000,3.33 TO 4.23,3.33 ;
        COLOR RGB(128, 128, 128, 128, 128, 128)
    @ 3.000,60.33 TO 4.31,60.33 ;
        COLOR RGB(255, 255, 255, 255, 255, 255)
    @ 3.000,3.33 TO 3.000,60.33 ;
        COLOR RGB(128, 128, 128, 128, 128, 128)
    @ 4.23,3.33 TO 4.23,60.33 ;
        COLOR RGB(255, 255, 255, 255, 255, 255)
ELSE
    @ 0.000, 0.000 TO 5.62, 63.830 PEN 2
    @ 0.230, 0.500 TO 5.39, 63.333 PEN 1
ENDIF
@ 0.5,3 SAY m.text FONT m.g_dlgface, m.g_dlgsize STYLE m.g_dlgstyle+"T" ;
    COLOR RGB(0,0,0,192,192,192)
@ 1.5,3 SAY m.prompt FONT m.g_dlgface, m.g_dlgsize STYLE m.g_dlgstyle+"T" ;
    COLOR RGB(0,0,0,192,192,192)

m.g_thermwidth = 56.27
IF !ISCOLOR()
    @ 3.000,3.33 TO 4.23,m.g_thermwidth + 3.33
ENDIF
ENDCASE
SHOW WINDOW thermomete TOP
ELSE

    DEFINE WINDOW thermomete;
        FROM INT((SROW()-6)/2), INT((SCOL()-57)/2) ;
        TO INT((SROW()-6)/2) + 6, INT((SCOL()-57)/2) + 57;
        DOUBLE COLOR SCHEME 5
    ACTIVATE WINDOW thermomete NOSHOW

    m.g_thermwidth = 50
    @ 0,3 SAY m.prompt
    @ 2,1 TO 4,m.g_thermwidth+4

    SHOW WINDOW thermomete TOP
ENDIF
RETURN

*
* UPDTHERM(<percent>) - Update thermometer.
*
*!*****
*!
*! Procedure: UPDTHERM
*!
*! Called by: BUILD (procedure in GENSCRN.PRG)
*! : DISPATCHBUILD (procedure in GENSCRN.PRG)
*! : BUILDCTRL (procedure in GENSCRN.PRG)
*! : EXTRACTPROCS (procedure in GENSCRN.PRG)
*! : BUILDFMT (procedure in GENSCRN.PRG)
*!
*!*****
PROCEDURE updtherm
PARAMETER m.percent
PRIVATE m.nblocks, m.percent

IF !WEXIST("thermomete")
    DO acttherm WITH "Generating cross-tabulation ..."
ENDIF
ACTIVATE WINDOW thermomete

m.nblocks = (m.percent/100) * (m.g_thermwidth)
DO CASE
CASE _WINDOWS

```

```

        @ 3.000,3.333 TO 4.231,m.nblocks + 3.333 ;
        PATTERN 1 COLOR RGB(128, 128, 128, 128, 128, 128)
CASE MAC
    @ 3.000,3.33 TO 4.23,m.nblocks + 3.33 ;
    PATTERN 1 COLOR RGB(0, 0, 128, 0, 0, 128)
OTHERWISE
    @ 3,3 SAY REPLICATE("U",m.nblocks)
ENDCASE
RETURN

*
* DEACTTHERMO - Deactivate and Release thermometer window.
*
*!*****
*!
*!      Procedure: DEACTTHERMO
*!
*!*****
PROCEDURE deactthermo
IF WEXIST("thermomete")
    RELEASE WINDOW thermomete
ENDIF
RETURN

*!*****
*!
*!      Procedure: PARTIALFNAME
*!
*!*****
FUNCTION partialfname
PARAMETER m.filename, m.fillen
* Return a filename no longer than m.fillen characters. Take some chars
* out of the middle if necessary. No matter what m.fillen is, this function
* always returns at least the file stem and extension.
PRIVATE m.bname, m.elipse
m.elipse = "... " + c_pathsep
m.bname = justfname(m.filename)
DO CASE
CASE LEN(m.filename) <= m.fillen
    RETURN filename
CASE LEN(m.bname) + LEN(m.elipse) >= m.fillen
    RETURN m.bname
OTHERWISE
    m.remain = MAX(m.fillen - LEN(m.bname) - LEN(m.elipse), 0)
    RETURN LEFT(justpath(m.filename),remain)+m.elipse+m.bname
ENDCASE

*!*****
*!
*!      Procedure: removequotes
*!
*!*****
FUNCTION removequotes
PARAMETER m.fname
PRIVATE m.leftchar, m.rightchar
m.fname = ALLTRIM(m.fname)
m.leftchar = LEFT(m.fname,1)
m.rightchar = RIGHT(m.fname, 1)

IF m.leftchar = '"' AND m.rightchar = '"' ;
    OR m.leftchar = "'" AND m.rightchar = "'" ;
    OR m.leftchar = '[' AND m.rightchar = ']'
    RETURN SUBSTR(m.fname, 2, LEN(m.fname) - 2)
ELSE
    RETURN m.fname
ENDIF

*: EOF: GENXTAB.PRG

```

D.13. EditIons

```
PROCEDURE EditIons
EditFile = mIonsPath + mSeason + "ions.dbf"
USE &EditFile IN 0 ALIAS Ions
IF NOT mBadLog
  IF NOT nofile
    DEFINE WINDOW OUTPUT FROM 0,0 TO 40,85 TITLE 'Ions data' ;
    CLOSE FLOAT GROW ZOOM MDI FONT 'Arial', 9
    ACTIVATE WINDOW OUTPUT
    SELECT Ions
    SET FILTER TO Ions.site=m.mSite
    BROWSE FIELDS site:W=.F., samdat, STATUS:P="!!",;
    c1:W=.F.:P="#####.##", no2:W=.F.:P="#####.##", ;
    no3:W=.F.:P="#####.##", so4:W=.F.:P="#####.##", ;
    nh4:W=.F.:P="#####.##";
    WINDOW OUTPUT
    HIDE WINDOW OUTPUT
  USE
ENDIF
ENDIF
```


D.14. EditCarbon

```
PROCEDURE EditCarbon
EditFile = mCarbonPath + mSeason + "Car.dbf"
USE &EditFile IN 0 ALIAS Carbon
IF NOT mBadLog
  IF NOT nofile
    DEFINE WINDOW OUTPUT FROM 0,0 TO 40,110 TITLE 'Carbon data' ;
      CLOSE FLOAT GROW ZOOM MDI FONT 'Arial', 9
    ACTIVATE WINDOW OUTPUT
    SELECT Carbon
    SET FILTER TO Carbon.site=m.mSite
    BROWSE FIELDS site:W=.F., samdat, STATUS:P="!!",;
      Octc:W=.F., Ectc:W=.F.,;
      O1tc:W=.F., O2tc:W=.F., O3tc:W=.F., O4tc:W=.F.,;
      OPtc:W=.F., E1tc:W=.F., E2tc:W=.F., E3tc:W=.F.;
    WINDOW OUTPUT
  HIDE WINDOW OUTPUT
  USE
ENDIF
ENDIF
```

D.15. FBions

```
PROCEDURE FBions
EditFile = mIonsPath + mSeason + "ions.dbf"
USE &EditFile IN 0 ALIAS Ions
IF NOT mBadLog
  SELECT Ions
  SELECT DISTINCT Ions.*;
  FROM Ions ;
  WHERE Ions.STATUS = "FB";
  ORDER BY Ions.site, Ions.samdat;
  INTO TABLE IonsBlanks
  SELECT IonsBlanks
  COUNT TO NumFB
  DEFINE WINDOW OUTPUT FROM 0,0 TO 35,160 ;
  TITLE STR(NumFB) + ' Ions Field Blanks - Sort by:  F1-Cl,  F2-NO2,  F3-NO3,  F4-
  SO4,  SHIFT F1-F4 for ascending' ;
  CLOSE FLOAT GROW ZOOM MDI FONT 'Arial', 9
  ACTIVATE WINDOW OUTPUT
  INDEX ON c1 TAG c1
  INDEX ON no2 TAG no2
  INDEX ON no3 TAG no3
  INDEX ON so4 TAG so4
  PUSH KEY
  ON KEY LABEL F1 DO ClOrderD
  ON KEY LABEL F2 DO NO2OrderD
  ON KEY LABEL F3 DO NO3OrderD
  ON KEY LABEL F4 DO SO4OrderD
  ON KEY LABEL F5 SKIP ROUND(NumFB/2,0) -1
  ON KEY LABEL SHIFT+F1 DO ClOrder
  ON KEY LABEL SHIFT+F2 DO NO2Order
  ON KEY LABEL SHIFT+F3 DO NO3Order
  ON KEY LABEL SHIFT+F4 DO SO4Order
  ON KEY LABEL ESC DO FBStop
  lEndFB=.F.
  DO WHILE NOT lEndFB
    IF FIELD("comments")="COMMENTS"
      BROWSE SAVE ;
      FIELDS site:R, samdat:R, strtim:R, STATUS, c1:R, no2:R, no3:R,
so4:R, nh4:R, comments;
      WINDOW OUTPUT
    ELSE
      BROWSE SAVE ;
      FIELDS site:R, samdat:R, strtim:R, STATUS, c1:R, no2:R, no3:R,
so4:R, nh4:R;
      WINDOW OUTPUT
    ENDIF
  ENDDO
  POP KEY
*      HIDE WINDOW OUTPUT
*      RETURN
ENDIF
RETURN

PROCEDURE ClOrderD
SET ORDER TO c1 DESCENDING  && CL
DEACTIVATE WINDOW OUTPUT
MODIFY WINDOW OUTPUT TITLE STR(NumFB) + " Field Blanks sorted by Cl.  F1-Cl,  F2-NO2,
  F3-NO3,  F4-SO4,  SHIFT F1-F4 for ascending,  F5 for Median"
GO TOP
ACTIVATE WINDOW OUTPUT
RETURN

PROCEDURE NO2OrderD
SET ORDER TO no2 DESCENDING  && NO2
DEACTIVATE WINDOW OUTPUT
MODIFY WINDOW OUTPUT TITLE STR(NumFB) + " Field Blanks sorted by NO2.  F1-Cl,  F2-NO2,
  F3-NO3,  F4-SO4,  SHIFT F1-F4 for ascending,  F5 for Median"
GO TOP
ACTIVATE WINDOW OUTPUT
```

```

RETURN

PROCEDURE NO3OrderD
SET ORDER TO no3 DESCENDING  && NO3
DEACTIVATE WINDOW OUTPUT
MODIFY WINDOW OUTPUT TITLE STR(NumFB) + " Field Blanks sorted by NO3.  F1-Cl,  F2-NO2,
    F3-NO3,  F4-SO4,  SHIFT F1-F4 for ascending,  F5 for Median"
GO TOP
ACTIVATE WINDOW OUTPUT
RETURN

PROCEDURE SO4OrderD
SET ORDER TO so4 DESCENDING  && SO4
DEACTIVATE WINDOW OUTPUT
MODIFY WINDOW OUTPUT TITLE STR(NumFB) + " Field Blanks sorted by SO4.  F1-Cl,  F2-NO2,
    F3-NO3,  F4-SO4,  SHIFT F1-F4 for ascending,  F5 for Median"
GO TOP
ACTIVATE WINDOW OUTPUT
RETURN

PROCEDURE ClOrder
SET ORDER TO cl  && CL
DEACTIVATE WINDOW OUTPUT
MODIFY WINDOW OUTPUT TITLE STR(NumFB) + " Field Blanks sorted by Cl.  F1-Cl,  F2-NO2,
    F3-NO3,  F4-SO4,  SHIFT F1-F4 for ascending,  F5 for Median"
GO TOP
ACTIVATE WINDOW OUTPUT
RETURN

PROCEDURE NO2Order
SET ORDER TO no2  && NO2
DEACTIVATE WINDOW OUTPUT
MODIFY WINDOW OUTPUT TITLE STR(NumFB) + " Field Blanks sorted by NO2.  F1-Cl,  F2-NO2,
    F3-NO3,  F4-SO4,  SHIFT F1-F4 for ascending,  F5 for Median"
GO TOP
ACTIVATE WINDOW OUTPUT
RETURN

PROCEDURE NO3Order
SET ORDER TO no3  && NO3
DEACTIVATE WINDOW OUTPUT
MODIFY WINDOW OUTPUT TITLE STR(NumFB) + " Field Blanks sorted by NO3.  F1-Cl,  F2-NO2,
    F3-NO3,  F4-SO4,  SHIFT F1-F4 for ascending,  F5 for Median"
GO TOP
ACTIVATE WINDOW OUTPUT
RETURN

PROCEDURE SO4Order
SET ORDER TO so4  && SO4
DEACTIVATE WINDOW OUTPUT
MODIFY WINDOW OUTPUT TITLE STR(NumFB) + " Field Blanks sorted by SO4.  F1-Cl,  F2-NO2,
    F3-NO3,  F4-SO4,  SHIFT F1-F4 for ascending,  F5 for Median"
GO TOP
ACTIVATE WINDOW OUTPUT
RETURN

PROCEDURE FBStop
DEACTIVATE WINDOW OUTPUT
POP KEY
USE
DELETE FILE IonsBlanks.DBF
SELECT Ions
USE
lEndFB=.T.
RETURN

```

D.16. CarbonCS

```
PROCEDURE CARBONCS
EditFile = mCarbonPath + mSeason + "car.dbf"
USE &EditFile IN 0 ALIAS Carbon
IF NOT mBadLog
  SELECT carbon
  SELECT DISTINCT Carbon.*;
  FROM Carbon ;
  WHERE Carbon.STATUS = "CS";
  ORDER BY Carbon.site, Carbon.samdat;
  INTO TABLE CarbonBlanks
  SELECT CarbonBlanks
  COUNT TO NumFB
  DEFINE WINDOW OUTPUT FROM 0,0 TO 35,160 ;
  TITLE STR(NumFB) + ' Carbon Field Blanks - Sort by: F1-o1, F2-O2, F3-O3, F4-O4,
  F5-OP, F6-E1, F7-E2, F8-E3, F9-Median SHIFT F1-F8 for ascending' ;
  CLOSE FLOAT GROW ZOOM MDI FONT 'Arial', 9
  ACTIVATE WINDOW OUTPUT
  INDEX ON O1TC TAG O1
  INDEX ON O2TC TAG O2
  INDEX ON O3TC TAG O3
  INDEX ON O4TC TAG O4
  INDEX ON OPTC TAG OP
  INDEX ON E1TC TAG E1
  INDEX ON E2TC TAG E2
  INDEX ON E3TC TAG E3
  PUSH KEY
  ON KEY LABEL F1 DO O1OrderD
  ON KEY LABEL F2 DO O2OrderD
  ON KEY LABEL F3 DO O3OrderD
  ON KEY LABEL F4 DO O4OrderD
  ON KEY LABEL F5 DO OPOrderD
  ON KEY LABEL F6 DO E1OrderD
  ON KEY LABEL F7 DO E2OrderD
  ON KEY LABEL F8 DO E3OrderD
  ON KEY LABEL F9 SKIP ROUND(NumFB/2,0) -1
  ON KEY LABEL SHIFT+F1 DO O1Order
  ON KEY LABEL SHIFT+F2 DO O2Order
  ON KEY LABEL SHIFT+F3 DO O3Order
  ON KEY LABEL SHIFT+F4 DO O4Order
  ON KEY LABEL SHIFT+F5 DO OPOrder
  ON KEY LABEL SHIFT+F6 DO E1Order
  ON KEY LABEL SHIFT+F7 DO E2Order
  ON KEY LABEL SHIFT+F8 DO E3Order
  ON KEY LABEL ESC DO FBStop
  lEndFB=.F.
  DO WHILE NOT lEndFB
    BROWSE SAVE ;
    FIELDS site:R, samdat:R, strtim:R, STATUS, o1TC:R, o2TC:R, o3TC:R,
    o4TC:R, optc:R, e1TC:R, e2TC:R, e3TC:R, comment;
    WINDOW OUTPUT
  ENDDO
  POP KEY
*       HIDE WINDOW OUTPUT
*       RETURN
ENDIF
RETURN

PROCEDURE O1OrderD
SET ORDER TO o1 DESCENDING  && O1
DEACTIVATE WINDOW OUTPUT
MODIFY WINDOW OUTPUT TITLE STR(NumFB) + " Field Blanks sorted by O1. F1-o1, F2-O2,
  F3-O3, F4-O4, F5-OP, F6-E1, F7-E2, F8-E3, F9-Median SHIFT F1-F8 for ascending"
GO TOP
ACTIVATE WINDOW OUTPUT
RETURN

PROCEDURE O2OrderD
SET ORDER TO o2 DESCENDING  && O2
```

```
DEACTIVATE WINDOW OUTPUT
MODIFY WINDOW OUTPUT TITLE STR(NumFB) + " Field Blanks sorted by O2. F1-o1, F2-O2,
  F3-O3, F4-O4, F5-OP, F6-E1, F7-E2, F8-E3, F9-Median SHIFT F1-F8 for ascending"
GO TOP
ACTIVATE WINDOW OUTPUT
RETURN
```

```
PROCEDURE O3OrderD
SET ORDER TO o3 DESCENDING  && O3
DEACTIVATE WINDOW OUTPUT
MODIFY WINDOW OUTPUT TITLE STR(NumFB) + " Field Blanks sorted by O3. F1-o1, F2-O2,
  F3-O3, F4-O4, F5-OP, F6-E1, F7-E2, F8-E3, F9-Median SHIFT F1-F8 for ascending"
GO TOP
ACTIVATE WINDOW OUTPUT
RETURN
```

```
PROCEDURE O4OrderD
SET ORDER TO o4 DESCENDING  && O4
DEACTIVATE WINDOW OUTPUT
MODIFY WINDOW OUTPUT TITLE STR(NumFB) + " Field Blanks sorted by O4. F1-o1, F2-O2,
  F3-O3, F4-O4, F5-OP, F6-E1, F7-E2, F8-E3, F9-Median SHIFT F1-F8 for ascending"
GO TOP
ACTIVATE WINDOW OUTPUT
RETURN
```

```
PROCEDURE OPOrderD
SET ORDER TO OP DESCENDING  && OP
DEACTIVATE WINDOW OUTPUT
MODIFY WINDOW OUTPUT TITLE STR(NumFB) + " Field Blanks sorted by OP. F1-o1, F2-O2,
  F3-O3, F4-O4, F5-OP, F6-E1, F7-E2, F8-E3, F9-Median SHIFT F1-F8 for ascending"
GO TOP
ACTIVATE WINDOW OUTPUT
RETURN
```

```
PROCEDURE E1OrderD
SET ORDER TO E1 DESCENDING  && E1
DEACTIVATE WINDOW OUTPUT
MODIFY WINDOW OUTPUT TITLE STR(NumFB) + " Field Blanks sorted by E1. F1-o1, F2-O2,
  F3-O3, F4-O4, F5-OP, F6-E1, F7-E2, F8-E3, F9-Median SHIFT F1-F8 for ascending"
GO TOP
ACTIVATE WINDOW OUTPUT
RETURN
```

```
PROCEDURE E2OrderD
SET ORDER TO E2 DESCENDING  && E1
DEACTIVATE WINDOW OUTPUT
MODIFY WINDOW OUTPUT TITLE STR(NumFB) + " Field Blanks sorted by E2. F1-o1, F2-O2,
  F3-O3, F4-O4, F5-OP, F6-E1, F7-E2, F8-E3, F9-Median SHIFT F1-F8 for ascending"
GO TOP
ACTIVATE WINDOW OUTPUT
RETURN
```

```
PROCEDURE E3OrderD
SET ORDER TO E3 DESCENDING  && E3
DEACTIVATE WINDOW OUTPUT
MODIFY WINDOW OUTPUT TITLE STR(NumFB) + " Field Blanks sorted by E3. F1-o1, F2-O2,
  F3-O3, F4-O4, F5-OP, F6-E1, F7-E2, F8-E3, F9-Median SHIFT F1-F8 for ascending"
GO TOP
ACTIVATE WINDOW OUTPUT
RETURN
```

```
PROCEDURE O1Order
SET ORDER TO O1  && O1
DEACTIVATE WINDOW OUTPUT
MODIFY WINDOW OUTPUT TITLE STR(NumFB) + " Field Blanks sorted by C1. F1-o1, F2-O2,
  F3-O3, F4-O4, F5-OP, F6-E1, F7-E2, F8-E3, F9-Median SHIFT F1-F8 for ascending"
GO TOP
ACTIVATE WINDOW OUTPUT
RETURN
```

```
PROCEDURE O2Order
SET ORDER TO o2  && O2
```

```

DEACTIVATE WINDOW OUTPUT
MODIFY WINDOW OUTPUT TITLE STR(NumFB) + " Field Blanks sorted by O2. F1-o1, F2-O2,
    F3-O3, F4-O4, F5-OP, F6-E1, F7-E2, F8-E3, F9-Median SHIFT F1-F8 for ascending"
GO TOP
ACTIVATE WINDOW OUTPUT
RETURN

PROCEDURE O3Order
SET ORDER TO o3  && O3
DEACTIVATE WINDOW OUTPUT
MODIFY WINDOW OUTPUT TITLE STR(NumFB) + " Field Blanks sorted by O3. F1-o1, F2-O2,
    F3-O3, F4-O4, F5-OP, F6-E1, F7-E2, F8-E3, F9-Median SHIFT F1-F8 for ascending"
GO TOP
ACTIVATE WINDOW OUTPUT
RETURN

PROCEDURE O4Order
SET ORDER TO o4  && O4
DEACTIVATE WINDOW OUTPUT
MODIFY WINDOW OUTPUT TITLE STR(NumFB) + " Field Blanks sorted by O4. F1-o1, F2-O2,
    F3-O3, F4-O4, F5-OP, F6-E1, F7-E2, F8-E3, F9-Median SHIFT F1-F8 for ascending"
GO TOP
ACTIVATE WINDOW OUTPUT
RETURN

PROCEDURE OPOrder
SET ORDER TO OP  && OP
DEACTIVATE WINDOW OUTPUT
MODIFY WINDOW OUTPUT TITLE STR(NumFB) + " Field Blanks sorted by OP. F1-o1, F2-O2,
    F3-O3, F4-O4, F5-OP, F6-E1, F7-E2, F8-E3, F9-Median SHIFT F1-F8 for ascending"
GO TOP
ACTIVATE WINDOW OUTPUT
RETURN

PROCEDURE E1Order
SET ORDER TO E1  && E1
DEACTIVATE WINDOW OUTPUT
MODIFY WINDOW OUTPUT TITLE STR(NumFB) + " Field Blanks sorted by E1. F1-o1, F2-O2,
    F3-O3, F4-O4, F5-OP, F6-E1, F7-E2, F8-E3, F9-Median SHIFT F1-F8 for ascending"
GO TOP
ACTIVATE WINDOW OUTPUT
RETURN

PROCEDURE E2Order
SET ORDER TO E2  && E1
DEACTIVATE WINDOW OUTPUT
MODIFY WINDOW OUTPUT TITLE STR(NumFB) + " Field Blanks sorted by E2. F1-o1, F2-O2,
    F3-O3, F4-O4, F5-OP, F6-E1, F7-E2, F8-E3, F9-Median SHIFT F1-F8 for ascending"
GO TOP
ACTIVATE WINDOW OUTPUT
RETURN

PROCEDURE E3Order
SET ORDER TO E3  && E3
DEACTIVATE WINDOW OUTPUT
MODIFY WINDOW OUTPUT TITLE STR(NumFB) + " Field Blanks sorted by E3. F1-o1, F2-O2,
    F3-O3, F4-O4, F5-OP, F6-E1, F7-E2, F8-E3, F9-Median SHIFT F1-F8 for ascending"
GO TOP
ACTIVATE WINDOW OUTPUT
RETURN

PROCEDURE FBStop
DEACTIVATE WINDOW OUTPUT
POP KEY
USE
DELETE FILE CarbonBlanks.DBF
SELECT Carbon
USE
lEndFB=.T.
RETURN
EditXRF
PROCEDURE EditXRF

```

```

EditFile = mXRFPath + mSeason + "ele.dbf"
USE &EditFile IN 0 ALIAS XRF
IF NOT mBadLog
  IF NOT nofile
    DEFINE WINDOW OUTPUT FROM 0,0 TO 40,110 TITLE 'XRF/PESA - Press F1 to create XLS
file' ;
    CLOSE FLOAT GROW ZOOM MDI FONT 'Arial', 9
    ACTIVATE WINDOW OUTPUT
    PUSH KEY
    ON KEY LABEL F1 DO XRFout
    SELECT XRF
    SET FILTER TO XRF.site=m.mSite
    IF XRF.samdat<{12/1/2001}
      BROWSE FIELDS site:W=.F., samdat, pesstat:P="!!", xrmstat:P="!!",
pixstat:P="!!",;
      h:14:W=.F., s:14:W=.F., ps:14:W=.F., fe:14:W=.F., pfe:14:W=.F.;
      WINDOW OUTPUT
    ELSE
      BROWSE FIELDS site:W=.F., samdat, pesstat:P="!!", xrmstat:P="!!",
xrcstat:P="!!",;
      h:14:W=.F., s:14:W=.F., ys:14:W=.F., fe:14:W=.F., yfe:14:W=.F.;
      WINDOW OUTPUT
    ENDIF
    POP KEY
    HIDE WINDOW OUTPUT
    SELECT XRF
    USE
  ENDIF
ENDIF
RETURN

PROCEDURE XRFout
outname = mPathtemp + ALLTRIM(XRF.site) + mSeason + "XRF.xls"
IF XRF.samdat<{12/1/2001}
  COPY TO &outname FIELDS site, samdat, pesstat, xrmstat, pixstat,;
  h, s, ps, fe, pfe XLS
ELSE
  COPY TO &outname FIELDS site, samdat, pesstat, xrmstat, xrcstat,;
  h, s, ys, fe, yfe XLS
ENDIF
RETURN

```

D.17. EditLaser

```
PROCEDURE EditLaser
EditFile = mLaserPath + mSeason + "laser.dbf"
USE &EditFile IN 0 ALIAS Laser
IF NOT mBadLog
  IF NOT nofile
    DEFINE WINDOW OUTPUT FROM 0,0 TO 40,67 TITLE 'Laser' ;
    CLOSE FLOAT GROW ZOOM MDI FONT 'Arial', 9
    ACTIVATE WINDOW OUTPUT
    SELECT Laser
    SET FILTER TO Laser.site=m.mSite
    BROWSE FIELDS site:W=.F., samdat, STATUS:8:P="!!",;
    t1:W=.F., r1:W=.F.,;
    Laser=10000 * LOG((1000-Laser.r1)/Laser.t1):12:W=.F.:P="#####.#";
    WINDOW OUTPUT
  HIDE WINDOW OUTPUT
  USE
  ENDIF
ENDIF
```


D.18. EditSO2

```
PROCEDURE EditSO2
EditFile = mSO2path + mSeason + "so2.dbf"
USE &EditFile IN 0 ALIAS SO2
IF NOT mBadLog
  IF NOT nofile
    DEFINE WINDOW OUTPUT FROM 0,0 TO 40,85 TITLE 'SO2 data (Multiply SO4 by 2/3)' ;
    CLOSE FLOAT GROW ZOOM MDI FONT 'Arial', 9
    ACTIVATE WINDOW OUTPUT
    SELECT SO2
    SET FILTER TO SO2.site=m.mSite
    BROWSE FIELDS site:W=.F., samdat, STATUS:P="!!",;
    so4:W=.F.:P="#####.##";
    WINDOW OUTPUT
    HIDE WINDOW OUTPUT
    USE
  ENDIF
ENDIF
ENDIF
```

D.19. EditProblems

```
PROCEDURE EditProblems
EditFile = mSuppPath + "Problms.dbf"
USE &EditFile IN 0 ALIAS Problems SHARED
IF RIGHT(m.msite,1)="X"
    m.msite=LEFT(m.msite,4)+"1"
ENDIF
IF NOT mBadLog
    IF NOT nofile
        DEFINE WINDOW OUTPUT FROM 0,0 TO 35,145 TITLE 'Problems File for &msite' ;
        CLOSE FLOAT GROW ZOOM MDI FONT 'Courier New', 11
    *
        ACTIVATE WINDOW output
        SELECT Problems
        SET FILTER TO Problems.site=m.mSite
        GO TOP
        SET CENTURY ON
        SET FUNCTION 3 TO DTOC(DATE())+"    "+TIME()+"        LLA"
    *
        BROWSE
        MODIFY MEMO memtext WINDOW OUTPUT RANGE 99999,99999
        SET FUNCTION 3 TO
    *
        IN WINDOW output
    *
        HIDE WINDOW output
        USE
    ENDIF
ENDIF
ENDIF
```

D.20. EditHolds

```
PROCEDURE EditHolds
EditFile = mHoldPath + "Holdsite.dbf"
USE &EditFile IN 0 ALIAS Holds SHARED
IF NOT mBadLog
  IF NOT nofile
    DEFINE WINDOW OUTPUT FROM 0,0 TO 35,47 TITLE 'Sites Held Back' ;
    CLOSE FLOAT GROW ZOOM MDI FONT 'Arial', 9
    ACTIVATE WINDOW OUTPUT
    SELECT Holds
    SET ORDER TO sitemon
    SET FILTER TO Holds.QTR=m.mSeason
  *
    BROWSE IN WINDOW output FOR NOT DELETED()
    ON ERROR
    BROWSE FIELDS site:P='!!!!N', QTR:P='!NN',;
      mon, reason, cleared;
    WINDOW OUTPUT
    HIDE WINDOW OUTPUT
    USE
  ENDIF
ENDIF
ENDIF
```

D.21. EditFlash

```
PROCEDURE EditFlash
EditFile = mFlashPath + mSite + mSeason + ".dbf"
USE &EditFile IN 0 ALIAS Flash
IF NOT mBadLog
  IF NOT nofile
    *      DEFINE WINDOW output FROM 0,0 TO 51,130 TITLE 'Flash Data' ;
SYSTEM CLOSE FLOAT GROW ZOOM MDI FONT 'Arial', 9
    *      ACTIVATE WINDOW output
            SELECT Flash
            BROWSE &&;
            IN WINDOW OUTPUT
    *      HIDE WINDOW output
            USE
ENDIF
ENDIF
```

D.22. EditMag

```
PROCEDURE EditMag
EditFile = mCalPath + "IMPMGCAL.DBF"
USE &EditFile IN 0 ALIAS MagFile
SELECT MagFile
COPY TO mag
USE Mag
IF NOT mBadLog
  IF NOT nofile
    DEFINE WINDOW OUTPUT FROM 0,0 TO 35,120 TITLE 'Mag Cal' ;
      CLOSE FLOAT GROW ZOOM MDI FONT 'Arial', 9
    ACTIVATE WINDOW OUTPUT
    SELECT Mag
    SET FILTER TO ALLTRIM(Mag.sitecode)=m.mSite
    GO BOTTOM
    BROWSE LOCK -2 WINDOW OUTPUT
    SET FILTER TO
    COPY TO &EditFile FOX2X
    HIDE WINDOW OUTPUT
    USE
  ENDIF
ENDIF
```

D.23. EditVac

```
PROCEDURE EditVac
EditFile = mCalPath + "IMPPSCAL.DBF"
IF NOT mBadLog
  USE &EditFile IN 0 ALIAS VacFile
  SELECT VacFile
  COPY TO vac
  USE vac
  IF NOT nofile
    DEFINE WINDOW OUTPUT FROM 0,0 TO 35,120 TITLE 'Vac Cal' ;
    CLOSE FLOAT GROW ZOOM MDI FONT 'Arial', 9
    ACTIVATE WINDOW OUTPUT
    SELECT Vac
    SET FILTER TO Vac.sitecode=m.mSite
    GO BOTTOM
    BROWSE LOCK -2 WINDOW OUTPUT
    SET FILTER TO
    COPY TO &EditFile FOX2X
    HIDE WINDOW OUTPUT
    USE
  ENDIF
ENDIF
```

D.24. EditBal

```
PROCEDURE EditBal
mBadLog=.F.
USE &EditFile IN 0 ALIAS Baleq
IF NOT mBadLog
  IF NOT nofile
    DEFINE WINDOW OUTPUT FROM 0,0 TO 35,85 TITLE "&cTitle" ;
      CLOSE FLOAT GROW ZOOM MDI FONT 'Arial', 9
    ACTIVATE WINDOW OUTPUT
    SELECT Baleq
    GO BOTTOM
    BROWSE WINDOW OUTPUT NOEDIT NODELETE
    HIDE WINDOW OUTPUT
    USE
  ENDIF
ENDIF
ENDIF
```

D.25. FlowRSD

PROCEDURE FLOWRSD

PARAMETERS mSwap && this is true if FLOWRSD is called from SWAP
PUBLIC mSite, mSeason, mVolsummary, mFile, mThisSeason, mLogFile, mBadLog, mHigh
DIMENSION QTR(4)

ON ERROR DO errhand WITH ;
 ERROR(), MESSAGE(), MESSAGE(1), PROGRAM(), LINENO()

SET TALK OFF
SET SAFETY OFF && Make this active to remove dialog box on overwriting site and sites.

QTR(1) = "A"
QTR(2) = "B"
QTR(3) = "C"
QTR(4) = "D"
iSeason=mSeason

CLEAR

IF NOT mSwap
 CLOSE DATABASES ALL
 STORE SYS(5)+SYS(2003) TO DefaultDir
 mGETENV = 'M:\zDEFAULT\FOXPRO\
 mVolsummary = mGETENV + "volsummary.dbf"
 mRootPath = "U:\IMPROVE\
 mLogsPath = mRootPath + "logs\
 mCalPath = mRootPath + "SiteCal\
 mSiteFile = mCalPath + "sitefile.dbf"
 mLogFile = "u:\improve\logs\logs.dbf"
ENDIF

*SET DEFAULT TO M:\IMPROVE\Temp
SET DEFAULT TO &mGETENV
ON ERROR DO QLOGSERR
GO TOP
IF NOT mSwap
 USE &mLogFile IN 0 ALIAS logs
 mSite = logs.site
 mDate = logs.samdat
 USE
ENDIF

IF SELECT("Site")>0
 SELECT site
 USE
ENDIF
IF FILE("site.dbf")
 DELETE FILE "site.dbf")
ENDIF
SELECT DISTINCT LogSites.site;
 FROM LogSites;
 ORDER BY LogSites.site;
 INTO TABLE Site
SELECT site
INDEX ON site TAG site
GO TOP
IF site<>" ALL"
 INSERT INTO Site (site) VALUES (" ALL")
ENDIF
SELECT site
LOCATE FOR site.site=mSite

*mMonth = MONTH(mDate)
*mYear = YEAR(mDate)
*mQtr = QTR(mMonth/3) + RIGHT(STR(mYear),2)
IF NOT mSwap
 mSeason = mQtr


```

    CurrSeas = mSeason
ENDIF
STORE "U:\IMPROVE\FLASHDBF\" TO mFlashDir
IF SELECT("volsummary")>0
    SELECT volsummary
    USE
ENDIF
IF FILE('&mvolsummary')
    DELETE FILE &mVolsummary.DBF
ENDIF

DO WHILE .T.
    SELECT site
    LOCATE FOR site.site=msite
    IF mSite = " ALL"
        WAIT WINDOW AT 15,30 NOWAIT "Please wait..."
        DO ALLSITES
        SET DEFAULT TO &DefaultDir
        SELECT site
        GO top
        mSite=site.site
    *CANCEL
    ELSE
        mSite = IIF(RIGHT(mSite,1) = "X", LEFT(mSite,4) + "1", mSite)
    *
        RELEASE mfile
        mFile = mFlashDir + mSite + mSeason + ".dbf"
        IF SELECT("Flash")>0
            SELECT Flash
            USE
        ENDIF
        IF NOT FILE("&mFile")      && Display message if Flashcard file does not exist
            WAIT WINDOW AT 11, 80 NOWAIT TIMEOUT 5 "File &mFile does not exist"
    *
        LOOP
    * ENDIF
    ELSE
        USE "&mFile" IN 0 ALIAS Flash
        DO SQLSel
    *
        Zero Channel E readings if there is no X site to correspond with this one.
        SELECT site
        STORE RECNO() TO mRecord
        IF NOT SEEK(LEFT(mSite,4) + "X")
            SELECT volsummary
            REPLACE ALL EMagRSD WITH 0
            REPLACE ALL EVacRSD WITH 0
    *
        DELETE ALL FOR AMagRsd<5 AND BMagRsd<5 AND CMagRsd<5 AND DVacRsd<5 AND
        EMagRsd<5 AND EVacRsd<5
        PACK
        ENDIF
        SELECT site
        GOTO mRecord
        SELECT volsummary
        REPLACE ALL RECNO WITH RECNO()
        GO TOP
        SELECT Flash
        USE
    ENDIF
ENDIF
ENDIF
SELECT site
LOCATE FOR site.site=msite

frmInput = CREATEOBJECT('Form')
frmInput.CLOSABLE = .F.
frmInput.MOVABLE = .T.
frmInput.BORDERSTYLE = 3
frmInput.MOVE(0,0,820,680)
frmInput.CAPTION = "Check Flow Transducer Variability - Percent Relative Standard
Deviation"
frmInput.NAME = "Form1"
frmInput.ADDOBJECT('Label2', 'SeasonLabel')
frmInput.ADDOBJECT('mSeason', 'SeasonBox')
frmInput.ADDOBJECT('Label1', 'SiteLabel')
frmInput.ADDOBJECT('mSite', 'SiteBox')

```

```

frmInput.ADDOBJECT('Command1', 'ContinueBtn')
frmInput.ADDOBJECT('QuitBtn', 'QuitBtn1')
frmInput.ADDOBJECT('Grid1', '1stGrid')
frmInput.Label1.VISIBLE = .T.
frmInput.mSite.VISIBLE = .T.
frmInput.Label2.VISIBLE = .T.
frmInput.mSeason.VISIBLE = .F.
ON KEY LABEL F2 DO SeasonChange
frmInput.Label2.CAPTION = "Season:      " + m.mSeason + CHR(13) + "F2 to change
"
*frmInput.Label2.refresh
frmInput.Command1.VISIBLE = .T.
frmInput.QuitBtn.VISIBLE = .T.
frmInput.Grid1.VISIBLE = .T.
mHigh = 3
frmInput.Grid1.COLUMNS(5).DYNAMICFORECOLOR = ;
  "IIF(volsummary.number > 96, RGB(255,0,0), IIF(volsummary.number < 96,
RGB(0,128,0), RGB(96,96,96)))"
frmInput.Grid1.COLUMNS(6).DYNAMICFORECOLOR = ;
  "IIF(volsummary.amagrsd >= mHigh, RGB(0,0,255), RGB(96,96,96))"
frmInput.Grid1.COLUMNS(7).DYNAMICFORECOLOR = ;
  "IIF(volsummary.avacrsd >= mHigh, RGB(0,0,255), RGB(96,96,96))"
frmInput.Grid1.COLUMNS(8).DYNAMICFORECOLOR = ;
  "IIF(volsummary.bmagrsd >= mHigh, RGB(0,0,255), RGB(96,96,96))"
frmInput.Grid1.COLUMNS(9).DYNAMICFORECOLOR = ;
  "IIF(volsummary.bvacrsd >= mHigh, RGB(0,0,255), RGB(96,96,96))"
frmInput.Grid1.COLUMNS(10).DYNAMICFORECOLOR = ;
  "IIF(volsummary.cmagrsd >= mHigh, RGB(0,0,255), RGB(96,96,96))"
frmInput.Grid1.COLUMNS(11).DYNAMICFORECOLOR = ;
  "IIF(volsummary.cvacrsd >= mHigh, RGB(0,0,255), RGB(96,96,96))"
frmInput.Grid1.COLUMNS(12).DYNAMICFORECOLOR = ;
  "IIF(volsummary.dvacrsd >= 2, RGB(0,0,255), RGB(96,96,96))"
frmInput.Grid1.COLUMNS(13).DYNAMICFORECOLOR = ;
  "IIF(volsummary.emagrsd < .005, RGB(255,255,255), IIF(volsummary.emagrsd >=
mHigh, RGB(0,0,255), RGB(96,96,96)))"
frmInput.Grid1.COLUMNS(14).DYNAMICFORECOLOR = ;
  "IIF(volsummary.emagrsd < .005, RGB(255,255,255), IIF(volsummary.emagrsd >=
mHigh, RGB(0,0,255), RGB(96,96,96)))"
frmInput.Grid1.COLUMNS(15).DYNAMICFORECOLOR = ;
  "IIF(volsummary.Temprsd >= 7, RGB(0,0,255), RGB(96,96,96))"
frmInput.Grid1.COLUMNS(16).DYNAMICFORECOLOR = ;
  "IIF(volsummary.TempC >= 30, RGB(0,0,255), RGB(96,96,96))"
frmInput.Grid1.COLUMNS(5).DYNAMICFONTBOLD = "IIF(volsummary.number <> 96, .T.,
.F.)"
frmInput.Grid1.COLUMNS(6).DYNAMICFONTBOLD = "IIF(volsummary.amagrsd >= mHigh, .T.,
.F.)"
frmInput.Grid1.COLUMNS(7).DYNAMICFONTBOLD = "IIF(volsummary.avacrsd >= mHigh, .T.,
.F.)"
frmInput.Grid1.COLUMNS(8).DYNAMICFONTBOLD = "IIF(volsummary.bmagrsd >= mHigh, .T.,
.F.)"
frmInput.Grid1.COLUMNS(9).DYNAMICFONTBOLD = "IIF(volsummary.bvacrsd >= mHigh, .T.,
.F.)"
frmInput.Grid1.COLUMNS(10).DYNAMICFONTBOLD = "IIF(volsummary.cmagrsd >= mHigh, .T.,
.F.)"
frmInput.Grid1.COLUMNS(11).DYNAMICFONTBOLD = "IIF(volsummary.cvacrsd >= mHigh, .T.,
.F.)"
frmInput.Grid1.COLUMNS(12).DYNAMICFONTBOLD = "IIF(volsummary.dvacrsd >= 2, .T.,
.F.)"
frmInput.Grid1.COLUMNS(13).DYNAMICFONTBOLD = "IIF(volsummary.emagrsd >= mHigh, .T.,
.F.)"
frmInput.Grid1.COLUMNS(14).DYNAMICFONTBOLD = "IIF(volsummary.evacrsd >= mHigh, .T.,
.F.)"
frmInput.Grid1.COLUMNS(15).DYNAMICFONTBOLD = "IIF(volsummary.Temprsd >= 7, .T.,
.F.)"
frmInput.Grid1.COLUMNS(16).DYNAMICFONTBOLD = "IIF(volsummary.TempC >= 30, .T.,
.F.)"
frmInput.Grid1.COLUMNS(6).INPUTMASK="###.##"
frmInput.Grid1.COLUMNS(7).INPUTMASK="###.##"
frmInput.Grid1.COLUMNS(8).INPUTMASK="###.##"
frmInput.Grid1.COLUMNS(9).INPUTMASK="###.##"
frmInput.Grid1.COLUMNS(10).INPUTMASK="###.##"
frmInput.Grid1.COLUMNS(11).INPUTMASK="###.##"

```

```

frmInput.Grid1.COLUMNS(12).INPUTMASK="###.##"
frmInput.Grid1.COLUMNS(13).INPUTMASK="###.##"
frmInput.Grid1.COLUMNS(14).INPUTMASK="###.##"
frmInput.Grid1.COLUMNS(15).INPUTMASK="###.##"
frmInput.Grid1.COLUMNS(16).INPUTMASK="###.##"
frmInput.Grid1.COLUMNS(1).Header1.CAPTION = "Record"
frmInput.Grid1.COLUMNS(2).Header1.CAPTION = "Site"
frmInput.Grid1.COLUMNS(3).Header1.CAPTION = "Date"
frmInput.Grid1.COLUMNS(4).Header1.CAPTION = "Solenoid"
frmInput.Grid1.COLUMNS(5).Header1.CAPTION = "Number"
frmInput.Grid1.COLUMNS(6).Header1.CAPTION = "A Mag "
frmInput.Grid1.COLUMNS(7).Header1.CAPTION = "A Vac "
frmInput.Grid1.COLUMNS(8).Header1.CAPTION = "B Mag "
frmInput.Grid1.COLUMNS(9).Header1.CAPTION = "B Vac "
frmInput.Grid1.COLUMNS(10).Header1.CAPTION = "C Mag "
frmInput.Grid1.COLUMNS(11).Header1.CAPTION = "C Vac "
frmInput.Grid1.COLUMNS(12).Header1.CAPTION = "D Vac "
frmInput.Grid1.COLUMNS(13).Header1.CAPTION = "E Mag "
frmInput.Grid1.COLUMNS(14).Header1.CAPTION = "E Vac "
frmInput.Grid1.COLUMNS(15).Header1.CAPTION = "Temp RSD "
frmInput.Grid1.COLUMNS(16).Header1.CAPTION = "Temp C "
frmInput.Grid1.COLUMNS(1).Header1.ALIGNMENT = 2
frmInput.Grid1.COLUMNS(2).Header1.ALIGNMENT = 2
frmInput.Grid1.COLUMNS(3).Header1.ALIGNMENT = 2
frmInput.Grid1.COLUMNS(4).Header1.ALIGNMENT = 2
frmInput.Grid1.COLUMNS(5).Header1.ALIGNMENT = 2
frmInput.Grid1.COLUMNS(6).Header1.ALIGNMENT = 1
frmInput.Grid1.COLUMNS(7).Header1.ALIGNMENT = 1
frmInput.Grid1.COLUMNS(8).Header1.ALIGNMENT = 1
frmInput.Grid1.COLUMNS(9).Header1.ALIGNMENT = 1
frmInput.Grid1.COLUMNS(10).Header1.ALIGNMENT = 1
frmInput.Grid1.COLUMNS(11).Header1.ALIGNMENT = 1
frmInput.Grid1.COLUMNS(12).Header1.ALIGNMENT = 1
frmInput.Grid1.COLUMNS(13).Header1.ALIGNMENT = 1
frmInput.Grid1.COLUMNS(14).Header1.ALIGNMENT = 1
frmInput.Grid1.COLUMNS(15).Header1.ALIGNMENT = 1
frmInput.Grid1.COLUMNS(16).Header1.ALIGNMENT = 1
frmInput.Grid1.COLUMNS(1).WIDTH = 55
frmInput.Grid1.COLUMNS(2).WIDTH = 45
frmInput.Grid1.COLUMNS(3).WIDTH = 70
frmInput.Grid1.COLUMNS(4).WIDTH = 45
frmInput.Grid1.COLUMNS(5).WIDTH = 45
frmInput.Grid1.COLUMNS(6).WIDTH = 40
frmInput.Grid1.COLUMNS(7).WIDTH = 40
frmInput.Grid1.COLUMNS(8).WIDTH = 40
frmInput.Grid1.COLUMNS(9).WIDTH = 40
frmInput.Grid1.COLUMNS(10).WIDTH = 40
frmInput.Grid1.COLUMNS(11).WIDTH = 40
frmInput.Grid1.COLUMNS(12).WIDTH = 40
frmInput.Grid1.COLUMNS(13).WIDTH = 40
frmInput.Grid1.COLUMNS(14).WIDTH = 40
frmInput.Grid1.COLUMNS(15).WIDTH = 65
frmInput.Grid1.COLUMNS(16).WIDTH = 60

```

```

frmInput.mSite.VISIBLE = .T.
SELECT site
LOCATE FOR site.site=mSite
frmInput.mSite.VALUE = mSite
frmInput.SHOW
READ EVENTS

```

```

ENDDO
ON ERROR  && restore system error handler
SET DEFAULT TO &DefaultDir
SELECT site
USE

```

```

* Object definitions for display form follow
DEFINE CLASS SeasonLabel AS LABEL
    AUTOSIZE = .F.
    FONTBOLD = .T.
    FONTSIZE = 9

```

```

ALIGNMENT = 1
CAPTION = "Season:          " + m.mSeason + CHR(13) + "F2 to change      "
LEFT = 50
TOP = 25
HEIGHT = 30
WIDTH = 98
NAME = "Label2"
ENDDDEFINE

DEFINE CLASS SeasonBox AS TEXTBOX
    FONTBOLD = .T.
    FONTSIZE = 9
    ALIGNMENT = 1
    VALUE = m.mSeason
    SELSTART = 0
    SELLENGTH = 3
    SELECTIONENTRY = .T.
    CONTROLSOURCE = "m.mSeason"
    FORMAT = "!"
    TOP = 24
    LEFT = 126
    HEIGHT = 25
    WIDTH = 44
    MAXLENGTH = 3
    SPECIALEFFECT = 0
    NAME = "mSeason"
    PROCEDURE VALID
    IF mSeason<>iSeason
        IF mSeason = CurrSeas
            mLogFile = mLogsPath + "logs.dbf"
        ELSE
            mLogFile = mLogsPath + mSeason + "logs.dbf"
        ENDIF
        mBadLog = .F.
        ON ERROR DO QLOGSERR
        IF SELECT("logs")<>0
            SELECT logs
            USE
        ENDIF
        USE &mLogFile IN 0 ALIAS logs
*    On Error
        IF NOT mBadLog
            IF SELECT("site")<>0
                SELECT site
                USE
            ENDIF
            SET DEFAULT TO &mGETENV

            SELECT DISTINCT logs.site;
            FROM logs;
            ORDER BY logs.site;
            INTO TABLE Site
            SELECT Site
            INDEX ON site TAG site
            GO TOP
            IF site<>" ALL"
                INSERT INTO Site (site) VALUES (" ALL")
            ENDIF
            LOCATE FOR Site.site = mSite
            SELECT logs
            USE
            SELECT site
            SET DEFAULT TO &DefaultDir
        ENDIF
    ENDIF
    frmInput.mSite.VISIBLE = .T.
    frmInput.REFRESH
    frmInput.mSeason.VISIBLE = .F.
    frmInput.Label2.CAPTION = "Season:          " + m.mSeason + CHR(13) + "F2 to change
    "
ENDPROC
ENDDDEFINE

```

```

DEFINE CLASS SiteLabel AS LABEL
    AUTOSIZE = .F.
    FONTBOLD = .T.
    FONTSIZE = 9
    ALIGNMENT = 1
    CAPTION = "Site Name:"+CHR(13)+"('ALL' for report)"
    TOP = 28
    LEFT = 190
    HEIGHT = 30
    WIDTH = 88
    NAME = "Label1"
ENDDDEFINE

DEFINE CLASS SiteBox AS LISTBOX
    FONTBOLD = .T.
    FONTSIZE = 8
    VALUE = mSite
    CONTROLSOURCE = Site.site
    TOP = 10
    LEFT = 290
    WIDTH = 75
    HEIGHT = 50
    MAXLENGTH = 5
    SPECIALEFFECT = 0
    COLUMNCOUNT = 1
    NAME = "mSite"
    SELSTART = 1
    SELLENGTH = 5
    SELECTIONENTRY = .T.
    ROWSOURCETYPE = 2
    ROWSOURCE = "site"
    DISPLAYVALUE = 1
    NUMBEROFELEMENTS = 180
    INPUTMASK = "!!!!!"
    READONLY = .F.
    INCREMENTALSEARCH = .T.
    PROCEDURE VALID
        m.mSite = site.site
        CLEAR EVENTS
        THISFORM.RELEASE
        WAIT CLEAR
        IF SELECT("Flash")>0
            SELECT Flash
            USE
        ENDIF
    ENDPROC
ENDDDEFINE

DEFINE CLASS ContinueBtn AS COMMANDBUTTON
    TOP = 24
    LEFT = 390
    HEIGHT = 25
    WIDTH = 121
    FONTBOLD = .T.
    FONTSIZE = 10
    CANCEL = .F.
    CAPTION = "\<Continue"
    DEFAULT = .T.
    TERMINATEREAD = .F.
    NAME = "Command1"

    PROCEDURE CLICK
*       IIF(len(msite)=0, m.msite=" ", m.msite = site.site)
    m.mSite = Site.site
    CLEAR EVENTS
    THISFORM.RELEASE
    WAIT CLEAR
    IF SELECT("Flash")>0
        SELECT Flash
        USE
    ENDIF

```

```

ENDPROC
ENDEDEFINE

DEFINE CLASS QuitBtn1 AS COMMANDBUTTON
TOP = 24
LEFT = 540
HEIGHT = 25
WIDTH = 121
FONTBOLD = .T.
FONTSIZE = 10
CANCEL = .T.
CAPTION = "\<Quit"
TERMINATEREAD = .F.
NAME = "QuitBtn"
PROCEDURE CLICK
IF SELECT("Flash")>0
    SELECT Flash
    USE
ENDIF
ON ERROR
SET DEFAULT TO &DefaultDir
SET SAFETY ON
DEACTIVATE WINDOW ALL
*   cancel
SET SAFETY OFF
RETURN TO MASTER
ENDPROC
ENDEDEFINE

DEFINE CLASS 1stGrid AS GRID
TOP = 70
LEFT = 12
HEIGHT = 600
WIDTH = 800
RECORDSOURCE = "Volsummary"
NAME = "Grid1"
COLUMNCOUNT = 16
Column1.ALIGNMENT = 2
Column2.ALIGNMENT = 2
Column3.ALIGNMENT = 2
Column4.ALIGNMENT = 2
Column5.ALIGNMENT = 2
ENDEDEFINE

PROCEDURE SeasonChange
mSite = IIF(RIGHT(mSite,1) = "X", LEFT(mSite,4) + "1", mSite)
mFile = mFlashDir + mSite + mSeason + ".dbf"
frmInput.mSite.VISIBLE = .T.
frmInput.mSeason.VISIBLE = .T.
frmInput.mSeason.SETFOCUS
frmInput.REFRESH
RETURN

```

D.26. AllSites

```
PROCEDURE AllSites
PUBLIC MissData(15)

mRootPath = "U:\IMPROVE\"
mCalPath = mRootPath + "SiteCal\"
mSiteFile = mCalPath + "sitefile.dbf"
mOutPath = mRootPath + "FlashDBF\Reports\"
mOutFile = mOutPath + "&mSeason" + "Flows.dbf"
Tempout = "m:\improve\temp\tOutfile.dbf"

* Set up output file and initialize array for scatter data
IF SELECT("Out")>0
    SELECT Out
    USE
ENDIF
SELECT 0
*CREATE &mOutFile
CREATE TABLE &mOutFile (Recno I,Site C(5), Date D, Solenoid I, Number I, ;
    AMagRsd N(20,5), AVacRsd N(20,5),;
    BMagRsd N(20,5), BVacRsd N(20,5),;
    CMagRsd N(20,5), CVacRsd N(20,5),;
    DVacRsd N(20,5),;
    EMagRsd N(20,5), EVacRsd N(20,5),;
    TempRsd N(20,5), TempC N(20,5))
APPEND BLANK
COPY TO &Tempout FOX2X
USE &Tempout ALIAS out
GO BOTTOM
SCATTER TO MissData
DELETE
PACK

SELECT site
DO WHILE NOT EOF('site')
    mSite = site.Site
    IF RIGHT(mSite,1)="X"
        SKIP IN site
    LOOP
ENDIF
mFile = mFlashDir + mSite + mSeason + ".dbf"
IF SELECT("Flash")>0
    SELECT Flash
    USE
ENDIF
IF NOT FILE("&mFile")
    Missdata(1) = 0
    MissData(2) = mSite
    MissData(4) = 0
    MissData(5) = 0
    MissData(6) = -99
    MissData(7) = -99
    MissData(8) = -99
    MissData(9) = -99
    MissData(10) = -99
    MissData(11) = -99
    MissData(12) = -99
    MissData(13) = -99
    MissData(14) = -99
    MissData(15) = -99
    SELECT out
    APPEND BLANK
    GO BOTTOM
    GATHER FROM MissData
*   SELECT site
*   SKIP
*   LOOP
*   ENDIF
ELSE
```

```

USE "&mFile" IN 0 ALIAS Flash
DO SQLSel
*
Zero Channel E readings if there is no X site to correspond with this one.
SELECT site
STORE RECNO() TO mRecord
IF NOT SEEK(LEFT(mSite,4) + "X")
    SELECT Volsummary
    REPLACE ALL EMagRSD WITH 0
    REPLACE ALL EVacRSD WITH 0
ENDIF
SELECT site
GOTO mRecord
SELECT Volsummary
REPLACE ALL Recno with Recno()
USE
SELECT out
APPEND FROM &mVolsummary
SELECT Flash
USE
ENDIF
SELECT site
SKIP
ENDDO
USE &mOutFile in 0 Alias Keep
SELECT Keep
DELETE ALL
PACK
APPEND FROM &Tempout
USE
SELECT out
*SET FILTER TO date>={8/1/2004}
WAIT clear
REPORT FORM Flow for (AMagRSd>mhigh OR AVacRsd>mhigh OR BMagRSd>mhigh OR BVacRSD>mhigh
    OR CMagRSd>mhigh OR CVacRsd>mhigh OR DVacRsd>3 OR EMagRSd>mhigh OR EVacRsd>mhigh OR
    Number<72 OR Number>104) Preview
USE
DELETE FILE &Tempout
SELECT site

RETURN

```


D.27. SQLSel

PROCEDURE SQLSEL

* Extracts data from FlashCard on daily variability

```
IF RIGHT(mSite,1)="9"
  SELECT recno() AS RecNo, Flash.Site AS Site, Flash.samdat AS Date, Flash.sol AS
  Solenoid,;
  COUNT(Flash.af1m) AS Number,;
  0 AS AMagRSd,;
  100*SQRT((SUM(Flash.af1g*Flash.af1g)-
SUM(Flash.af1g)^2/COUNT(Flash.af1g))/COUNT(Flash.af1g)/AVG(Flash.af1g) AS
AVacRSd,;
  0 AS BMagRSd,;
  100*SQRT((SUM(Flash.bf1g*Flash.bf1g)-
SUM(Flash.bf1g)^2/COUNT(Flash.bf1g))/COUNT(Flash.bf1g)/AVG(Flash.bf1g) AS
BVacRSd,;
  0 AS CMagRSd,;
  100*SQRT((SUM(Flash.cf1g*Flash.cf1g)-
SUM(Flash.cf1g)^2/COUNT(Flash.cf1g))/COUNT(Flash.cf1g)/AVG(Flash.cf1g) AS
CvacRSd,;
  100*SQRT((SUM(Flash.df1g*Flash.df1g)-
SUM(Flash.df1g)^2/COUNT(Flash.df1g))/COUNT(Flash.df1g)/AVG(Flash.df1g) AS
DVacRSd,;
  100*SQRT((SUM(Flash.ef1g*Flash.ef1g)-
SUM(Flash.ef1g)^2/COUNT(Flash.ef1g))/COUNT(Flash.ef1g)/AVG(Flash.ef1g) AS
EvacRSd,;
  0 AS EMagRSd,;
  100*SQRT((SUM(Flash.Temp*Flash.Temp)-
SUM(Flash.Temp)^2/COUNT(Flash.Temp))/COUNT(Flash.Temp)/AVG(Flash.Temp) AS
TempRSd,;
  AVG(Flash.Temp)*0.1141-85.024 AS TempC;
  INTO TABLE &mVolSummary;
  HAVING Number>15;
  FROM &mFile AS Flash;
  GROUP BY Flash.samdat, Flash.sol, Flash.site;
  ORDER BY Flash.samdat
ELSE
  SELECT recno() AS RecNo, Flash.Site AS Site, Flash.samdat AS Date, Flash.sol AS
  Solenoid,;
  COUNT(Flash.af1m) AS Number,;
  100*SQRT((SUM(Flash.af1m*Flash.af1m)-
SUM(Flash.af1m)^2/COUNT(Flash.af1m))/COUNT(Flash.af1m)/AVG(Flash.af1m) AS
AMagRSd,;
  100*SQRT((SUM(Flash.af1g*Flash.af1g)-
SUM(Flash.af1g)^2/COUNT(Flash.af1g))/COUNT(Flash.af1g)/AVG(Flash.af1g) AS
AVacRSd,;
  100*SQRT((SUM(Flash.bf1m*Flash.bf1m)-
SUM(Flash.bf1m)^2/COUNT(Flash.bf1m))/COUNT(Flash.bf1m)/AVG(Flash.bf1m) AS
BMagRSd,;
  100*SQRT((SUM(Flash.bf1g*Flash.bf1g)-
SUM(Flash.bf1g)^2/COUNT(Flash.bf1g))/COUNT(Flash.bf1g)/AVG(Flash.bf1g) AS
BVacRSd,;
  100*SQRT((SUM(Flash.cf1m*Flash.cf1m)-
SUM(Flash.cf1m)^2/COUNT(Flash.cf1m))/COUNT(Flash.cf1m)/AVG(Flash.cf1m) AS
CMagRSd,;
  100*SQRT((SUM(Flash.cf1g*Flash.cf1g)-
SUM(Flash.cf1g)^2/COUNT(Flash.cf1g))/COUNT(Flash.cf1g)/AVG(Flash.cf1g) AS
CvacRSd,;
  100*SQRT((SUM(Flash.df1g*Flash.df1g)-
SUM(Flash.df1g)^2/COUNT(Flash.df1g))/COUNT(Flash.df1g)/AVG(Flash.df1g) AS
DVacRSd,;
  100*SQRT((SUM(Flash.ef1m*Flash.ef1m)-
SUM(Flash.ef1m)^2/COUNT(Flash.ef1m))/COUNT(Flash.ef1m)/AVG(Flash.ef1m) AS
EMagRSd,;
  100*SQRT((SUM(Flash.ef1g*Flash.ef1g)-
SUM(Flash.ef1g)^2/COUNT(Flash.ef1g))/COUNT(Flash.ef1g)/AVG(Flash.ef1g) AS
EvacRSd,;
```

```
        100*SQRT((SUM(Flash.Temp*Flash.Temp) -
SUM(Flash.Temp)^2/COUNT(Flash.Temp))/COUNT(Flash.Temp))/AVG(Flash.Temp) AS
TempRSd,;
        AVG(Flash.Temp)*0.1141-85.024 AS TempC;
        INTO TABLE &mVolSummary;
        HAVING Number>15;
        FROM &mFile AS Flash;
        GROUP BY Flash.samdat, Flash.sol, Flash.site;
        ORDER BY Flash.samdat
    ENDIF
RETURN
```

D.28. FlowRate

```
PROCEDURE FLOWRATE

PARAMETERS mSwap
PUBLIC mSite, mSeason, mVolsummary, mFile, mThisSeason, mLogFile, mhigh, mFlashDir
PUBLIC xSite
DIMENSION QTR(4)

ON ERROR DO errhand WITH ;
    ERROR( ), MESSAGE( ), MESSAGE(1), PROGRAM( ), LINENO( )

SET TALK OFF
SET SAFETY OFF

QTR(1) = "A"
QTR(2) = "B"
QTR(3) = "C"
QTR(4) = "D"
iSeason=mSeason
mhigh=0

CLEAR
IF NOT mSwap
    CLOSE DATABASES ALL
    STORE SYS(5)+SYS(2003) TO DefaultDir
    mGETENV = 'M:\zDEFAULT\FOXPRO\'
    mVolsummary = mGETENV + "volsummary.dbf"
    mRootPath = "U:\IMPROVE\"
    mLogPath = mRootPath + "logs\"
    mCalPath = mRootPath + "SiteCal\"
    mSiteFile = mCalPath + "sitefile.dbf"
    mLogFile = "M:\IMPROVE\LOGS\logs.dbf"
ENDIF

*SET DEFA TO M:\IMPROVE\PROGRAMS
SET DEFAULT TO &mGETENV
*ON ERROR DO QLOGSERR
GO TOP
IF NOT mSwap
    USE &mLogFile IN 0 ALIAS logs
    mSite = logs.site
    mDate = logs.samdat
    USE
ENDIF

IF SELECT("Site")>0
    SELECT site
    USE
ENDIF
IF FILE("site.dbf")
    DELETE FILE "site.dbf"
ENDIF
SELECT DISTINCT LogSites.site;
    FROM LogSites;
    ORDER BY LogSites.site;
    INTO TABLE site
SELECT site
INDEX ON site TAG site
GO TOP
IF site<>" ALL"
    INSERT INTO site (site) VALUES (" ALL")
ENDIF
SELECT site
LOCATE FOR site.site=mSite

mMonth = MONTH(mDate)
mYear = YEAR(mDate)
mQtr = QTR(mMonth/3) + RIGHT(STR(mYear),2)
IF NOT mSwap
```

```

        mSeason = mQtr
        CurrSeas = mSeason
ENDIF
STORE "U:\IMPROVE\FLASHDBF\" TO mFlashDir
IF SELECT("volsummary")>0
    SELECT volsummary
    USE
ENDIF
IF FILE('&mvolsummary')
    DELETE FILE &mVolsummary.DBF
ENDIF

DO WHILE .T.
    SELECT site
    LOCATE FOR site.site=msite
    IF mSite = " ALL"
        WAIT WINDOW AT 15,30 NOWAIT "Please wait..."
        DO ALLFLOWS
        SET DEFAULT TO &DefaultDir
        SELECT site
        GO top
        mSite=site.site
    *CANCEL
    ELSE
        xSite = IIF(RIGHT(mSite,1) = "X", mSite, " ")
        mSite = IIF(RIGHT(mSite,1) = "X", LEFT(mSite,4) + "1", mSite)
        mFile = mFlashDir + mSite + mSeason + ".dbf"
        IF SELECT("Flash")>0
            SELECT Flash
            USE
        ENDIF
        IF NOT FILE("&mFile")      && Display message if Flashcard file does not exist
            WAIT WINDOW AT 11, 80 NOWAIT TIMEOUT 5 "File &mFile does not exist"
        * LOOP
        * ENDIF
        ELSE
            USE "&mFile" IN 0 ALIAS Flash
            DO sqlflow
        * Zero Channel E readings if there is no X site to correspond with this one.
            SELECT site
            STORE RECNO() TO mRecord
            IF NOT SEEK(LEFT(mSite,4) + "X")
                SELECT volsummary
                REPLACE ALL Emagrsd WITH 0
                REPLACE ALL Evacrsd WITH 0
        * DELETE ALL FOR Amagrsd<5 AND Bmagrsd<5 AND Cmagrsd<5 AND Dvacrsd<5 AND
            Emagrsd<5 AND Evacrsd<5
                PACK
            ENDIF
            SELECT site
            GOTO mRecord
            SELECT volsummary
            REPLACE ALL RECNO WITH RECNO()
            GO TOP
            DO FlowCalc
            SELECT Flash
            USE
        ENDIF
    ENDIF
    SELECT site
    LOCATE FOR site.site=msite

    frmInput = CREATEOBJECT('Form')
    frmInput.CLOSABLE = .F.
    frmInput.MOVABLE = .T.
    frmInput.BORDERSTYLE = 3
    frmInput.MOVE(100,10,820,780)
    frmInput.CAPTION = "Flow Transducer Daily Average"
    frmInput.NAME = "Form1"
    frmInput.ADDOBJECT('Label2', 'SeasonLabel')
    frmInput.ADDOBJECT('mSeason', 'Seasonbox')
    frmInput.ADDOBJECT('Label1', 'SiteLabel')

```

```

frmInput.ADDOBJECT('mSite', 'Sitebox')
frmInput.ADDOBJECT('Command1', 'ContinueBtn')
frmInput.ADDOBJECT('QuitBtn', 'QuitBtn1')
frmInput.ADDOBJECT('Grid1', '1stGrid')
frmInput.Label1.VISIBLE = .T.
frmInput.mSite.VISIBLE = .T.
frmInput.Label2.VISIBLE = .T.
frmInput.mSeason.VISIBLE = .F.
ON KEY LABEL F2 DO SeasonChange
frmInput.Label2.CAPTION = "Season:          " + m.mSeason + CHR(13) + "F2 to change
"
*frmInput.Label2.refresh
frmInput.Command1.VISIBLE = .T.
frmInput.QuitBtn.VISIBLE = .T.
frmInput.Grid1.VISIBLE = .T.
mhigh = 5
frmInput.Grid1.COLUMNS(6).DYNAMICFORECOLOR = ;
    "IIF(volsummary.amagrsd >= mHigh , RGB(0,0,255), RGB(96,96,96))"
frmInput.Grid1.COLUMNS(7).DYNAMICFORECOLOR = ;
    "IIF(volsummary.avacrsd >= mHigh , RGB(0,0,255), RGB(96,96,96))"
frmInput.Grid1.COLUMNS(8).DYNAMICFORECOLOR = ;
    "IIF(volsummary.bmagrsd >= mHigh , RGB(0,0,255), RGB(96,96,96))"
frmInput.Grid1.COLUMNS(9).DYNAMICFORECOLOR = ;
    "IIF(volsummary.bvacrsd >= mHigh , RGB(0,0,255), RGB(96,96,96))"
frmInput.Grid1.COLUMNS(10).DYNAMICFORECOLOR = ;
    "IIF(volsummary.cmagrsd >= mHigh , RGB(0,0,255), RGB(96,96,96))"
frmInput.Grid1.COLUMNS(11).DYNAMICFORECOLOR = ;
    "IIF(volsummary.cvacrsd >= mHigh , RGB(0,0,255), RGB(96,96,96))"
frmInput.Grid1.COLUMNS(12).DYNAMICFORECOLOR = ;
    "IIF(volsummary.dvacrsd >= 2, RGB(0,0,255), RGB(96,96,96))"
frmInput.Grid1.COLUMNS(13).DYNAMICFORECOLOR = ;
    "IIF(volsummary.emagrsd < .005, RGB(255,255,255), IIF(volsummary.emagrsd >= 5,
RGB(0,0,255), RGB(96,96,96)))"
frmInput.Grid1.COLUMNS(14).DYNAMICFORECOLOR = ;
    "IIF(volsummary.evacrsd < .005, RGB(255,255,255), IIF(volsummary.evacrsd >= 5,
RGB(0,0,255), RGB(96,96,96)))"
frmInput.Grid1.COLUMNS(15).DYNAMICFORECOLOR = ;
    "IIF(volsummary.TempC >30, RGB(255,0,0), IIF(volsummary.TempC <5, RGB(0,128,0),
RGB(0,0,255)))"
frmInput.Grid1.COLUMNS(6).DYNAMICFONTBOLD = "IIF(volsummary.amagrsd >= 40, .T.,
.F.)"
frmInput.Grid1.COLUMNS(7).DYNAMICFONTBOLD = "IIF(volsummary.avacrsd >= 40, .T.,
.F.)"
frmInput.Grid1.COLUMNS(8).DYNAMICFONTBOLD = "IIF(volsummary.bmagrsd >= 40, .T.,
.F.)"
frmInput.Grid1.COLUMNS(9).DYNAMICFONTBOLD = "IIF(volsummary.bvacrsd >= 40, .T.,
.F.)"
frmInput.Grid1.COLUMNS(10).DYNAMICFONTBOLD = "IIF(volsummary.cmagrsd >= 40, .T.,
.F.)"
frmInput.Grid1.COLUMNS(11).DYNAMICFONTBOLD = "IIF(volsummary.cvacrsd >= 40, .T.,
.F.)"
frmInput.Grid1.COLUMNS(12).DYNAMICFONTBOLD = "IIF(volsummary.dvacrsd >= 40, .T.,
.F.)"
frmInput.Grid1.COLUMNS(13).DYNAMICFONTBOLD = "IIF(volsummary.emagrsd >= 40, .T.,
.F.)"
frmInput.Grid1.COLUMNS(14).DYNAMICFONTBOLD = "IIF(volsummary.evacrsd >= 40, .T.,
.F.)"
frmInput.Grid1.COLUMNS(15).DYNAMICFONTBOLD = "IIF(volsummary.TempC < 5 OR
volsummary.TempC >= 30, .T., .F.)"
frmInput.Grid1.COLUMNS(6).INPUTMASK="###.#"
frmInput.Grid1.COLUMNS(7).INPUTMASK="###.#"
frmInput.Grid1.COLUMNS(8).INPUTMASK="###.#"
frmInput.Grid1.COLUMNS(9).INPUTMASK="###.#"
frmInput.Grid1.COLUMNS(10).INPUTMASK="###.#"
frmInput.Grid1.COLUMNS(11).INPUTMASK="###.#"
frmInput.Grid1.COLUMNS(12).INPUTMASK="###.#"
frmInput.Grid1.COLUMNS(13).INPUTMASK="###.#"
frmInput.Grid1.COLUMNS(14).INPUTMASK="###.#"
frmInput.Grid1.COLUMNS(15).INPUTMASK="###.#"
frmInput.Grid1.COLUMNS(1).Header1.CAPTION = "Record"
frmInput.Grid1.COLUMNS(2).Header1.CAPTION = "Site"
frmInput.Grid1.COLUMNS(3).Header1.CAPTION = "Date"

```

```

frmInput.Grid1.COLUMNS(4).Header1.CAPTION = "Solenoid"
frmInput.Grid1.COLUMNS(5).Header1.CAPTION = "Number"
frmInput.Grid1.COLUMNS(6).Header1.CAPTION = "A Mag "
frmInput.Grid1.COLUMNS(7).Header1.CAPTION = "A Vac "
frmInput.Grid1.COLUMNS(8).Header1.CAPTION = "B Mag "
frmInput.Grid1.COLUMNS(9).Header1.CAPTION = "B Vac "
frmInput.Grid1.COLUMNS(10).Header1.CAPTION = "C Mag "
frmInput.Grid1.COLUMNS(11).Header1.CAPTION = "C Vac "
frmInput.Grid1.COLUMNS(12).Header1.CAPTION = "D Vac "
frmInput.Grid1.COLUMNS(13).Header1.CAPTION = "E Mag "
frmInput.Grid1.COLUMNS(14).Header1.CAPTION = "E Vac "
frmInput.Grid1.COLUMNS(15).Header1.CAPTION = "Temp C"
frmInput.Grid1.COLUMNS(1).Header1.ALIGNMENT = 2
frmInput.Grid1.COLUMNS(2).Header1.ALIGNMENT = 2
frmInput.Grid1.COLUMNS(3).Header1.ALIGNMENT = 2
frmInput.Grid1.COLUMNS(4).Header1.ALIGNMENT = 2
frmInput.Grid1.COLUMNS(5).Header1.ALIGNMENT = 2
frmInput.Grid1.COLUMNS(6).Header1.ALIGNMENT = 1
frmInput.Grid1.COLUMNS(7).Header1.ALIGNMENT = 1
frmInput.Grid1.COLUMNS(8).Header1.ALIGNMENT = 1
frmInput.Grid1.COLUMNS(9).Header1.ALIGNMENT = 1
frmInput.Grid1.COLUMNS(10).Header1.ALIGNMENT = 1
frmInput.Grid1.COLUMNS(11).Header1.ALIGNMENT = 1
frmInput.Grid1.COLUMNS(12).Header1.ALIGNMENT = 1
frmInput.Grid1.COLUMNS(13).Header1.ALIGNMENT = 1
frmInput.Grid1.COLUMNS(14).Header1.ALIGNMENT = 1
frmInput.Grid1.COLUMNS(15).Header1.ALIGNMENT = 1
frmInput.Grid1.COLUMNS(1).WIDTH = 55
frmInput.Grid1.COLUMNS(2).WIDTH = 45
frmInput.Grid1.COLUMNS(3).WIDTH = 65
frmInput.Grid1.COLUMNS(4).WIDTH = 45
frmInput.Grid1.COLUMNS(5).WIDTH = 45
frmInput.Grid1.COLUMNS(6).WIDTH = 40
frmInput.Grid1.COLUMNS(7).WIDTH = 40
frmInput.Grid1.COLUMNS(8).WIDTH = 40
frmInput.Grid1.COLUMNS(9).WIDTH = 40
frmInput.Grid1.COLUMNS(10).WIDTH = 40
frmInput.Grid1.COLUMNS(11).WIDTH = 40
frmInput.Grid1.COLUMNS(12).WIDTH = 40
frmInput.Grid1.COLUMNS(13).WIDTH = 40
frmInput.Grid1.COLUMNS(14).WIDTH = 40
frmInput.Grid1.COLUMNS(15).WIDTH = 65

frmInput.mSite.VISIBLE = .T.
SELECT site
LOCATE FOR site.site=mSite
frmInput.mSite.VALUE = mSite
frmInput.SHOW
READ EVENTS

ENDDO
ON ERROR  && restore system error handler
SET DEFAULT TO &DefaultDir
SELECT site
USE

* Object definitions for display form follow
DEFINE CLASS SeasonLabel AS LABEL
    AUTOSIZE = .F.
    FONTBOLD = .T.
    FONTSIZE = 9
    ALIGNMENT = 1
    CAPTION = "Season:          " + m.mSeason + CHR(13) + "F2 to change          "
    LEFT = 50
    TOP = 25
    HEIGHT = 30
    WIDTH = 98
    NAME = "Label2"
ENDDDEFINE

DEFINE CLASS SeasonBox AS TEXTBOX
    FONTBOLD = .T.

```

```

    FONTSIZE = 9
    ALIGNMENT = 1
    VALUE = m.mSeason
    SELSTART = 0
    SELLENGTH = 3
    SELECTIONENTRY = .T.
    CONTROLSOURCE = "m.mSeason"
    FORMAT = "!"
    TOP = 24
    LEFT = 126
    HEIGHT = 25
    WIDTH = 44
    MAXLENGTH = 3
    SPECIALEFFECT = 0
    NAME = "mSeason"
    PROCEDURE VALID
    IF mSeason<>iSeason
        IF mSeason = CurrSeas
            mLogFile = mLogsPath + "logs.dbf"
        ELSE
            mLogFile = mLogsPath + mSeason + "logs.dbf"
        ENDIF
        mBadLog = .F.
    *   ON ERROR DO QLOGSERR
        IF SELECT("logs")<>0
            SELECT logs
            USE
        ENDIF
        USE &mLogFile IN 0 ALIAS logs
    *   On Error
        IF NOT mBadLog
            IF SELECT("site")<>0
                SELECT site
                USE
            ENDIF
            SET DEFAULT TO &mGETENV

            SELECT DISTINCT logs.site;
            FROM logs;
            ORDER BY logs.site;
            INTO TABLE site

            SELECT Site
            INDEX ON site TAG site
            GO TOP
            IF site<>" ALL"
                INSERT INTO site (site) VALUES (" ALL")
            ENDIF
            LOCATE FOR site.site = mSite
            SELECT logs
            USE
            SELECT site
            SET DEFAULT TO &DefaultDir
        ENDIF
    ENDIF
    frmInput.mSite.VISIBLE = .T.
    frmInput.REFRESH
    frmInput.mSeason.VISIBLE = .F.
    frmInput.Label2.CAPTION = "Season:          " + m.mSeason + CHR(13) + "F2 to change
    "
    ENDPROC
ENDDDEFINE

DEFINE CLASS SiteLabel AS LABEL
    AUTOSIZE = .F.
    FONTBOLD = .T.
    FONTSIZE = 9
    ALIGNMENT = 1
    CAPTION = "Site Name:"+CHR(13)+"('ALL' for report)"
    TOP = 28
    LEFT = 190
    HEIGHT = 30
    WIDTH = 88

```

```

NAME = "Label1"
ENDDDEFINE

DEFINE CLASS SiteBox AS LISTBOX
  FONTBOLD = .T.
  FONTSIZE = 8
  VALUE = mSite
  CONTROLSOURCE = Site.site
  TOP = 10
  LEFT = 290
  WIDTH = 75
  HEIGHT = 50
  MAXLENGTH = 5
  SPECIALEFFECT = 0
  COLUMNCOUNT = 1
  NAME = "mSite"
  SELSTART = 1
  SELLENGTH = 5
  SELECTONENTRY = .T.
  ROWSOURCETYPE = 2
  ROWSOURCE = "site"
  DISPLAYVALUE = 1
  NUMBEROFELEMENTS = 180
  INPUTMASK = "!!!!!"
  READONLY = .F.
  INCREMENTALSEARCH = .T.
  PROCEDURE VALID
    m.mSite = site.site
    CLEAR EVENTS
    THISFORM.RELEASE
    WAIT CLEAR
    IF SELECT("Flash")>0
      SELECT Flash
    USE
  ENDIF
ENDPROC
ENDDDEFINE

DEFINE CLASS ContinueBtn AS COMMANDBUTTON
  TOP = 24
  LEFT = 390
  HEIGHT = 25
  WIDTH = 121
  FONTBOLD = .T.
  FONTSIZE = 10
  CANCEL = .F.
  CAPTION = "\<Continue"
  DEFAULT = .T.
  TERMINATEREAD = .F.
  NAME = "Command1"

  PROCEDURE CLICK
*   IIF(len(msite)=0, m.msite=" ", m.msite = site.site)
  m.mSite = site.site
  CLEAR EVENTS
  THISFORM.RELEASE
  WAIT CLEAR
  IF SELECT("Flash")>0
    SELECT Flash
  USE
  ENDIF
ENDPROC
ENDDDEFINE

DEFINE CLASS QuitBtn1 AS COMMANDBUTTON
  TOP = 24
  LEFT = 540
  HEIGHT = 25
  WIDTH = 121
  FONTBOLD = .T.
  FONTSIZE = 10

```



```

CANCEL = .T.
CAPTION = "\<Quit"
TERMINATEREAD = .F.
NAME = "QuitBtn"
PROCEDURE CLICK
IF SELECT("Flash")>0
    SELECT Flash
    USE
ENDIF
ON ERROR
SET DEFAULT TO &DefaultDir
SET SAFETY ON
DEACTIVATE WINDOW ALL
*   cancel
SET SAFETY OFF
RETURN TO MASTER
ENDPROC
ENDEDEFINE

DEFINE CLASS 1stGrid AS GRID
TOP = 70
LEFT = 12
HEIGHT = 700
WIDTH = 800
RECORDSOURCE = "Volsummary"
NAME = "Grid1"
COLUMNCOUNT = 15
Column1.ALIGNMENT = 2
Column2.ALIGNMENT = 2
Column3.ALIGNMENT = 2
Column4.ALIGNMENT = 2
Column5.ALIGNMENT = 2
ENDEDEFINE

PROCEDURE SeasonChange
mSite = IIF(RIGHT(mSite,1) = "X", LEFT(mSite,4) + "1", mSite)
mFile = mFlashDir + mSite + mSeason + ".dbf"
frmInput.mSite.VISIBLE = .T.
frmInput.mSeason.VISIBLE = .T.
frmInput.mSeason.SETFOCUS
frmInput.REFRESH
RETURN

```

D.29. AllFlows

```
PROCEDURE AllFlows
PUBLIC MissData(15)

mRootPath = "U:\IMPROVE\"
mCalPath = mRootPath + "SiteCal\"
mSiteFile = mCalPath + "sitefile.dbf"
mOutPath = mRootPath + "FlashDBF\Reports\"
mOutFile = mOutPath + "&mSeason" + "Flows.dbf"
Tempout = "m:\improve\temp\tOutfile.dbf"

* Set up output file and initialize array for scatter data
IF SELECT("Out")>0
    SELECT Out
    USE
ENDIF
SELECT 0
*CREATE &mOutFile
CREATE TABLE &mOutFile (Recno I,Site C(5), Date D, Solenoid I, Number I, ;
    AMagRsd N(20,5), AVacRsd N(20,5),;
    BMagRsd N(20,5), BVacRsd N(20,5),;
    CMagRsd N(20,5), CVacRsd N(20,5),;
    DVacRsd N(20,5),;
    EMagRsd N(20,5), EVacRsd N(20,5),;
    TempRsd N(20,5), TempC N(20,5))
APPEND BLANK
COPY TO &Tempout FOX2X
USE &Tempout ALIAS out
GO BOTTOM
SCATTER TO MissData
DELETE
PACK

SELECT logsites
DO WHILE NOT EOF('logsites')
    mSite = logsites.Site
    IF RIGHT(mSite,1)="X"
        SKIP IN logsites
    LOOP
ENDIF
mFile = mFlashDir + mSite + mSeason + ".dbf"
IF SELECT("Flash")>0
    SELECT Flash
    USE
ENDIF
IF NOT FILE("&mFile")
    Missdata(1) = 0
    MissData(2) = mSite
    MissData(4) = 0
    MissData(5) = 0
    MissData(6) = -99
    MissData(7) = -99
    MissData(8) = -99
    MissData(9) = -99
    MissData(10) = -99
    MissData(11) = -99
    MissData(12) = -99
    MissData(13) = -99
    MissData(14) = -99
    MissData(15) = -99
    SELECT out
    APPEND BLANK
    GO BOTTOM
    GATHER FROM MissData
*   SELECT sites
*   SKIP
*   LOOP
*   ENDIF
ELSE
```

```

        USE "&mFile" IN 0 ALIAS Flash
        DO SQLflow
*       Zero Channel E readings if there is no X site to correspond with this one.
        SELECT logsites
        STORE RECNO() TO mRecord
        IF NOT SEEK(LEFT(mSite,4) + "X")
            SELECT Volsummary
            REPLACE ALL EMagRSD WITH 0
            REPLACE ALL EVacRSD WITH 0
        ENDIF
        SELECT logsites
        GOTO mRecord
        SELECT Volsummary
        REPLACE ALL Recno with Recno()
        USE
        SELECT out
        APPEND FROM &mVolsummary
        SELECT Flash
        USE
    ENDIF
    SELECT logsites
    SKIP
ENDDO
USE &mOutFile in 0 Alias Keep
SELECT Keep
DELETE ALL
PACK
APPEND FROM &Tempout
USE
*SELECT Sites
*USE
SELECT out
*SET FILTER TO date>={8/1/2004}
*REPORT FORM Flow for (AMagRsd>5 OR AVacRsd>5 OR BMagRsd>5 OR BVacRSD>5 OR CMagRsd>5
    OR CVacRsd>5 OR DVacRsd>3 OR EMagRsd>5 OR EVacRsd>5 OR Number<72 OR Number>104)
    Preview
REPORT FORM Flow for site<>" ALL" Preview
USE
DELETE FILE &Tempout

RETURN

```

D.30. SQLFlow

PROCEDURE SQLFLOW

* Extracts data from FlashCard on daily variability

```
IF RIGHT(mSite,1)="9"
    SELECT recno() AS RecNo, Flash.Site AS Site, Flash.samdat AS Date,
Flash.sol AS Solenoid,;
    COUNT(Flash.af1m) AS Number,;
    0 AS Amagrsd,;
    AVG(Flash.af1g)/100 AS Avacrsd,;
    0 AS Bmagrsd,;
    AVG(Flash.bf1g)/100 AS Bvacrsd,;
    0 AS Cmagrsd,;
    AVG(Flash.cf1g)/100 AS Cvacrsd,;
    0 AS Dvacrsd,;
    0 AS Emagrsd,;
    0 AS Evacrsd,;
    AVG(Flash.Temp)*0.1141-85.024 AS TempC;
    INTO TABLE &mVolSummary;
    FROM &mFile AS Flash;
    GROUP BY Flash.samdat, Flash.sol, Flash.Site;
    HAVING Number>12;
    ORDER BY Flash.samdat
ELSE
    SELECT recno() AS RecNo, Flash.Site AS Site, Flash.samdat AS Date, Flash.sol AS
Solenoid,;
    COUNT(Flash.af1m) AS Number,;
    AVG(Flash.af1m)/100 AS Amagrsd,;
    AVG(Flash.af1g)/100 AS Avacrsd,;
    AVG(Flash.bf1m)/100 AS Bmagrsd,;
    AVG(Flash.bf1g)/100 AS Bvacrsd,;
    AVG(Flash.cf1m)/100 AS Cmagrsd,;
    AVG(Flash.cf1g)/100 AS Cvacrsd,;
    AVG(Flash.df1g)/100 AS Dvacrsd,;
    AVG(Flash.ef1m)/100 AS Emagrsd,;
    AVG(Flash.ef1g)/100 AS Evacrsd,;
    AVG(Flash.Temp)*0.1141-85.024 AS TempC;
    INTO TABLE &mVolSummary;
    HAVING Number>12;
    FROM &mFile Flash;
    GROUP BY Flash.samdat, Flash.sol, Flash.Site;
    ORDER BY Flash.samdat
ENDIF
* AND (AMagRSd>5 OR BMagRSd>5 OR CMagRSd>5 OR DVacRSd>5 OR EMagRSd>5 OR EVacRSd>5)
RETURN
```

D.31. FlowCalc

```
PROCEDURE FlowCalc
*
* Calculate flow rates from flashcard averages for display in SWAP
*
SELECT volsummary
GO BOTTOM
STORE DATE TO lastdate

MagCal = mcalpath + "Impmgcal.dbf"
VacCal = mcalpath + "Imppscal.dbf"

USE &MagCal IN 0 ALIAS MagC
SELECT MagC.sitecode, MagC.channel, MagC.elevfact,;
      MagC.tempc, MagC.DATE, MagC.a_davis, MagC.b_davis,;
      MagC.nom_flow, MagC.sampler, MagC.TYPE, MagC.COMMENT;
FROM MagC;
WHERE MagC.sitecode = msite;
AND MagC.DATE < lastdate;
INTO CURSOR Mag
SELECT MagC
USE

USE &VacCal IN 0 ALIAS VacC
SELECT VacC.sitecode, VacC.channel, VacC.elevfact,;
      VacC.tempc, VacC.DATE, VacC.a_davis, VacC.b_davis,;
      VacC.nom_flow, VacC.sampler, VacC.Inlet, VacC.TYPE, VacC.COMMENT;
FROM VacC;
WHERE VacC.sitecode = msite;
AND VacC.DATE < lastdate;
INTO CURSOR Vac
SELECT VacC
USE

SELECT volsummary
GO TOP

TempR = 293

DO WHILE NOT EOF("volsummary")
  SELECT Mag
  SET FILTER TO DATE<volsummary.DATE AND channel="A1"
  GO BOTTOM
  SELECT Vac
  SET FILTER TO DATE<volsummary.DATE AND channel="A1"
  GO BOTTOM
  SELECT volsummary
  magFLOWA = ROUND(10^(Mag.a_davis+Mag.b_davis*LOG10(volsummary.Amagrsd+0.00001))*;
    Mag.elevfact*SQRT((volsummary.tempc+273.)/(TempR)),1)
  vacFLOWA = ROUND((Vac.a_davis+Vac.b_davis*(volsummary.Avacrsd+0.00001))*;
    Vac.elevfact*SQRT((volsummary.tempc+273.)/(TempR)),1)
  REPLACE volsummary.Amagrsd WITH magFLOWA
  REPLACE volsummary.Avacrsd WITH vacFLOWA

  SELECT Mag
  SET FILTER TO DATE<volsummary.DATE AND channel="B1"
  GO BOTTOM
  SELECT Vac
  SET FILTER TO DATE<volsummary.DATE AND channel="B1"
  GO BOTTOM
  SELECT volsummary
  magFLOWB = ROUND(10^(Mag.a_davis+Mag.b_davis*LOG10(volsummary.Bmagrsd+0.00001))*;
    Mag.elevfact*SQRT((volsummary.tempc+273.)/(TempR)),1)
  vacFLOWB = ROUND((Vac.a_davis+Vac.b_davis*(volsummary.Bvacrsd+0.00001))*;
    Vac.elevfact*SQRT((volsummary.tempc+273.)/(TempR)),1)
  REPLACE volsummary.Bmagrsd WITH magFLOWB
  REPLACE volsummary.Bvacrsd WITH vacFLOWB

  SELECT Mag
```

```

SET FILTER TO DATE<volsummary.DATE AND channel="C1"
GO BOTTOM
SELECT Vac
SET FILTER TO DATE<volsummary.DATE AND channel="C1"
GO BOTTOM
SELECT volsummary
magFLOWC = ROUND(10^(Mag.a_davis+Mag.b_davis*LOG10(volsummary.Cmagrsd+0.00001))* ;
    Mag.elevfact*SQRT((volsummary.tempc+273.)/(TempR)),1)
vacFLOWC = ROUND((Vac.a_davis+Vac.b_davis*(volsummary.Cvacrsd+0.00001))* ;
    Vac.elevfact*SQRT((volsummary.tempc+273.)/(TempR)),1)
REPLACE volsummary.Cmagrsd WITH magFLOWC
REPLACE volsummary.Cvacrsd WITH vacFLOWC

SELECT Vac
SET FILTER TO DATE<volsummary.DATE AND channel="D1"
GO BOTTOM
SELECT volsummary
vacFLOWD = ROUND((Vac.a_davis+Vac.b_davis*(volsummary.Dvacrsd+0.00001))* ;
    Vac.elevfact*SQRT((volsummary.tempc+273.)/(TempR)),1)
REPLACE volsummary.Dvacrsd WITH vacFLOWD

SKIP IN volsummary
ENDDO

IF xSite<>" "
USE &MagCal IN 0 ALIAS MagC
SELECT MagC.sitecode, MagC.channel, MagC.elevfact, ;
    MagC.tempc, MagC.DATE, MagC.a_davis, MagC.b_davis, ;
    MagC.nom_flow, MagC.sampler, MagC.TYPE, MagC.COMMENT;
FROM MagC;
WHERE MagC.sitecode = xSite;
AND MagC.DATE < lastdate;
INTO CURSOR Mag
SELECT MagC
USE

USE &VacCal IN 0 ALIAS VacC
SELECT VacC.sitecode, VacC.channel, VacC.elevfact, ;
    VacC.tempc, VacC.DATE, VacC.a_davis, VacC.b_davis, ;
    VacC.nom_flow, VacC.sampler, VacC.Inlet, VacC.TYPE, VacC.COMMENT;
FROM VacC;
WHERE VacC.sitecode = xSite;
AND VacC.DATE < lastdate;
INTO CURSOR Vac
SELECT VacC
USE

SELECT volsummary
GO TOP

DO WHILE NOT EOF("volsummary")
    SELECT Mag
    SET FILTER TO DATE<volsummary.DATE
    GO BOTTOM
    SELECT Vac
    SET FILTER TO DATE<volsummary.DATE
    GO BOTTOM
    SELECT volsummary
    IF vac.channel="D1"
    *     REPLACE volsummary.Emagrsd WITH 0
    *
    *     ELSE
    *         magFLOWE =
    *         ROUND(10^(Mag.a_davis+Mag.b_davis*LOG10(volsummary.Emagrsd+0.00001))* ;
    *             Mag.elevfact*SQRT((volsummary.tempc+273.)/(TempR)),1)
    *         REPLACE volsummary.Emagrsd WITH magFLOWE
    *     ENDIF
    vacFLOWE = ROUND((Vac.a_davis+Vac.b_davis*(volsummary.Evacrsd+0.00001))* ;
    Vac.elevfact*SQRT((volsummary.tempc+273.)/(TempR)),1)
    REPLACE volsummary.Evacrsd WITH vacFLOWE
    SKIP
ENDDO
ENDIF

```

```
GO TOP
SELECT Mag
USE
SELECT Vac
USE

RETURN
```

D.32. SwapCheck

```
PROCEDURE swapcheck && Checks output files for potential filter swaps

PARAMETERS mSwap
PUBLIC mCalled
mCalled = mSwap
SET DEFAULT TO M:\IMPROVE\Programs
IF NOT mCalled
    CLOSE DATABASES ALL
ENDIF
*USE m:\improve\temp\d04qa12.DBF ALIAS out
*GO TOP

DO GetMonth
outfile = "m:\improve\temp\" + mSeason + "QA" + mMonth
IndexFile = outfile + ".cdx"
IF FILE('&IndexFile')
    USE &OutFile IN 0 ALIAS out
    REINDEX
ELSE
    USE &OutFile IN 0 ALIAS out
    INDEX ON site + DTOC(samdat) TAG sitdat
ENDIF

IF SELECT('swaps')>0
    SELECT swaps
    USE
ENDIF
IF FILE("m:\improve\temp\swaps.dbf")
    DELETE FILE M:\IMPROVE\temp\swaps.DBF
ENDIF
CREATE TABLE M:\IMPROVE\temp\swaps (site c(5), samdat d, ;
    Prev_Ratio N(6,2), This_Ratio N(6,2), SwIn1 N(6,2), SwIn2 N(6,2), MODULE c(3))
SELECT out
*INDEX ON site+DTOC(samdat) TAG sitdat
SET ORDER TO sitdat
GO TOP

* For A-B mods, set index1<-.05 and ABS(index2)>.05
mIndex1 = -.03
mIndex2 = .05
LowRatio = .667
HiRatio = 1.5
mPar1 = "out.s3"
mPar2 = "out.so4"
mPar1a = "0"
mPar2a = "0"
mModule = "A-B"
FOR nTimes = 1 TO 2
    SELECT out
    GO TOP
    DO WHILE NOT EOF('out')
        msite=out.site
        Var1_1 = IIF(&mPar1>0 .and. &mPar1a>=0, &mPar1 + &mPar1a, -999)
        Var2_1 = IIF(&mPar2>0 .and. &mPar2a>=0, &mPar2 + &mPar2a, -999)
        SKIP
        DO WHILE out.site=msite
            Var1_2 = IIF(&mPar1>0 .and. &mPar1a>=0, &mPar1 + &mPar1a, -999)
            Var2_2 = IIF(&mPar2>0 .and. &mPar2a>=0, &mPar2 + &mPar2a, -999)
            sRatio_1 = IIF(Var1_1>0 AND Var2_1>0, Var1_1/Var2_1, 1)
            sRatio_2 = IIF(Var1_2>0 AND Var2_2>0, Var1_2/Var2_2, 1)
            lratio = NOT (BETWEEN(sRatio_1, LowRatio, HiRatio) .AND.
                BETWEEN(sRatio_2, LowRatio, HiRatio))
            SwIndex1 = (sRatio_1-1)*(sRatio_2-1)
        * SwIndex = (sRatio_1-1)*(sRatio_2-1)
        * IF SwIndex < -.05
            sRatio_3 = IIF(Var1_1>0 AND Var2_2>0, Var1_1/Var2_2, 1)
            sRatio_4 = IIF(Var1_2>0 AND Var2_1>0, Var1_2/Var2_1, 1)
            SwIndex2 = (sRatio_3-1)*(sRatio_4-1)
```



```

        IF (SwIndex1 < mIndex1 AND ABS(SwIndex2) < mIndex2) OR lratio
            SELECT swaps
            APPEND BLANK
            GO BOTTOM
            REPLACE site WITH out.site, samdat WITH out.samdat, MODULE WITH
mModule
                IF sRatio_1<>1
                    REPLACE Prev_Ratio WITH sRatio_1
                ENDIF
                IF sRatio_2<>1
                    REPLACE This_Ratio WITH sRatio_2
                ENDIF
                IF SwIndex1<>0
                    REPLACE SwIn1 WITH SwIndex1
                    REPLACE SwIn2 WITH SwIndex2
                ENDIF
            SELECT out
        ENDIF
        Var1_1 = Var1_2
        Var2_1 = Var2_2
        SKIP
    ENDDO
    ENDDO
* For A-C mods, set index1<-1 and ABS(index2)>.05
mIndex1 = -1
mIndex2 = .05
LowRatio = .5
HiRatio = 4
*mPar1 = "out.omh"
*mPar2 = "out.omcn"
mPar1 = "out.omh"
mPar1a = "out.lrnrc"
mPar2 = "out.omcn"
mpar2a = "out.lacn"
mModule = "A-C"
ENDFOR

*RETURN
SELECT swaps
*BROWSE LAST
REPORT FORM swaps PREVIEW
RETURN

```

D.33. GetMonth

* This procedure displays a window to select a month and year to build
* the IMPROVE database.

*
* Parameters returned:
* mMonth: Character (width 2) e.g. "12" for December
* mQTR: Character (width 1) e.g. "B" for summer season
* mSeason: Character (width 3) e.b. "B03" for summer 2003
*

PROCEDURE getmonth

**CLEAR

PUBLIC mMonth, mQtr, mSeason, aMonth, mYear
PUBLIC mPMonth, mYearMonth
STORE "00" TO mMonth
STORE "000" TO mSeason
DIMENSION gaMonth(12), gaYear(20), cQtr(12)
STORE "0000" TO mYear

cQtr(12) = "D"
cQtr(1) = "D"
cQtr(2) = "D"
cQtr(3) = "A"
cQtr(4) = "A"
cQtr(5) = "A"
cQtr(6) = "B"
cQtr(7) = "B"
cQtr(8) = "B"
cQtr(9) = "C"
cQtr(10) = "C"
cQtr(11) = "C"

FOR gnCount = 1 TO 12 && Fill the month array
STORE cmonth(date(year(date()),gncount,1)) TO gaMonth(gnCount)
NEXT
FOR gnCount = 1 TO 20 && Fill the year array
STORE str(1987+gncount,4,0) TO gaYear(gnCount)
NEXT

frmMyForm = CREATEOBJECT('Form') && Create a Form
frmMyForm.Caption = "Build data for month"
frmMyForm.Closable = .f. && Disable the Control menu box
frmMyForm.Move(150,10,550,400) && Move the form
frmMyForm.AddObject('ListBox1','1stMyListBox') && Add ListBox control
frmMyForm.AddObject('ListBox2','2ndMyListBox') && Add ListBox control
frmMyForm.AddObject('cmbCommand1','cmdMyCmdBtn') && Add "Continue" Command button
frmMyForm.AddObject('cmbCommand2','QuitBtn') && Add "Quit" Command button
frmMyForm.AddObject('MyLabel','Caution')
*frmMyForm.AddObject('Label1','SiteLabel')
*frmMyForm.AddObject('mSite','SiteBox')

frmMyForm.ListBox1.RowSourceType = 5 && Specifies an array
frmMyForm.ListBox1.RowSource = 'gaMonth' && Array containing listbox items
frmMyForm.ListBox2.RowSourceType = 5 && Specifies an array
frmMyForm.ListBox2.RowSource = 'gaYear' && Array containing listbox items

*frmMyForm.cmbCommand1.Visible = .T. && "Continue" Command button visible
frmMyForm.cmbCommand2.Visible = .T. && "Quit" Command button visible
frmMyForm.ListBox1.Visible = .T. && "List Box visible"
frmMyForm.MyLabel.Visible = .T.
*frmMyForm.ListBox2.Visible = .F. && "List Box not visible"

frmMyForm.SHOW && Display the form
frmMyForm.ListBox2.Visible = .F. && "List Box not visible"
READ EVENTS && Start event processing

DEFINE CLASS Caution AS Label

```

Caption = "   Be sure the following files are up to date" ;
+ CHR(13) + CHR(10) + "           - Elements" ;
+ CHR(13) + CHR(10) + "           - Ions" ;
+ CHR(13) + CHR(10) + "           - Carbon" ;
+ CHR(13) + CHR(10) + "           - Laser"
Left = 270
Top = 20
Height = 80
Width = 240
BorderStyle = 1
BackStyle = 1
FontBold = .T.
BackColor = RGB(128, 255, 255)
ForeColor = RGB(128, 0, 0)
ENDDDEFINE

DEFINE CLASS cmdMyCmdBtn AS CommandButton  && Create Command button
Caption = '\<Continue'  && Caption on the Command button
Cancel = .T.  && Default Cancel Command button (Esc)
Left = 10  && Command button column
Top = 310  && Command button row
Height = 25  && Command button height

PROCEDURE Click
  IF mMonth = "00" OR mSeason = "000"
**    CLEAR
      ? 'Select MONTH and YEAR'
      Wait
**    CLEAR
      frmMyForm.cmbCommand1.Visible = .T.  && "Continue" Command button visible
      frmMyForm.cmbCommand2.Visible = .T.  && "Quit" Command button visible
      frmMyForm.ListBox1.Visible = .T.  && "List Box visible
      frmMyForm.ListBox2.Visible = .F.  && "List Box visible
      frmMyForm.SHOW
  ELSE
    CLEAR EVENTS  && Stop event processing, close Form
  *    CLEAR  && Clear main Visual FoxPro window
      RELEASE gaMonth, gaYear, cQtr
  ENDDIF
  WAIT CLEAR
ENDDDEFINE

DEFINE CLASS QuitBtn AS CommandButton  && Create Command button
Caption = '\<Quit'  && Caption on the Command button
Cancel = .T.  && Default Cancel Command button (Esc)
Left = 10  && Command button column
Top = 260  && Command button row
Height = 25  && Command button height

PROCEDURE Click
  WAIT CLEAR
  SET SAFETY ON
  POP KEY ALL
  IF INLIST(mUser, "ashbaugh", "sengupta")
    IF mCalled
      RETURN TO master
    ENDDIF
  CANCEL
ENDDIF
QUIT
ENDPROC
ENDDDEFINE

DEFINE CLASS lstMyListBox AS ListBox  && Create ListBox control
Left = 10  && List Box column
Top = 25  && List Box row
MultiSelect = .F.  && Don't allow selecting more than 1 item
Height = 210

PROCEDURE Click
  ACTIVATE SCREEN
**  CLEAR

```

```

FOR nCnt = 1 TO ThisForm.ListBox1.ListCount
  IF ThisForm.ListBox1.Selected(nCnt)  && Is item selected?
    aMonth = gaMonth(nCnt)
    mMonth = IIF(nCnt<10,"0"+str(nCnt,1,0),str(nCnt,2,0))
    mYearMonth = mYear + mMonth
    mQtr = cQtr(nCnt)
    mPMonth = IIF(nCnt=1,"12", IIF(nCnt<11,"0"+str(nCnt-1,1,0),str(nCnt-1,2,0)))
    frmMyForm.ListBox2.Visible =.T.  && "List Box visible
  ENDIF
ENDIF
ENDFOR
ENDDDEFINE

DEFINE CLASS 2ndMyListBox AS ListBox  && Create ListBox control
  Left = 140                                && List Box column
  Top = 25                                  && List Box row
  MultiSelect = .F.                        && Don't allow selecting more than 1 item
  Height = 355

PROCEDURE Click
  ACTIVATE SCREEN
  ** CLEAR
  FOR mCnt = 1 TO ThisForm.ListBox2.ListCount
    IF ThisForm.ListBox2.Selected(mCnt)  && Is item selected?
      ? aMonth+" "+GaYear(mCnt)
      STORE GaYear(mCnt) TO mYear
      mYearMonth = mYear + mMonth
      mSeason = mQtr + IIF(val(mMonth)>2,;
      right(ThisForm.ListBox2.List(mCnt),2),;
      right(ThisForm.ListBox2.List(mCnt-1),2))
      frmMyForm.cmbCommand1.Visible =.T.  && "Continue" Command button visible
    ENDIF
  ENDFOR
ENDDDEFINE

```

D.34. ArtHist

```
PROCEDURE ArtHist
EditFile = mSuppPath + "arthist.dbf"
USE &EditFile IN 0 ALIAS Arthist
IF NOT mBadLog
  IF NOT nofile
    DEFINE WINDOW OUTPUT FROM 0,0 TO 40,190 TITLE 'Artifacts History' ;
    CLOSE FLOAT GROW ZOOM MDI FONT 'Arial', 9
    ACTIVATE WINDOW OUTPUT
    SELECT Arthist
    LOCATE FOR arthist.qtr=mSeason
    BROWSE NOEDIT ;
    WINDOW OUTPUT
    HIDE WINDOW OUTPUT
    USE
  ENDIF
ENDIF
ENDIF
```

D.35. ClearQD

* This program clears all remaining QDs from the logs file for a given month.
* It does not clear them if the site is in the HOLDS file.
*

DO getmonth

```
SELECT site FROM U:\IMPROVE\QAPlots\Holds\holdsite WHERE qtr=mSeason AND  
mon=VAL(mMonth) INTO ARRAY holdsites  
LogFile = "m:\IMPROVE\LOGS\" + mSeason + "Logs.dbf"
```

```
IF SELECT("logs")>0  
  SELECT logs  
  USE  
ENDIF  
USE &LogFile ALIAS logs
```

```
SELECT logs  
SET FILTER TO ASCAN(holdsites, site)=0 AND MONTH(samdat)=VAL(mMonth)  
REPLACE ALL aflnm WITH "NM" FOR aflnm="QD"  
REPLACE ALL bflnm WITH "NM" FOR bflnm="QD"  
REPLACE ALL cflnm WITH "NM" FOR cflnm="QD"  
REPLACE ALL dflnm WITH "NM" FOR dflnm="QD"  
SET FILTER to
```

RETURN

D.36. OutCheck

```
* Program OutCheck
*
* This routine opens the output file as alias out using the getmonth screen as input
* It then runs through a series of checks on the file to be wure it's ready to deliver
* to CIRA.
```

```
PROCEDURE outcheck
```

```
PUBLIC mcalled
mcalled=.T.
```

```
DO WHILE .T.
```

```
DO getmonth
TotPath = "u:\improve\TotalDBF\"
```

```
IF SELECT("OUT")>0
    SELECT out
    USE
ENDIF
```

```
OutFile = TotPath + mSeason + "tot" + mMonth + ".dbf"
USE &OutFile IN 0 ALIAS out
```

```
SELECT out
SET FILTER TO MF>MT AND MT>-999
COUNT TO num
GO TOP
BROWSE TITLE STR(Num) + " Records with MF>MT" FIELDS site:10, samdat, ;
    Flow_A, Flow_D, Time_A, Time_D, Flag_A, Flag_D, ;
    Ratio=1.41*(MF-MT)/SQRT(MT_ERR^2+MF_ERR^2):H="Difference/Error":20, ;
    MF, MF_ERR, MF_MDL, MT, MT_ERR, MT_MDL
SET FILTER TO
```

```
SET FILTER TO Flag_A="RF" OR FLAG_B="RF" OR FLAG_C="RF" OR Flag_D="RF"
COUNT TO num
GO TOP
BROWSE TITLE STR(Num) + " Records with RF Flag"
SET FILTER TO
```

```
* SET FILTER TO Flag_A="LF" OR FLAG_B="LF" OR FLAG_C="LF" OR Flag_D="LF"
* BROWSE TITLE "Sites with LF Flag"
* SET FILTER TO
```

```
SET FILTER TO Flag_A="CG" OR FLAG_B="CG" OR FLAG_C="CG" OR Flag_D="CG"
COUNT TO num
GO TOP
BROWSE TITLE STR(Num) + " Records with CG Flag"
SET FILTER TO
```

```
SET FILTER TO Flag_A="CL" OR FLAG_B="CL" OR FLAG_C="CL" OR Flag_D="CL"
COUNT TO num
GO TOP
BROWSE TITLE STR(Num) + " Records with CL Flag"
SET FILTER TO
```

```
SET FILTER TO Flag_A="QD" OR FLAG_B="QD" OR FLAG_C="QD" OR Flag_D="QD"
COUNT TO num
GO TOP
BROWSE TITLE STR(Num) + " Records with QD Flag"
SET FILTER TO
```

```
SET FILTER TO Flag_A="SA" OR FLAG_B="SA" OR FLAG_C="SA" OR Flag_D="SA"
COUNT TO num
GO TOP
BROWSE TITLE STR(Num) + " Records with SA Flag"
SET FILTER TO
```

```
SET FILTER TO Flag_A="QA" OR FLAG_B="QA" OR FLAG_C="QA" OR Flag_D="QA"
```

```

COUNT TO num
GO TOP
BROWSE TITLE STR(Num) + " Records with QA Flag"
SET FILTER TO

USE u:\improve\qaplots\holds\holdsite.dbf IN 0 ALIAS holds
SELECT holds
SET FILTER TO holds.qtr=mSeason AND holds.mon=VAL(mMonth)
COUNT TO num
GO TOP
BROWSE TITLE STR(Num) + " Sites held for further review"
USE
SELECT out

* SET FILTER TO Flag_A="SW" OR FLAG_B="SW" OR FLAG_C="SW" OR Flag_D="SW"
* BROWSE TITLE "Sites with SW Flag"
* SET FILTER TO
*
* SET FILTER TO Flag_A="SP" OR FLAG_B="SP" OR FLAG_C="SP" OR Flag_D="SP"
* BROWSE TITLE "Sites with SP Flag"
* SET FILTER TO
*
* SET FILTER TO Flag_A="OL" OR FLAG_B="OL" OR FLAG_C="OL" OR Flag_D="OL"
* BROWSE TITLE "Sites with OL Flag"
* SET FILTER TO
*

SELECT out.site, COUNT(out.site);
    FROM out;
    GROUP BY out.site;
    HAVING COUNT(out.site) > 11;
    OR COUNT(out.site) < 10;
    ORDER BY out.site;
    INTO CURSOR Oddsites

SELECT Oddsites
COUNT TO num
GO TOP
BROWSE TITLE STR(Num) + " Records with less than 10 or more than 11 records"

*SELECT out.site, COUNT(out.site);
FROM out;
GROUP BY out.site;
HAVING COUNT(out.site) > 31;
OR COUNT(out.site) < 30;
ORDER BY out.site;
INTO CURSOR Oddsites

*SELECT Oddsites
*BROWSE TITLE "Sites with less than 30 or more than 31 records"

USE
SELECT out
USE
ENDDO
RETURN TO MASTER

```


APPENDIX E. NETWORK_METRIC_CHECKS CODE

Table of Contents

| | | |
|------|----------------------------|------|
| E.1. | NETWORK_METRIC_CHECKS..... | E-2 |
| E.2. | CARBON_METRIC_CHECKS..... | E-3 |
| E.3. | ELEMENT_METRIC_CHECKS..... | E-7 |
| E.4. | ION_METRIC_CHECKS..... | E-12 |
| E.5. | ERRHAND..... | E-16 |
| E.6. | NMC_OPENFILES..... | E-17 |
| E.7. | NMC_USERINPUT..... | E-19 |

E.1. NETWORK_METRIC_CHECKS

```

***** * M:\IMPROVE\NETWORK_METRICS\SOURCE_CODE\NETWORK_METRIC_CHECKS.PRG
*:*****
*:
*: Procedure File M:\IMPROVE\NETWORK_METRICS\NETWORK_METRIC_CHECKS.PRG
*:
*:
*:
*:
*:
*:
*:
*:
*:
*:
*:
*:
*:
*:
*: Documented using Visual FoxPro Formatting wizard version .05
*:*****
*: NETWORK_METRIC_CHECKS
* PROGRAM network_metric_checks
* This program calculates the network metrics for the month and year requested
* by the user and compares the metrics to tables of acceptance criteria. Any
* metrics that are outside the acceptance criteria are reported in the results.dbf
* table.
Public mOutput1, mOutput2

Set Default To m:\improve\programs\network_metrics\
Set Path To m:\improve\programs\flow_rate\
On Error Do errhand With ;
    ERROR( ), Message( ), Message(1), Program( ), Lineno( )

Do While .T.
    Wait Window At 45, 45 "Make sure the following files have been updated:" ;
        + Chr(13) + Chr(10) + "- Elements" ;
        + Chr(13) + Chr(10) + "- Ions" ;
        + Chr(13) + Chr(10) + "- Carbon" ;
        + Chr(13) + Chr(10) + "- Laser" ;
        NOWAIT Timeout(5) Noclear

    Do nmc_openfiles
    Do nmc_userinput

    * Allow chance to bail out of program
    Wait Window At 10, 40 "Press 'C' to continue" + Chr(13) + Chr(10) To mcont
    If Not Inlist (mcont, "c", "C")
        Cancel
    Endif

    *run the procedures to calculate and check the metrics
    Do carbon_metric_checks
    Do ion_metric_checks
    Do element_metric_checks

    Use &mOutput1 In 25
    Select 25
    * SET FILTER TO month = themonth and year = theyear
    Delete For Month = themonth And Year = theyear
    Pack
    Append From &c_out
    Append From &e_out
    Append From &i_out
    Browse

    Select network_metric_limits
    Use
    Select artifact_metric_limits
    Use

Enddo
Cancel

```

E.2. CARBON_METRIC_CHECKS

```
***** * M:\IMPROVE\NETWORK_METRICS\SOURCE_CODE\CARBON_METRIC_CHECKS.PRG
*: *****
*:
*: Procedure File M:\IMPROVE\NETWORK_METRICS\CARBON_METRIC_CHECKS.PRG
*:
*:
*:
*:
*:
*:
*:
*:
*:
*:
*:
*:
*:
*: Documented using Visual FoxPro Formatting wizard version .05
*: *****
*: carbon_metric_checks
*: carbon_metric_checks
*! *****
*!
*! Procedure CARBON_METRIC_CHECKS
*!
*! Calls
*! c_artifacts
*! c_metrics
*!
*! *****
Procedure carbon_metric_checks

Public martifact, mcount, mmin, mmax, mmedian, mavg, c_var, c_param, mmonthly, myear
Public mmyy_var, mOutPath, themonth, theyear, c_out, c_artout
char_year = Transform(theyear)
char_month = Transform(themonth)
*c_out = mOutPath + "c_metricchecks" + char_month + char_year + ".dbf"
c_out = mOutPath + "c_metricchecks.dbf"
c_artout = mOutPath + "c_artifactchecks.dbf"

* this query cleans up the carbon data by eliminating duplicate records
* and records that are not from sites in the site table
Select Distinct c_rawdata.*;
FROM c_rawdata INNER Join sites ;
ON c_rawdata.site = sites.site;
WHERE c_rawdata.qid Is Not Null And Year(c_rawdata.samdat) > 0 And;
right(sites.site,1) <> "9";
INTO Table c_data
*close the raw data file
Select c_rawdata
Use

* set indices on the cleaned data file
Select c_data
Index On Trim(site) + Dtoc(samdat) + strtim Tag sitdatim
Index On Octc Tag Octc
Index On Ectc Tag Ectc
Index On Tctc Tag Tctc
Index On O1tc Tag O1tc
Index On O2tc Tag O2tc
Index On O3tc Tag O3tc
Index On O4tc Tag O4tc
Index On Optc Tag Optc
Index On E1tc Tag E1tc
Index On E2tc Tag E2tc
Index On E3tc Tag E3tc

* create and fill an array for the OCEC parameter names
Public c_param(11)
c_param(1) = "octc"
c_param(2) = "ectc"
c_param(3) = "tctc"
c_param(4) = "o1tc"
c_param(5) = "o2tc"
c_param(6) = "o3tc"
```

```

c_param(7) = "o4tc"
c_param(8) = "optc"
c_param(9) = "eltc"
c_param(10) = "e2tc"
c_param(11) = "e3tc"

*delete the existing records in c_metrics for the months/years that will be
* processed by this run of the program
For Each mmyy_var In amonthyr
  Select monthyr
  Set Filter To monthyr.Number = mmyy_var
  Go Top
  mmonth = monthyr.Month
  myear = monthyr.Year
  Select c_artifacts
  Set Filter To Month = mmonth And Year = myear
  Delete All
  Select c_metrics
  Set Filter To Month = mmonth And Year = myear
  Delete All
Endfor
Select c_artifacts
Pack
Select c_metrics
Pack

* This FOR EACH...ENDFOR loop executes for each variable in the c_param matrix
For Each c_var In c_param
  * This FOR EACH...ENDFOR loop executes for the requested month/year and the
  * two prior month/years
  For Each mmyy_var In amonthyr
    * Display parameter being processed
    Wait Window At 10,50 "Processing &c_var artifact data" Nowait
    Select monthyr
    Set Filter To monthyr.Number = mmyy_var
    Go Top
    mmonth = monthyr.Month
    myear = monthyr.Year
    Select c_data
    Set Filter To &c_var > -99 And Month(samdat) = mmonth And;
    STATUS = "CS" And Year(samdat) = myear
    Set Order To c_var
    Count To mcount
    If mcount > 0
      Calculate Std(&c_var) To mstdev
      * following code finds the median value
      Go Top
      n50 = Round(mcount*0.5,0)
      Skip (n50-1)
      mmedian = &c_var
    Else
      mstdev = -999
      mmedian = -999
    Endif
    Insert Into c_artifacts (Parameter, Year, Month, Count, median, stdev) ;
    VALUE (c_var, myear, mmonth, mcount, mmedian, mstdev)
  Endfor
Endfor

* This FOR EACH...ENDFOR loop executes for each variable in the c_param matrix
For Each c_var In c_param
  * This FOR EACH...ENDFOR loop executes for the requested month/year and the
  * two prior month/years
  For Each mmyy_var In amonthyr
    * Display parameter being processed
    Wait Window At 10,50 "Processing &c_var data" Nowait
    Select monthyr
    Set Filter To monthyr.Number = mmyy_var
    Go Top
    mmonth = monthyr.Month
    myear = monthyr.Year
    Select c_data
    Set Filter To &c_var > -99 And Month(samdat) = mmonth And;
    STATUS = "CP" And Year(samdat) = myear
    Set Order To &c_var
    Count To mcount
    If mcount > 0
      Calculate Min(&c_var), Max(&c_var) To m_min, m_max
      Calculate Std(&c_var), Avg(&c_var) To m_stdev, m_avg
      * following code finds the 10th, 50th, and 90th percentiles
      Go Top

```

```

n10 = Round(mcount*0.1,0)
Skip (n10-1)
m_10percent = &c_var
Go Top
n50 = Round(mcount*0.5,0)
Skip (n50-1)
m_median = &c_var
Go Top
n90 = Round(mcount*0.9,0)
Skip (n90-1)
m_90percent = &c_var
Select c_artifacts
Set Filter To Parameter = c_var And Year = myear And Month = mmonth
Go Top
martifact = c_artifacts.median
If martifact > -99
    mmin = m_min - martifact
    mmax = m_max - martifact
    mstdev = m_stdev - martifact
    mavg = m_avg - martifact
    m10percent = m_10percent - martifact
    mmedian = m_median - martifact
    m90percent = m_90percent - martifact
Else
    mmin = m_min
    mmax = m_max
    mstdev = m_stdev
    mavg = m_avg
    m10percent = m_10percent
    mmedian = m_median
    m90percent = m_90percent
Endif
Else
    Select c_artifacts
    Set Filter To Parameter = c_var And Year = myear And Month = mmonth
    Go Top
    martifact = c_artifacts.median
    mmin = -999
    mmax = -999
    mstdev = -999
    mavg = -999
    m10percent = -999
    mmedian = -999
    m90percent = -999
Endif
Insert Into c_metrics (Parameter, Year, Month, Count, artifact, Min, ;
max, stdev, Avg, percent10, median, percent90) ;
VALUE (c_var, myear, mmonth, mcount, martifact, mmin, mmax, mstdev, mavg,
;
m10percent, mmedian, m90percent)
Endfor
Endfor

*close the carbon data file
Select c_data
Set Filter To
Use
Select c_artifacts
Set Filter To
Use
Select c_metrics
Set Order To paramyyymm

*extract only the current month from the metrics file
Select *;
FROM c_metrics;
WHERE c_metrics.Month = themonth And c_metrics.Year = theyear;
INTO Table temp.Dbf

*compare the current month metrics to the metric limits determined from
* previous months
Select network_metric_limits.Parameter, network_metric_limits.Month,;
temp.Year, temp.median, network_metric_limits.l_low50,;
network_metric_limits.l_high50, temp.percent90,;
network_metric_limits.l_low90, network_metric_limits.l_high90,;
temp.Max, network_metric_limits.l_max;
FROM network_metric_limits INNER Join temp ;
ON network_metric_limits.Parameter = temp.Parameter;
WHERE (network_metric_limits.Month = temp.Month) And ;
(temp.Max > network_metric_limits.l_max Or ;
temp.median < network_metric_limits.l_low50 Or ;

```

```

        (temp.median > network_metric_limits.l_high50 And ;
network_metric_limits.l_high50 > 0.0) Or ;
temp.percent90 < network_metric_limits.l_low90 Or ;
(temp.percent90 > network_metric_limits.l_high90 And ;
network_metric_limits.l_high90 > 0.0));
INTO Table &c_out

Select temp
Use

*extract only the current month from the metrics file
Select *;
FROM c_artifacts;
WHERE c_artifacts.Month = themonth And c_artifacts.Year = theyear;
INTO Table temp.Dbf

*compare the current month artifacts values to the artifact limits determined from
* previous months
Select artifact_metric_limits.Parameter, artifact_metric_limits.Month,;
temp.Year, temp.Count, temp.median, artifact_metric_limits.l_low50,;
artifact_metric_limits.l_high50, temp.stdev,;
artifact_metric_limits.l_stdev;
FROM artifact_metric_limits INNER Join temp ;
ON artifact_metric_limits.Parameter = temp.Parameter;
WHERE ((artifact_metric_limits.Month = temp.Month) And ;
(temp.stdev > artifact_metric_limits.l_stdev Or ;
temp.median < artifact_metric_limits.l_low50 Or ;
temp.median > artifact_metric_limits.l_high50)) Or;
(temp.Count < artifact_metric_limits.Count);
INTO Table &c_artout

Select temp
Use

```



```

Index On SR Tag SR
Index On ZR Tag ZR
Index On PNA Tag PNA
Index On PMG Tag PMG
Index On PAL Tag PAL
Index On PSI Tag PSI
Index On PS Tag PS
Index On PCL Tag PCL
Index On PK Tag PK
Index On PCA Tag PCA
Index On PTI Tag PTI
Index On Pv Tag Pv
Index On PCR Tag PCR
Index On PMN Tag PMN
Index On PFE Tag PFE

```

```

* save the earliest year and latest year into variables
Calculate Min(Year(samdat)) To min_yr
Calculate Max(Year(samdat)) To max_yr

```

```

* create and fill matrices for the parameter names. one matrix is created
* for each type of analysis b/c each analysis has a separate status flag
* and later the statistics queries use the status flags

```

```

Public CuXRF_param(14)
CuXRF_param(1) = "YNa"
CuXRF_param(2) = "YMg"
CuXRF_param(3) = "YAl"
CuXRF_param(4) = "YSi"
CuXRF_param(5) = "YP"
CuXRF_param(6) = "YS"
CuXRF_param(7) = "YCl"
CuXRF_param(8) = "YK"
CuXRF_param(9) = "YCa"
CuXRF_param(10) = "YTi"
CuXRF_param(11) = "YV"
CuXRF_param(12) = "YCr"
CuXRF_param(13) = "YMn"
CuXRF_param(14) = "YFe"
Public MoXRF_param(10)
MoXRF_param(1) = "Ni"
MoXRF_param(2) = "Cu"
MoXRF_param(3) = "Zn"
MoXRF_param(4) = "As"
MoXRF_param(5) = "Pb"
MoXRF_param(6) = "Se"
MoXRF_param(7) = "Br"
MoXRF_param(8) = "Rb"
MoXRF_param(9) = "Sr"
MoXRF_param(10) = "Zr"
Public PIXE_param(13)
PIXE_param(1) = "PNa"
PIXE_param(2) = "PMg"
PIXE_param(3) = "PAL"
PIXE_param(4) = "PSi"
PIXE_param(5) = "PS"
PIXE_param(6) = "PCL"
PIXE_param(7) = "PK"
PIXE_param(8) = "PCa"
PIXE_param(9) = "PTi"
PIXE_param(10) = "PV"
PIXE_param(11) = "PCr"
PIXE_param(12) = "PMn"
PIXE_param(13) = "PFe"

```

```

*delete the existing records in e_metrics for the months/years that will be
* processed by this run of the program

```

```

For Each mmyy_var In amonthyr
  Select monthyr
  Set Filter To monthyr.Number = mmyy_var
  Go Top
  mmonth = monthyr.Month
  myear = monthyr.Year
  Select e_metrics
  Set Filter To Month = mmonth And Year = myear
  Delete All
Endfor
Select e_metrics
Pack

```

```

* This set of loops executes on the H data, which is the only PESA parameter
Mvar = "H"

```



```

For Each mmyy_var In amonthyr
  * Display parameter being processed
  Wait Window At 10,50 "Processing H data" Nowait
  Select monthyr
  Set Filter To monthyr.Number = mmyy_var
  Go Top
  mmonth = monthyr.Month
  myear = monthyr.Year
  Select e_data
  Set Filter To H > -99 And Month(samdat) = mmonth And;
  PESSTAT = "NM" And Year(samdat) = myear
  Set Order To H
  Count To mcount
  If mcount > 0
    Calculate Min(H), Max(H) To mmin, mmax
    Calculate Std(H), Avg(H) To mstd, mavg
    * following code finds the 10th, 50th, and 90th percentiles
    Go Top
    n10 = Round(mcount*0.1,0)
    Skip (n10-1)
    m10percent = H
    Go Top
    n50 = Round(mcount*0.5,0)
    Skip (n50-1)
    mmedian = H
    Go Top
    n90 = Round(mcount*0.9,0)
    Skip (n90-1)
    m90percent = H
  Else
    mmin = -999
    mmax = -999
    mstd = -999
    mavg = -999
    m10percent = -999
    mmedian = -999
    m90percent = -999
  Endif
  Insert Into e_metrics (Parameter, Year, Month, Count, Min, Max, Std,;
  avg, percent10, median, percent90) ;
  VALUE (Mvar, myear, mmonth, mcount, mmin, mmax, mstd, mavg, m10percent,;
  mmedian, m90percent)
Endfor

* This FOR...ENDFOR loop executes for each of the Copper XRF parameters
For Each e_var In CuXRF_param
  * This FOR EACH...ENDFOR loop executes for the requested month/year and the
  * two prior month/years
  For Each mmyy_var In amonthyr
    * Display parameter being processed
    Wait Window At 10,50 "Processing &e_var data" Nowait
    Select monthyr
    Set Filter To monthyr.Number = mmyy_var
    Go Top
    mmonth = monthyr.Month
    myear = monthyr.Year
    Select e_data
    Set Filter To &e_var > -99 And Month(samdat) = mmonth And;
    Xrcstat = "NM" And Year(samdat) = myear
    Set Order To e_var
    Count To mcount
    If mcount > 0
      Calculate Min(&e_var), Max(&e_var) To mmin, mmax
      Calculate Std(&e_var), Avg(&e_var) To mstd, mavg
      * following code finds the 10th, 50th, and 90th percentiles
      Go Top
      n10 = Round(mcount*0.1,0)
      Skip (n10-1)
      m10percent = &e_var
      Go Top
      n50 = Round(mcount*0.5,0)
      Skip (n50-1)
      mmedian = &e_var
      Go Top
      n90 = Round(mcount*0.9,0)
      Skip (n90-1)
      m90percent = &e_var
      Insert Into e_metrics (Parameter, Year, Month, Count, Min, Max, Std,;
      avg, percent10, median, percent90) ;
      VALUE (e_var, myear, mmonth, mcount, mmin, mmax, mstd, mavg,
      m10percent,;

```

```

                                mmedian, m90percent)
        Endif
    Endfor
Endfor

* This FOR...ENDFOR loop executes for every Molybdenum XRF parameter
For Each e_var In MoXRF_param
    * This FOR EACH...ENDFOR loop executes for the requested month/year and the
    * two prior month/years
    For Each mmyy_var In amonthyr
        * Display parameter being processed
        Wait Window At 10,50 "Processing &e_var data" Nowait
        Select monthyr
        Set Filter To monthyr.Number = mmyy_var
        Go Top
        mmonth = monthyr.Month
        myear = monthyr.Year
        Select e_data
        Set Filter To &e_var > -99 And Month(samdat) = mmonth And;
        Xrmstat = "NM" And Year(samdat) = myear
        Set Order To e_var
        Count To mcount
        If mcount > 0
            Calculate Min(&e_var), Max(&e_var) To mmin, mmax
            Calculate Std(&e_var), Avg(&e_var) To mstd, mavg
            * following code finds the 10th, 50th, and 90th percentiles
            Go Top
            n10 = Round(mcount*0.1,0)
            Skip (n10-1)
            m10percent = &e_var
            Go Top
            n50 = Round(mcount*0.5,0)
            Skip (n50-1)
            mmedian = &e_var
            Go Top
            n90 = Round(mcount*0.9,0)
            Skip (n90-1)
            m90percent = &e_var
        Else
            mmin = -999
            mmax = -999
            mstd = -999
            mavg = -999
            m10percent = -999
            mmedian = -999
            m90percent = -999
        Endif
        Insert Into e_metrics (Parameter, Year, Month, Count, Min, Max, Std,;
        avg, percent10, median, percent90) ;
        VALUE (e_var, myear, mmonth, mcount, mmin, mmax, mstd, mavg, m10percent,;
        mmedian, m90percent)
    Endfor
Endfor

If (theyear < 2001) Or (themoth < 12 And theyear = 2001)
    * This FOR...ENDFOR loop executes for each PIXE parameter
    For Each e_var In PIXE_param
        * This FOR EACH...ENDFOR loop executes for the requested month/year and the
        * two prior month/years
        For Each mmyy_var In amonthyr
            * Display parameter being processed
            Wait Window At 10,50 "Processing &e_var data" Nowait
            Select monthyr
            Set Filter To monthyr.Number = mmyy_var
            Go Top
            mmonth = monthyr.Month
            myear = monthyr.Year
            Select e_data
            Set Filter To &e_var > -99 And Month(samdat) = mmonth And;
            Pixstat = "NM" And Year(samdat) = myear
            Set Order To e_var
            Count To mcount
            If mcount > 0
                Calculate Min(&e_var), Max(&e_var) To mmin, mmax
                Calculate Std(&e_var), Avg(&e_var) To mstd, mavg
                * following code finds the 10th, 50th, and 90th percentiles
                Go Top
                n10 = Round(mcount*0.1,0)
                Skip (n10-1)
                m10percent = &e_var
                Go Top

```

```

        n50 = Round(mcount*0.5,0)
        Skip (n50-1)
        mmedian = &e_var
        Go Top
        n90 = Round(mcount*0.9,0)
        Skip (n90-1)
        m90percent = &e_var
    Else
        mmin = -999
        mmax = -999
        mstd = -999
        mavg = -999
        m10percent = -999
        mmedian = -999
        m90percent = -999
    Endif
    Insert Into e_metrics (Parameter, Year, Month, Count, Min, Max, Std,
    avg, percent10, median, percent90) ;
    VALUE (e_var, myear, mmonth, mcount, mmin, mmax, mstd, mavg,
m10percent, ;
        mmedian, m90percent)
    Endfor
Endif

* close the PIXE/XRF/PESA data file
Select e_data
Set Filter To
Use
Select e_metrics
Set Order To paramyymm

*extract only the current month from the metrics file
Select *;
    FROM e_metrics;
    WHERE e_metrics.Month = themonth And e_metrics.Year = theyear;
    INTO Table temp.Dbf

*compare the current month metrics to the metric limits determined from
* previous months
Select network_metric_limits.Parameter, network_metric_limits.Month,;
temp.Year, temp.median, network_metric_limits.l_low50,;
network_metric_limits.l_high50, temp.percent90,;
network_metric_limits.l_low90, network_metric_limits.l_high90,;
temp.Max, network_metric_limits.l_max;
FROM network_metric_limits INNER Join temp ;
ON network_metric_limits.Parameter = temp.Parameter;
WHERE (network_metric_limits.Month = temp.Month) And ;
(temp.Max > network_metric_limits.l_max Or ;
temp.median < network_metric_limits.l_low50 Or ;
(temp.median > network_metric_limits.l_high50 And ;
network_metric_limits.l_high50 > 0.0) Or ;
temp.percent90 < network_metric_limits.l_low90 Or ;
(temp.percent90 > network_metric_limits.l_high90 And ;
network_metric_limits.l_high90 > 0.0));
INTO Table &e_out

Select temp
Use

```

E.4. ION_METRIC_CHECKS

```
***** * M:\IMPROVE\NETWORK_METRICS\SOURCE_CODE\ION_METRIC_CHECKS.PRG
*:*****
*:
*: Procedure File M:\IMPROVE\NETWORK_METRICS\ION_METRIC_CHECKS.PRG
*:
*:
*:
*:
*:
*:
*:
*:
*:
*:
*:
*:
*:
*:
*:
*:
*:
*:
*:
*:
*:
*: Documented using Visual FoxPro Formatting wizard version .05
*:*****
*: ion_metric_checks
*: ion_metric_checks
*|*****
*!
*! Procedure ION_METRIC_CHECKS
*!
*! Calls
*!     i_artifacts
*!     i_metrics
*!
*|*****
Procedure ion_metric_checks

Public martifact, mcount, mmin, mmax, mmedian, mavg, i_var, i_param, mmonth, myear
Public mmyy_var, mOutPath, themonth, theyear, i_out, i_artout
char_year = Transform(theyear)
char_month = Transform(themonth)
*i_out = mOutPath + "i_metricchecks" + char_month + char_year + ".dbf"
i_out = mOutPath + "i_metricchecks.dbf"
i_artout = mOutPath + "i_artifactchecks.dbf"

* this query cleans up the ion data by eliminating duplicate records
* and records that are not from sites in the site table
Select Distinct i_rawdata.*;
FROM i_rawdata INNER Join sites ;
ON i_rawdata.site = sites.site;
WHERE Year(i_rawdata.samdat) > 0 And Right(sites.site,1) <> "9";
INTO Table i_data
*close the raw data file
Select i_rawdata
Use

* open the cleaned data file
Select i_data
* replace the 0.000 nh4 values with missing values (Lowell's program
* incorrectly replaced null values in the RTI files with 0.000 values
* for several months)
Replace nh4 With -999 For nh4 = 0.000
*set indices
Index On Trim(site) + Dtoc(samdat) + strtim Tag sitdatim
Index On cl Tag cl
Index On no2 Tag no2
Index On no3 Tag no3
Index On so4 Tag so4
Index On nh4 Tag nh4

* create and fill a matrix for the OCEC parameter names
Public i_param(5)
i_param(1) = "cl"
i_param(2) = "no2"
i_param(3) = "no3"
i_param(4) = "so4"
i_param(5) = "nh4"

*delete the existing records in i_metrics for the months/years that will be
* processed by this run of the program
```

```

For Each mmyy_var In amonthyr
  Select monthyr
  Set Filter To monthyr.Number = mmyy_var
  Go Top
  mmonth = monthyr.Month
  myear = monthyr.Year
  Select i_artifacts
  Set Filter To Month = mmonth And Year = myear
  Delete All
  Select i_metrics
  Set Filter To Month = mmonth And Year = myear
  Delete All
Endfor
Select i_artifacts
Pack
Select i_metrics
Pack

* This FOR EACH...ENDFOR loop executes for each variable in the i_param matrix
For Each i_var In i_param
  * This FOR EACH...ENDFOR loop executes for the requested month/year and the
  * two prior month/years
  For Each mmyy_var In amonthyr
    * Display parameter being processed
    Wait Window At 10,50 "Processing &i_var artifact data" Nowait
    Select monthyr
    Set Filter To monthyr.Number = mmyy_var
    Go Top
    mmonth = monthyr.Month
    myear = monthyr.Year
    Select i_data
    Set Filter To &i_var > -99 And Month(samdat) = mmonth And;
      STATUS = "FB" And Year(samdat) = myear
    Set Order To i_var
    Count To mcount
    If mcount > 0
      Calculate Std(&i_var) To mstdev
      * following code finds the median value
      Go Top
      n50 = Round(mcount*0.5,0)
      Skip (n50-1)
      mmedian = &i_var
    Else
      mstdev = -999
      mmedian = -999
    Endif
    Insert Into i_artifacts (Parameter, Year, Month, Count, median, stdev) ;
      VALUE (i_var, myear, mmonth, mcount, mmedian, mstdev)
  Endfor
Endfor

* This FOR EACH...ENDFOR loop executes for each variable in the i_param matrix
For Each i_var In i_param
  * This FOR EACH...ENDFOR loop executes for the requested month/year and the
  * two prior month/years
  For Each mmyy_var In amonthyr
    * Display parameter being processed
    Wait Window At 10,50 "Processing &i_var data" Nowait
    Select monthyr
    Set Filter To monthyr.Number = mmyy_var
    Go Top
    mmonth = monthyr.Month
    myear = monthyr.Year
    Select i_data
    Set Filter To &i_var > -99 And Month(samdat) = mmonth And;
      STATUS = "NM" And Year(samdat) = myear
    Set Order To i_var
    Count To mcount
    If mcount > 0
      Calculate Min(&i_var), Max(&i_var) To m_min, m_max
      Calculate Std(&i_var), Avg(&i_var) To m_stdev, m_avg
      * following code finds the 10th, 50th, and 90th percentiles
      Go Top
      n10 = Round(mcount*0.1,0)
      Skip (n10-1)
      m_10percent = &i_var
      Go Top
      n50 = Round(mcount*0.5,0)
      Skip (n50-1)
      m_median = &i_var
      Go Top
    Endif
  Endfor
Endfor

```

```

n90 = Round(mcount*0.9,0)
Skip (n90-1)
m_90percent = &i_var
Select i_artifacts
Set Filter To Parameter = i_var And Year = myear And Month = mmonth
Go Top
martifact = i_artifacts.median
If martifact > -99
    mmin = m_min - martifact
    mmax = m_max - martifact
    mstdev = m_stdev - martifact
    mavg = m_avg - martifact
    m10percent = m_10percent - martifact
    mmedian = m_median - martifact
    m90percent = m_90percent - martifact
Else
    mmin = m_min
    mmax = m_max
    mstdev = m_stdev
    mavg = m_avg
    m10percent = m_10percent
    mmedian = m_median
    m90percent = m_90percent
Endif
Else
    Select i_artifacts
    Set Filter To Parameter = i_var And Year = myear And Month = mmonth
    Go Top
    martifact = median
    mmin = -999
    mmax = -999
    mstdev = -999
    mavg = -999
    m10percent = -999
    mmedian = -999
    m90percent = -999
Endif
Insert Into i_metrics (Parameter, Year, Month, Count, artifact, Min, ;
max, stdev, Avg, percent10, median, percent90);
VALUE (i_var, myear, mmonth, mcount, martifact, mmin, mmax, mstdev, mavg,
;
m10percent, mmedian, m90percent)
Endfor
Endfor

Select i_data
Set Filter To
Use
Select i_artifacts
Set Filter To
Use
Select i_metrics
Set Order To paramyyymm

*extract only the current month from the metrics file
Select *;
FROM i_metrics;
WHERE i_metrics.Month = themonth And i_metrics.Year = theyear;
INTO Table temp.Dbf

*compare the current month metrics to the metric limits determined from
* previous months
Select network_metric_limits.Parameter, network_metric_limits.Month,;
temp.Year, temp.median, network_metric_limits.l_low50,;
network_metric_limits.l_high50, temp.percent90,;
network_metric_limits.l_low90, network_metric_limits.l_high90,;
temp.Max, network_metric_limits.l_max;
FROM network_metric_limits INNER Join temp ;
ON network_metric_limits.Parameter = temp.Parameter;
WHERE (network_metric_limits.Month = temp.Month) And ;
(temp.Max > network_metric_limits.l_max Or ;
temp.median < network_metric_limits.l_low50 Or ;
(temp.median > network_metric_limits.l_high50 And ;
network_metric_limits.l_high50 > 0.0) Or ;
temp.percent90 < network_metric_limits.l_low90 Or ;
(temp.percent90 > network_metric_limits.l_high90 And ;
network_metric_limits.l_high90 > 0.0));
INTO Table &i_out

Select temp
Use

```

```

Select *;
  FROM i_artifacts;
  WHERE i_artifacts.Month = themonth And i_artifacts.Year = theyear;
  INTO Table temp.Dbf

*compare the current month artifacts values to the artifact limits determined from
* previous months
Select artifact_metric_limits.Parameter, temp.Month, temp.Year,;
  temp.Count, temp.median, artifact_metric_limits.l_low50,;
  artifact_metric_limits.l_high50, temp.stdev,;
  artifact_metric_limits.l_stdev;
FROM artifact_metric_limits INNER Join temp ;
ON artifact_metric_limits.Parameter = temp.Parameter;
WHERE (temp.stdev > artifact_metric_limits.l_stdev) Or ;
(temp.median < artifact_metric_limits.l_low50) Or ;
(temp.median > artifact_metric_limits.l_high50) Or;
(temp.Count < artifact_metric_limits.Count);
INTO Table &i_artout

Select temp
Use

```

E.5. ERRHAND

```
***** * M:\IMPROVE\PROGRAMS\ERRHAND.PRG
*.*****
*:
*: Procedure File M:\IMPROVE\NETWORK_METRICS\ERRHAND.PRG
*:
*:
*:
*:
*:
*:
*:
*:
*:
*:
*:
*:
*:
*:
*:
*:
*:
*:
*:
*:
*:
*: Documented using Visual FoxPro Formatting wizard version .05
*.*****
*:   errhand
*:   errhand
*|*****
*!
*! Procedure ERRHAND
*!
*|*****
Procedure errhand
    Parameter merror, Mess, mess1, mprog, mlineno
    Clear
    Set Alternate To m:\improve\temp\errorlog.txt Additive
    Set Alternate On
    ? Datetime()
    ? 'Error number: ' + Ltrim(Str(merror))
    ? 'Error message: ' + Mess
    ? 'Line of code with error: ' + mess1
    ? 'Line number of error: ' + Ltrim(Str(mlineno))
    ? 'Program with error: ' + mprog
    Set Alternate Off
    Set Alternate To
    *CLOSE DATABASES ALL
    Wait
    On Error  && restore system error handler
    Cancel
```


E.6. NMC_OPENFILES

```

***** * M:\IMPROVE\NETWORK_METRICS\SOURCE_CODE\NMC_OPENFILES.PRG
*: *****
*:
*: Procedure File M:\IMPROVE\NETWORK_METRICS\NMC_OPENFILES.PRG
*:
*:
*:
*:
*:
*:
*:
*:
*:
*:
*:
*:
*:
*:
*:
*:
*:
*:
*:
*:
*: Documented using Visual FoxPro Formatting wizard version .05
*: *****
*:   nmc_openfiles
*:   nmc_openfiles
*! *****
*!
*! Procedure NMC_OPENFILES
*!
*! *****
Procedure nmc_openfiles
* Written Nov 2004 by Nicole to calculate network-wide metrics on
* raw ions, carbon, and PESA/XRF/HIPS data for data validation

Public mmonth, myear, mOutPath, mOutput1, mOutput2

Set Talk Off
Close Databases All

* set paths
mCalPath = "U:\improve\sitecal\"
mSiteFile = mCalPath + "sitefile.dbf"
mOutPath = "m:\improve\programs\network_metrics\output\"
*e_out = mOutPath + "e_outoflimits" + themonth + theyear + ".dbf"
*i_out = mOutPath + "i_outoflimits" + themonth + theyear + ".dbf"
mOutput1 = mOutPath + "metric_checks.dbf"
mOutput2 = mOutPath + "artifact_checks.dbf"

If File('c_data.dbf')
  Delete File c_data.Dbf
Endif
If File('i_data.dbf')
  Delete File i_data.Dbf
Endif
If File('e_data.dbf')
  Delete File e_data.Dbf
Endif
If File('tsites.dbf')
  Delete File tsites.Dbf
Endif

*open the current site file, copy it, and close it
Use &mSiteFile In 0 Alias sitefile
Select sitefile
Copy All To tsites
Use
Use tsites In 0 Alias sites
Index On site Tag site
Set Order To site
Go Top
Use monthyr.Dbf In 0

*open the data files
Use Carbon_a97onward.Dbf In 0 Alias c_rawdata
Use ions_a95onward.Dbf In 0 Alias i_rawdata
Use Elements_a95onward.Dbf In 0 Alias e_rawdata

Use c_metrics.Dbf In 0
Select c_metrics

```

```
Index On (c_metrics.Parameter + Str(c_metrics.Year,4,0) + Str(c_metrics.Month,2,0)) Tag
paramyymm Candidate
Use c_artifacts.Dbf In 0
Select c_artifacts
Index On (c_artifacts.Parameter + Str(c_artifacts.Year,4,0) + Str(c_artifacts.Month,2,0)) Tag
paramyymm Candidate
Use i_metrics.Dbf In 0
Select i_metrics
Index On (Parameter + Str(Year,4,0) + Str(Month,2,0)) Tag paramyymm Candidate
Use i_artifacts.Dbf In 0
Select i_artifacts
Index On (Parameter + Str(Year,4,0) + Str(Month,2,0)) Tag paramyymm Candidate
Use e_metrics.Dbf In 0
Select e_metrics
Index On (Parameter + Str(Year,4,0) + Str(Month,2,0)) Tag paramyymm Candidate
```

E.7. NMC_USERINPUT

```
***** * M:\IMPROVE\NETWORK_METRICS\SOURCE_CODE\NMC_USERINPUT.PRG
*: *****
*:
*: Procedure File M:\IMPROVE\NETWORK_METRICS\NMC_USERINPUT.PRG
*:
*:
*:
*:
*:
*:
*:
*:
*:
*:
*:
*:
*:
*:
*:
*:
*:
*:
*:
*:
*:
*:
*: Documented using Visual FoxPro Formatting wizard version .05
*: *****
*: nmc_userinput
*: nmc_userinput
*| *****
*!
*! Procedure NMC_USERINPUT
*!
*! Calls
*!     aYear
*!     amonthyr
*!
*| *****
Procedure nmc_userinput
* This procedure displays a window to select a month and year.
* Parameters returned:
*   theMonth:    Integer (e.g., "12" for December)
*   theYear:     Integer

Clear
Public themonth, theyear, themonthyr
Public Array amonthyr(3)
Dimension aMonth(12), aYear(20)
themonth = 0
theyear = 0
themonthyr = 0

For mcount = 1 To 12  && Fill the month array
    aMonth(mcount) = mcount
Next
For mcount = 1 To 20  && Fill the year array
    aYear(mcount) = 1987 + mcount
Next

frmMyForm = Createobject('Form')  && Create a Form
frmMyForm.Closable = .F.  && Disable the Control menu box

frmMyForm.Move(150,10,500,400)  && Move the form
frmMyForm.Caption = 'Select Month & Year for Network Metrics Check'  && Caption on the form

frmMyForm.AddObject('ListBox1','1stMyListBox')  && Add ListBox control
frmMyForm.AddObject('ListBox2','2ndMyListBox')  && Add ListBox control
frmMyForm.AddObject('cmbCommand1','cmdMyCmdBtn')  && Add "Continue" Command button
frmMyForm.AddObject('cmbCommand2','QuitBtn')  && Add "Quit" Command button

frmMyForm.AddObject('Label1','Textbox')  && Add label to listbox
frmMyForm.Label1.Visible = .T.  && Make textbox visible
frmMyForm.Label1.Value = 'Select Month'
frmMyForm.Label1.Left = 10  && Specify textbox row
frmMyForm.Label1.Top = 10  && Specify textbox column

frmMyForm.AddObject('Label2','Textbox')  && Add label to listbox
frmMyForm.Label2.Visible = .T.  && Make textbox visible
frmMyForm.Label2.Value = 'Select Year'
frmMyForm.Label2.Left = 140  && Specify textbox row
frmMyForm.Label2.Top = 10  && Specify textbox column

frmMyForm.ListBox1.RowSourceType = 5  && Specifies an array
```

```

frmMyForm.ListBox1.RowSource = 'aMonth' && Array containing listbox items
frmMyForm.ListBox1.ControlSource = 'theMonth'
frmMyForm.ListBox2.RowSourceType = 5 && Specifies an array
frmMyForm.ListBox2.RowSource = 'aYear' && Array containing listbox items
frmMyForm.ListBox2.ControlSource = 'theYear'

frmMyForm.cmbCommand2.Visible = .T. && "Quit" Command button visible
frmMyForm.ListBox1.Visible = .T. && "List Box visible"

frmMyForm.Show && Display the form
frmMyForm.ListBox2.Visible = .F. && "List Box not visible"
Read Events && Start event processing

Select monthyr
Locate For Month = themonth And Year = theyear
*GO TOP
themonthyr = monthyr.Number
amonthyr(1) = themonthyr
amonthyr(2) = (themonthyr - 1)
amonthyr(3) = (themonthyr - 2)

*.*****
*:
*: Class:cmdMyCmdBtn BaseClass: CommandButton
*:
*.*****
Define Class cmdMyCmdBtn As CommandButton && Create Command button
Caption = '\<Continue' && Caption on the Command button
Cancel = .T. && Default Cancel Command button (Esc)
Left = 10 && Command button column
Top = 310 && Command button row
Height = 25 && Command button height

Procedure Click
If themonth = 0 Or theyear = 0
Clear
? 'Select MONTH and YEAR'
Wait
Clear
frmMyForm.cmbCommand1.Visible = .T. && "Continue" Command button visible
frmMyForm.cmbCommand2.Visible = .T. && "Quit" Command button visible
frmMyForm.ListBox1.Visible = .T. && "List Box visible"
frmMyForm.ListBox2.Visible = .F. && "List Box visible"
frmMyForm.Show
Else
Clear Events && Stop event processing, close Form
Release aMonth, aYear
theyear = theyear + 1987
Endif
Wait Clear
Enddefine

*.*****
*:
*: Class:QuitBtn BaseClass: CommandButton
*:
*.*****
Define Class QuitBtn As CommandButton && Create Command button
Caption = '\<Quit' && Caption on the Command button
Cancel = .T. && Default Cancel Command button (Esc)
Left = 10 && Command button column
Top = 260 && Command button row
Height = 25 && Command button height

Procedure Click
Wait Clear
Select c_rawdata
Use
Select i_rawdata
Use
Select e_rawdata
Use
Cancel
Enddefine

*.*****
*:
*: Class:lstMyListBox BaseClass: ListBox
*:
*.*****
Define Class lstMyListBox As ListBox && Create ListBox control

```

```

*   Caption = 'Select Month'  && Caption on the 1st listbox
Left = 10  && List Box column
Top = 30  && List Box row
MultiSelect = .F.  && Don't allow selecting more than 1 month
Height = 210

Procedure Click
  Activate Screen
  Clear
  frmMyForm.ListBox2.Visible = .T.  && "List Box visible
Enddefine

Define Class 2ndMyListBox As ListBox  && Create ListBox control
  Left = 140  && List Box column
  Top = 30  && List Box row
  MultiSelect = .F.  && Don't allow selecting more than 1 year
  Height = 355

  Procedure Click
    Activate Screen
    Clear
    frmMyForm.cmbCommand1.Visible = .T.  && "Continue" Command button visible
  Enddefine

```

APPENDIX F. MEASUREMENT OF IMPROVE FLOW RATE

Table of Contents

| | | |
|------|---|-----|
| F.1. | Introduction..... | F-2 |
| F.2. | Flow through an Orifice..... | F-2 |
| F.3. | Flow Rate Equation for the Cyclone Transducer..... | F-3 |
| F.4. | Flow Rate Equation for the Critical Orifice Transducer..... | F-3 |
| F.5. | Flow rate Equation for the Audit Devices | F-4 |
| F.6. | References..... | F-5 |

F.1. Introduction

The IMPROVE sampler measures pressure drops to determine flow rate. Both pressure transducers and Magnehelic® gauges are used to measure pressure drops across flow restrictions. The pressure drop is proportional to the mass flow rate. The three instances of this measurement technique in the IMPROVE sampler are

1. A transducer monitors the pressure drop across the cyclone. The size-selective PM_{2.5} cyclone serves as a *de facto* orifice. This is the primary flow measurement for the PM_{2.5} modules.
2. A needle valve serves as a critical orifice just upstream of the pump on all the PM_{2.5} and PM₁₀ modules. The valve is adjusted just prior to each calibration to achieve the desired flow rate. A transducer monitors the differential between atmospheric pressure and the pressure just upstream of the critical orifice. This is the primary flow measurement for the PM₁₀ modules and a backup flow measurement for the PM_{2.5} modules.
3. An audit device is used to calibrate the sampler flow rate. The pressure drop across this orifice is measured using a Magnehelic® gauge associated with the audit device.

F.2. Flow through an Orifice

The mass flow rate through an orifice depends on the square root of the density of the air and the square root of the pressure drop across the orifice. The basic equation for mass flow through an orifice is (Baker, 1989)

$$Q_m = K\sqrt{\rho}\sqrt{\delta P} \quad \text{F-1}$$

where Q_m is the mass flow rate, K is a constant representing the physical dimensions of the orifice, ρ is the density of the air just upstream of the orifice, and δP is the pressure drop across the orifice. IMPROVE defines standard reference conditions as pressure at sea level (P_0) and temperature at 20°C or 293 K (T_0). At these standard conditions the air density can be incorporated into the constant, K , to define a new constant, Q_1 :

$$Q_{m,STD} = Q_1\sqrt{\delta P} \quad \text{F-2}$$

where $Q_{m,STD}$ is the mass flow rate at sea level and 293 K. At any other pressure (P) and temperature (T , expressed here in °C), the new mass flow rate due to the change in air density can be expressed in terms of the square roots of T and P relative to standard conditions as

$$Q_m = Q_1\sqrt{\delta P} \sqrt{\frac{P}{P_0}} \sqrt{\frac{T_0}{T + 273.15}} \quad \text{F-3}$$

The volumetric flow rate, Q , can be calculated from the mass flow rate by applying the temperature (T) and pressure (P) under the conditions of the measurement:

$$Q = Q_m \left(\frac{P_0}{P} \right) \left(\frac{T + 273.15}{T_0} \right) = Q_1\sqrt{\delta P} \sqrt{\frac{P_0}{P}} \sqrt{\frac{T + 273.15}{T_0}} \quad \text{F-4}$$

We will write the pressure and temperature functions as F(elev) and f(T):

$$F(elev) = \sqrt{\frac{P_o}{P}} \quad \text{F-5}$$

Because we do not measure atmospheric pressure at IMPROVE sites, we have chosen to use an average pressure based on the elevation of each site as an approximation for the site pressure. Based on the 1954 tables of Trewartha, the elevation (i.e., pressure) factor at an elevation Z in meters is

$$F(elev) = \sqrt{\frac{P_o}{P}} = \exp\left[\frac{Z}{16874} + \left(\frac{Z}{37648}\right)^2\right] \quad \text{F-6}$$

An elevation factor for each site is tabulated in the IMPROVE database. The same fixed elevation factor is used for all calculations at a given site.

F.3. Flow Rate Equation for the Cyclone Transducer

The primary flow rate measurement for the PM_{2.5} module is made with a transducer measuring the pressure drop across the cyclone, which acts as a *de facto* orifice. The volumetric flow rate, Q, is calculated according to equation F-7, which is an analogue of equation F-4:

$$Q = 10^a M^b \sqrt{\frac{P_o}{P}} \sqrt{\frac{T + 273.15}{293.15}} = 10^a (M)^b * F(elev) * f(T) \quad \text{F-7}$$

where M is the transducer reading of the pressure drop. The term 10^a is a parameterized form of the orifice constant Q₁ in equation F-4. The constant b is typically around 0.5, reflecting the square root dependence of the pressure drop across the orifice (δP) in the theoretical orifice equation. The coefficients a and b are empirically determined constants established during calibrations performed at least annually using an independent flow device. The values of a and b are unique to each sampler module at each calibration. If the cyclone transducer reading is taken from the controller screen, it can be plugged into equation F-7 directly; if the cyclone transducer reading is taken from the flashcard file, it must be divided by 100 before being plugged into equation F-7.

By IMPROVE's definition of standard conditions, the parameters a and b are always calculated relative to 293 K and sea level. One advantage of expressing the parameters relative to sea level is that all modules should have parameters with similar values independent of the site elevation.

F.4. Flow Rate Equation for the Critical Orifice Transducer

The critical orifice is designed to ensure stable flow rates by shielding the flow from vacuum pump fluctuations, but it also provides another opportunity for measuring the flow rate based on the pressure drop across the orifice. Because the downstream side of the critical orifice is at the pump, the downstream pressure can be approximated as zero. Thus, the pressure drop across the critical orifice, δP in equations F-1 through F-4, is simply the pressure just upstream of the orifice.

The mass flow rate at any atmospheric pressure, P, and temperature, T, can be expressed for the critical orifice using a form of equation F-3:

$$Q_m = Q_1 \sqrt{\delta P} \sqrt{\frac{P'}{P_0}} \sqrt{\frac{T_0}{T + 273.15}} \equiv Q_1 \sqrt{P - \Delta P} \sqrt{\frac{P - \Delta P}{P_0}} \sqrt{\frac{T_0}{T + 273.15}} \quad \text{F-8}$$

where P' is the pressure at the critical orifice entrance and ΔP represents the pressure drop across all components of the flow system upstream of the critical orifice (principally across the filter). Thus, (P – ΔP) is the pressure just upstream of the critical orifice, P'. Equation F-8 can be rearranged to give

$$Q_m = Q_1 * P \left(1 - \frac{\Delta P}{P}\right) \sqrt{\frac{1}{P_0}} \sqrt{\frac{T_0}{T + 273.15}} \quad \text{F-9}$$

Using an analogue of equation F-4, the volumetric flow rate, Q, can be calculated as

$$Q = Q_m \left(\frac{P_0}{P}\right) \left(\frac{T + 273.15}{T_0}\right) = Q_1 \sqrt{P} \left(1 - \frac{\Delta P}{P}\right) \sqrt{\frac{P_0}{P}} \sqrt{\frac{T + 273.15}{T_0}} \quad \text{F-10}$$

We can redefine the Q₁ constant to include the local pressure:

$$Q = Q_3 \left(1 - \frac{\Delta P}{P}\right) * F(elev) * f(T), \quad \text{F-11}$$

where Q₃ is another constant.

The primary flow rate measurement for the PM₁₀ module and the secondary measurement for the PM_{2.5} module are made with a transducer that measures ΔP, the differential between atmospheric pressure and the pressure just upstream of the critical orifice. The volumetric flow rate, Q, is calculated from the critical orifice transducer reading, G, according to equation F-12, which is an analogue of equation F-11:

$$Q = (c + d * G) * F(elev) * f(T) \quad \text{F-12}$$

where c and d are constants empirically determined at calibration. The constant d is almost always negative, yielding a relationship similar to that expressed in equation F-11, in which the volumetric flow rate, Q, decreases as the differential pressure reading, G, increases. If the orifice transducer reading is taken from the controller screen, it can be plugged into equation F-12 directly; if the orifice transducer reading is taken from the flashcard file, it must be divided by 100 before being plugged into equation F-12.

Because the constant Q₃ in equation F-11 incorporates the ambient pressure, the constants c and d will be independent of temperature, but not independent of pressure. That is, they would change if the sampler were moved to a new location. In practice, however, any sampler that is moved is also recalibrated, so there is no practical ramification.

F.5. Flow rate Equation for the Audit Devices

The audit device consists of a probe that is inserted into the sampler inlet, diverting the inlet flow through a calibrated orifice. A Magnehelic® gauge measures the pressure drop across the orifice. Each audit device is calibrated in Davis prior to being taken to the field. The reference flow rate for calibrating the audit device is provided by a BIOS DryCal meter located in the

sampler laboratory at UC Davis. Taking the log of equation F-7, the volumetric flow rate equation for the audit device is

$$\log(Q) = a_o + \log \sqrt{\left(\frac{29.92}{P}\right)\left(\frac{T + 273}{293}\right)} + b_o * \log(M_o) \quad \text{F-13}$$

where T and P are the actual temperature and pressure in the laboratory in Davis at the time of the calibration. Applying this pressure and temperature correction ensures that the resulting audit device constants, a_o and b_o , will always be referenced to standard conditions.

The log of the magnehelic gauge reading, M_o , is regressed against the log of the flow rate from the BIOS DryCal meter, Q, for a set of four flow rates covering the normal range of the audit device. The constants relative to the nominal sea level pressure (29.92 in Hg) and 20°C are calculated using

$$a_o = \text{intercept} - \log \sqrt{\left(\frac{29.92}{P}\right)\left(\frac{T + 273}{293}\right)} \quad b_o = \text{slope} . \quad \text{F-14}$$

Once calibrated, the audit device is used in the field during sampler calibration to determine volumetric flow rates referenced to 20°C and sea level. Hence, the audit flow rate in the field, Q_o , is calculated using a version of equation F-7 that does not require the elevation and temperature factors:

$$Q_o = 10^{a_o} M^{b_o} \quad \text{F-15}$$

F.6. References

Baker, Roger C., *An Introductory Guide to Flow Measurement*, 1989 (Mechanical Engineering Publications Limited, London)

Trewartha, Glenn T., *An Introduction to Climate*, 1954 (McGraw-Hill, New York)