A Generalized Approach to Data Quality Assurance and Control of Large, Complex Data Structures Using SAS

by Herbert C. Reynolds
and Donald P. Trees [1]
Viar and Company, Inc.
Alexandria, Virginia 22314

Introduction

The occurrence of errors in data is one of the most serious problems affecting information management and utilization. As a direct result, the identification and resolution of data errors is critical to any information management system. All data acquisition and management efforts utilize some form of error detection or data editing prior to loading or updating information into a data base management or archival system. Although the specific nature of the data and data structures may differ and the required precision and accuracy may vary, the occurrence of errors and systems to control errors seems constant. Approaches and techniques to identify and resolve data errors have general applicability independent of specific data content areas.

The purpose of this paper is to discuss the use of SAS to support a general data quality assurance and control system within a large, complex data acquisition and management effort. The specific application is in the area of analytical environmental data or data produced when environmental samples are analyzed in laboratories to determine the occurrence and concentration of priority pollutants and hazardous substances. The nature of data quality assurance and control and the nature of the data and application are briefly introduced. The rational and effectiveness of using SAS is then presented.

[1]Presented at the International Association for Social Science Information Service and Technology (IASSIST) Conference held in Washington, D.C., U.S.A. on May 26-29, 1988

Finally, the application of specific SAS procedures within each major data quality assurance and control activity is discussed and examples given. Primary SAS features discussed include format tables, multiple record keys, variable length record emulation, macros, conditional output, temporary work files and automatic character/numeric conversion. Each of these capabilities are used for this particular environmental data application, but have general applicability to any data error identification and resolution problem.

Nature of Data Quality Assurance

In large scale data acquisition and management systems, data validation, verification, or quality control editing becomes so important that a separate operation known as data quality assurance and control is usually created. The process is also known as data editing or cleaning. The primary focus of such a system is on the identification and resolution of data errors. Such a process insures that data is of a known quality and within specific data quality control limits.

Three interrelated activities are involved in data quality control or editing:

- Limit specification: developing and stating quality control limits or specifications for defining data errors. Errors are defined by users of data and are not an inherent property of data
- Error identification: identifying and reporting samples or data elements within samples which violate established limits
- Error resolution: developing methods and techniques to resolve issues of data integrity resulting from the identification of a value or values as being inconsistent with or outside the limits.

Each of these three activities may be conducted as a manual or computer assisted process depending on the format of the data being evaluated and designed according to theoretical or empirically based information.

Environmental Data Structures

The basis for conducting most environmental assessment, monitoring, or remedial actions are the results from the laboratory analysis of soil and water samples. Each sample is collected and screened for between 25 to 150 priority pollutants and hazardous substances. Actions are planned according to the compounds identified and the concentration levels determined.

For this particular environmental program, over 80,000 samples per year are analyzed. The number of data elements per sample or analysis may total over 2,000 depending on the method of analysis and the laboratory quality control specifications. The volume of information being checked for a single group of samples is also enormous. Data for only 50 samples involves approximately 900,000 bytes and over 10,000 records. Many different units of collection and analysis occur and include sampling episode or group, sample, method, instrument, and compound. Linkage and data association may involve 6–7 identifiers or keys. Data types and range variability is both numeric and alphabetic with values from 10–7 to 106 including missing data problems.

The applications area for this particular data quality assurance and control system is in the Environmental Protection Agency's Analytical Operations Branch within the Superfund Program and the Office of Emergency and Remedial Response.

SAS as a Generalized Data Quality Assurance Support System

The system discussed in this paper runs on an IBM 3090 mainframe and consists of 2 program libraries, 2 table libraries, and 3 SAS databases (containing over 20 individual SAS files) which reside on both disk and tape. The data checking portion of the system resides in 12 program modules which total over 15,000 lines of code. The entire system is supported by 25 programs totaling over 35,000 lines of code. This application is large and was designed to run on a mainframe. It processes, on the average, 12 million bytes of information per day. Because of the modular design of this system, certain portions can be easily modified to run at the PC level if needed.

The utilization of SAS in developing this data quality assurance system has several advantages over using a procecural language. If procedural languages are used to design data quality control systems in scientific areas, several problems can occur:

- Initial development, frequent modification and necessary documentation by professional programmers represents considerable time and resource expenditures.
- Users, or knowledgeable persons cannot make changes directly. Programmers do not understand the substance and meaning of the data, so lengthy requirements analysis discussions must be conducted in a third language.
- The need for formalized requirements analysis and system documentation again impacts time and resources and diverts the data users time from research, analysis and problem solving to problem representation and monitoring.
- Techniques and approach are not directly generalizable to new applications.

The desired objectives for a data quality assurance and control support system language in scientific areas include:

- A user oriented non-procedural language
- Ease in implementation, extension and modification
- Availability of statistical procedures and functions
- Capabilities for dealing with missing data and scientific notation
- Facility for supporting effectively within the same system users of different expertise levels
- Generalizability to other data and applications.

The following section discusses and demonstrates SAS features and capabilities which are used for this particular application. These features and the general approach can be extended to a variety of data validation and quality and quality control efforts. SAS features and examples of their application to the data, are presented according to each of the major areas of data quality assurance: designing QC limits, error identification and error resolution.

SAS Features Used in Data Quality Control

For each of the major data quality assurance and control areas, the nature of the required activities, functions or problems is presented. Within each activity or problem, the SAS features that are important to this application are then discussed and specific examples presented. Table 1, "SAS Features Used by QA/QC Activity," summarizes the SAS features by activity performed for each of the three applications areas: designing limits, error identification, and error resolution.

Designing Quality Control Limits

Data quality control limits or specifications usually include one range check for each data element or variable and as many logic checks as necessary to evaluate all relationships of interest. A range check defines the expected or "allowable" codes or values for a given data element. A logic specification defines the expected logical relationship between two or more data elements. Logic checks usually involve issues of discrepancy, consistency or invalid relationships between data items.

Within the range and logic classification, two other distinctions can be made. Specifications can be delineated according to the underlying justification or basis for the check: syntax (structure), semantic (meaning) and statistical (distributional properties). Specifications can also be differentiated on the basis of whether the checks occur within a single logical file or across multiple files. The longitudinal checking of data or the comparison of different data sources are examples of multiple file or cross file checking.

Table 1

TABLE 1
FEATURES USED BY QA/QC ACTIVITY

SAS FEATURE	QA/QC ACTIVITIES		
	DESIGNING QC LIMITS	ERROR IDENTIFICATION	ERROR RESOLUTION
FORMAT TABLES	Storing Data Specifications	Retrieving Data Specifications	Retrieving Error Message
MULTIPLE FILE KEYS	Data Linkage	Data Linkage	
VARIABLE LENGTH RECORD EMULATION	Efficient Space Utilization		
SORT AND MERGE	File Manipulation	File Manipulation	File Manipulation
FUNCTIONS	Field Manipulation	Field Manipulation	
MACROS, %INCLUDE, LINK, ETC.		Processing Control	
CONDITIONAL OUTPUT	File Manipulation	File Manipulation	File Manipulation
TEMPORARY WORK FILES	File Manipulation	File Manipulation	File Manipulation
SAS REPORT FEATURE			Error Reporting
MISSING DATA FEATURE		Data Editing	
AUTOMATIC CHAR.-NUM./ NUM.-CHAR. CONV.	Data Editing	Data Editing	
UPDATE PROCEDURE			Correcting Errors on Data
FULL SCREEN FILE EDITING			Correcting Errors on Error Record

If quality control is conducted by another person or group other than the end user, the customary procedure is for the person or group to create all implied range and logic syntax checks. Certain obvious semantic checks might also be proposed. The quality control specifications are then reviewed and confirmed by the user prior to implementation. Additional semantic and statistical checks are contributed by the data user to the general set and complete the logical specifications.

The major component of designing quality control specifications include:

- Defining representing and storing limits and relationships
- Linking files and data elements and
- Storing and reporting results.

The use of SAS features in performing each function is discussed and examples given. Primary SAS features include: format tables, multiple file keys, variable length record emulation, functions, conditional output, temporary work files and automatic character/numerical conversion.

Defining, Representing, and Storing Variable Limits

Defining limits and relationships of variables requires each variable to be examined to determine its appropriate value or range of values. It is also to necessary to determine values depending on some other data qualifier. For example the valid values for variable A is '0' or '1' – these are the only two correct entries which variable A can contain. Variable B, however, contains conditional values: if variable A is '0' then variable B may contain either 'A' or 'B', if variable A is '1' then variable B may contain either 'C', 'D', or 'E'.

All variable comparisons are written:

```
    VAR1 = VARA OR VAR1 = VARB THEN...
```

Conditional edits must be coded:

```
    VAR1 = '0' THEN DO...
 IF VAR1 = VARA OR VAR1 = VARB THEN...

    END
```

Complex conditional statements similar to the following are not uncommon.

```
IF A1 = 'EVALA' AND A2 = 'EVALB' AND A3 = 'EVALC' THEN DO...
   IF A4 = 'INDA' AND A5 = 'INBA' AND A6 = 'TOXAPH' THEN ...
   IF A7 = 'AR1016' AND A8 = 'AR1242' AND A9 = 'AR1248' THEN ...

END
```

Problems are encountered when trying to represent and store these multiple values/limits and conditionally determine the validity of variables. Dozens of possible values may need to be stored and extracted for each variable. Additional problems are encountered when trying to develop the logic and software needed to perform the edits. The main SAS feature used in representing and storing QC limits is the use of SAS format tables.

SAS format tables allow convenient storage and retrieval of data specifications. Values can be extracted from a SAS table and compared to data values for a variable. The use of SAS tables allows storage of specifications in one place that are necessary to many programs. Format tables facilitate organizing and updating specifications, to update a specification, one simply changes the table and reloads the format library. This change is immediately picked up by all programs using this table.

In the current application, edit specifications (i.e. values, limits, and ranges) are stored in SAS format tables. These tables are used to structure and organize the edit specifications. The table consists of a 'key' lookup field followed by one or more values. For example, given the following data specifications for four chemical compounds:

	Variable 1	Variable 2	Variable 3
	Range	Limit	Valid Codes
Compound 1 (code=110)	10-330	19800	M
Compound 2 (code=120)	10-340	19810	LF
Compound 3 (code=130)	50-340	19800	ML
Compound 4 (code=140)	10-300	19800	MLE

These specifications, after being organized and tabulated, can easily be transformed into a SAS format table. The following is an example of the above data specifications converted to a table which is ready to be loaded into a SAS format table.

(1)	(2)	(3)	(4)	(5) ...
110	10	330	19800	M
120	10	340	19810	LF
130	50	340	19800	ML
140	10	300	19800	MLE

This example shows four data specifications for each of the four compounds. Column (1) is the key, or lookup code, for the compound, followed by four columns, each column being a limit or valid code for various fields associated with this compound. Column (2) is the lower limit for variable 1, column (3) is the upper limit for the same variable, column (4) is the upper limit for variable 2 (note that in the specifications, it is necessary to check the upper limit only of variable 2), and column 5 contains the valid codes which variable 3 may contain.

This file is easy to build and maintain – it is simply an 80 byte text file which can be created and maintained by any text editor. This file can then be loaded into SAS table format (by using the SAS format procedure) where it is accessible to all programs.

Multiple tables for various edits and reports may be stored in one file. In this application all format tables are stored in one text file. By adding a 'table code' to the beginning of each line an unlimited number of tables can be stored in one file. The 'table code' signals the SAS program which loads the text tables into SAS format tables to write each group of 'table codes' to a different format table. Keeping all tables for a system in one place can make maintenance and data specification modifications much easier. An example of how this text file is structured is shown in Appendix A.

The Appendix A example consists of several different types of tables. Table #1 stores error codes and error text messages, Tables #2 and #3 store data QC limits, Table #4 stores chemical compound names used in formatting reports, and Table #5 stores more data QC limits. Notice that each table consists of the same components: a table code, a look-up code or key, followed by a text field which may consist of one or more character or numeric fields (up to 40 bytes long). Appendix B is an example of the SAS program which reads the text file (Appendix A) and creates a SAS format table for each text table.

The program in Appendix B builds the SAS code necessary to load each individual text table into a SAS format table. The benefits of using this method of loading tables is the ease with which updates to the data specifications can be made – it also allows the user to control the maintenance of the data specifications. To change edit specifications, the user simply edits the text table file (Appendix A) and submits the code in Appendix B to execute in a batch job. The changes will then be represented in all programs accessing these tables.

Linking Data Elements, Files and Specifications

Related data residing on various files must be linked together by key fields before it can be compared and edited. Problems are encountered when multiple files are involved, each requiring linkage to another file with a different set of key variables. After file linkage is completed, the data specifications must be linked into the correct observations for comparison and data editing. Here the database management capabilities of SAS become important.

The SAS features used in linking data files, records, data elements and specifications include:

- Multiple key fields –
  SAS allows any variable on a SAS file to be used as a key field. Any field which is common to two or more files can be used to link and join observations on these files.

- SAS sort and merge procedures –
  The use of the SAS sort and merge procedures facilitates the linking of observations from various files.

- SAS files – variable length record emulation –
  By storing sets of data in multiple files and then linking these files as needed, variable length record data storage can be emulated in SAS. Because all observations in a given SAS file are the same length and inefficient as far as space utilization is concerned. Redundant fields can be summarized and stored on a separate file to save space.

- Temporary work files -
  Linkages between observations can be made at the time of the edits in temporary work files. After data is extracted from the permanent files it is then linked as required for the particular edits to be performed. Once the edits are performed, the temporary file containing the linked data is erased – this ensures that workspace will be available for later use.

The current application consists of a large collection of variable length data records linked together by various keys. That is, separate data files are used to partition the redundant sections of records. Through careful file design, the large and inefficient data structures are efficiently stored and accessed in SAS files.

File design is very important. SAS files can be inefficient if not designed carefully with the applications in mind. Large amounts of data may become difficult and costly to maintain. Efficient storage can be obtained through variable length record emulation. Many of the data forms received from the labs contain multiple observations with the same value for a specific field. For example, header information applies to many detail records. Redundant fields can be extracted and stored just once in a separate file and merged back when needed using the proper keys. This technique considerably reduces storage space requirements and processing the data for specific applications is more efficient. There are some drawbacks in certain data applications – these drawbacks include the need for additional logical data manipulation operations.

An example of this variable length record emulation in SAS files is shown by the design of the SAS files for 'Form I'. Data received from the laboratories on 'Form I' actually consists of 3 types of data (see example of Form I in Appendix C). The least efficient (but simple) SAS representation of this file would be one SAS observation per page of the form. Keeping in mind how the data will be processed and more importantly storage efficiency, the data may be partitioned into header information and compound results. Further analysis of the forms shows that much of the header information on each page is repetitive and can be summarized and stored in a separate file partition. Ultimately four files resulted from the partitioning of 'Form I': Header Information, Results, TIC Results, and SDG Information (see examples of these files in Appendix D). These four files are processed independently or merged together to create the original form format. The impact on storage efficiency is dramatic compared to the single record case. Storage requirements are reduced by a factor of four.

Extensive editing and data checking requirements require that files be built with multiple linkages. Each observation on each file has explicit linkages to observations on every other file to which it must be linked. Key fields which link units of data differ depending on the units being linked. This can be demonstrated in the four 'Form I' files mentioned above (see Appendix D).

The basic unit of analysis for this application is the sample delivery group (SDG). During the editing process, all data for a given SDG (SDG is the highest level key on all files) is extracted from the master files and placed in temporary work files. As shown in the example files in Appendix D, each file contains the master SDG key (SDG_NO). To link files 1, 2 and 3 together (after extraction by SDG) is a simple merge by the variable 'SAMPLE'. Variables from file 4 (SDG information) are merged in by the variable 'SDG_NO'. Other data files, however, do not contain 'SAMPLE' and must be linked by additional variables. For example, file 5 (instrument tuning information) is linked to file 1 by 'SAMPLE'. File 6 (instrument calibration information) is linked to file 5 by instrument identifier (IN_IDF5, IN_IDF6). The only way to link file 1 to file 6 is by using file 5 as an intermediate link. One file 6 observation applies to many file 1 observations, in other words one 'tune' applies to many 'samples'. This example begins to show some of the complexity of the data.

Observations in each file for each SDG must be checked against corresponding fields on other files. In some cases a single observation on one file may apply to multiple observations on another file. Using SAS sorts and merges, each unit of data can be temporarily linked to its corresponding data on any other file, the edits applied, and error records generated. *Storing and Reporting Results*

Reporting and storing results from complex multi-leveled edits can be very complicated. Multiple error records may be generated from a single data problem and determining which piece of data generated the error can be difficult. The error record must contain all information necessary to fully explain the nature of the error and indicate the location of the observation on which the error resides. Error reports must be written to summarize the errors in an convenient and usable form which can be understood by the user.

The major SAS features used in results reporting include:

- SAS format tables –
  SAS format tables are useful for storing error message text information.

- Data manipulation –
  The ease with which files and observations can be handled in SAS allows various error reports and summaries to be produced from the error records.

- SAS functions –
  SAS functions allow the manipulation of results for reporting purposes.

- Conditional outputs –
  Conditional output statements aid in manipulating data and files.

In the current application, as each error or discrepant data field is encountered an error record is generated. This error record contains all necessary information to fully document the nature of the error. This includes the variable which caused the error, the related variable from which the comparison was made, the location of the observation on which the error resides, the date and time at which the edits were executed and the value which the field should be (if it is possible to calculate). This information allows detailed error reports to be produced and also allows the discrepant record to be located and corrected if desired.

Since many error records can be produced, the error file needs to use space as efficiently as possible. Multiple error messages may need to be created to fully explain a data problem. As the data is checked, each time an error is encountered an error record is output. A discrepant variable on one observation may cause multiple errors to be produced on related observations. For example, in this application, if an instrument calibration run is bad, all samples associated with this calibration run are flagged as in error.

The structure of the error record is shown below.

Key Fields:	SDG number
	Case number
	Sample number
	Date and time of edit
	Additional key field (if necessary)
Error	Error Code
Information:	Related observation and field name (from variable comparisons)
Error	Current value of discrepant field
Correction	Correct value of discrepant field

SAS format tables are used to store the text for the error code which explains the nature of the error. The following is an example of the text storage for several error codes.

Error Code	Text Message
980	Injection date
990	Instrument identifier
1000	Column
1010	Percent decanted incorrect for standard
1020	Preliminary sequence incorrect
1030	Excessive standards between standards
1040	Date sample received outside limits
1050	Sequence not in chronological order
1060	12 hour standard exceeds limits

The table consists of a unique error code or error number followed by a text string which when combined with the other error information on the error record tells the nature of the error and its location on the files. These text messages are decoded from the error code for reporting purposes.

To produce usable error reports, the error records for a particular set of data are selected, analyzed, and summarized. As shown on the sample error report (see Appendix E) the report is summarized by sample and form number. For each error, enough information is presented to the user to locate the field in error on the hardcopy forms. Note that redundant errors are summarized and not displayed. Additional fields are stored on the error record but are not needed on this report. These additional fields will be used later to locate and update the discrepant variables.

Error Identification

Once error specifications are designed and implemented, data can be evaluated accordingly. Each machine readable record and data element is passed against the pre-programmed specifications using SAS. Response fields are checked as to allowable values, consistency and substantial relationships between data elements. Error report listings are generated for data that do not conform to cleaning specifications. Errors must be identified unambiguously according to location in the original hardcopy document or package. Such identification requires the use of multiple keys. The original hardcopy documents, existing policy decision and queries to the generator of this source data maybe used to determine the nature of the error and the appropriate method of resolution.

Major activities performed under the error identification system include:

- Reformatting and restructuring of data
- Retrieval and comparison of tolerance specifications with the data,
- Writing data checks, and
- Unambiguously identifying errors.

Primary SAS features used include format tables, multiple file keys, functions, macros, conditional output, temporary work files, missing data features and automatic character/numerical conversion.

Reformatting and Restructuring the Data

Before data can be checked against tolerance specifications it must first be organized in a form allowing related fields to be compared. With certain types of data this is a complex problem. Analytical results data are reduced from 80 raw data record types to 16 partitioned files (SAS files). These 16 files must be restructured and merged for the various edits.

The major SAS features used to restructure the data are:

- Conditional output to temporary work files
- File sorting and merging

In the current application, data are extracted from the permanent files by SDG (sample delivery group). A batch job is submitted to edit one SDG which reads through the permanent files and extracts all data for this one SDG using conditional outputs, the extracted records are written to temporary work files. All data necessary to execute the edits are then available to the program. The work files are then structured (through sorting and merging) as needed for each edit.

Retrieval and Comparison of Specifications with the Data

Hundreds of variables are edited, each of which may contain multiple discrete values or ranges of values. The retrieval of the data specifications for each variable must be integrated with the edit process to compare these specifications with the data fields. Through the use of SAS format tables to store data specifications, the retrieval of these specifications can be done as needed throughout the edit process.

The SAS features used to retrieve and compare specifications with the data are:

- SAS format tables –
  SAS format tables allow quick revival of data specifications.

- SAS functions –
  The SAS 'SUBSTRING' and 'INDEX' functions are useful in manipulating data specification strings. The 'PUT' function allows retrieval of data specifications from format tables.

- SAS character/numeric conversion handling –
  SAS will automatically convert a character string to a numeric field (and vice versa) if it is necessary for a data comparison.

In this application table specifications are retrieved from the format tables using both the SAS 'PUT' and 'SUBSTR' functions. The following sample of SAS code shows how the low and high limits for a particular variable for one compound are extracted from the format file.

```
 CRQL = PUT(CAS_NO,$CRQL.);
LCRQL = SUBSTR(CRQL,2,5);
HCRQL = SUBSTR(CRQL,8,5);
```

The "PUT" function takes the compound code stored in the variable "CAS-NO" and retrieves the values for that compound from the SAS Format table named "$CRQL". The variable CRQL contains the entire set of limits for that chemical compound. CRQL is divided into the low and high limits through the use of the SAS 'SUBSTR' function. Variables LCRQL and HCRQL now contain the low and high limits for a given variable and can be used for comparisons.

Another feature of SAS which is very helpful in coding edits and data comparisons is the automatic numeric/character and character/numeric conversion. Fields will be automatically converted by SAS if needed. This feature is helpful when retrieving values from format tables because all fields stored in format tables are character.

Writing Data Checks

Writing the code to check the data against other data elements and against specifications is probably the most complex task in the entire editing process. The logic involved and amount of code required to perform the edits is substantial. The SAS features used to write data checks include:

- SAS functions –
  SAS maintains an extensive library of character string manipulation, numeric and date/time handling functions. The use of built-in SAS functions can greatly simplify the storage, retrieval, and comparison of tabled specifications.

- Processing control features –
  The SAS MACRO facility, %INCLUDE, LINK, and conditional output statements give added control over program processing.

- SAS missing data feature –
  Empty or missing values can be recognized by SAS by using special program statements which help simplify the handling of missing data.

Many SAS features were used to write data checks for this application. The large number of data fields to be checked and the varied nature of the checks and data requires various techniques to optimize and make the processing more efficient.

Functions greatly facilitate date/time handling which can be difficult and time consuming. SAS date and time functions can read, store, and format date and time variables in a number of formats.

SAS string manipulation functions like the 'INDEX' and 'INDEXC' functions are used to search strings for specific characters or character strings. SAS 'LEFT' and 'RIGHT' functions are used to justify data within fields before comparisons. The 'COMPRESS' function is used to remove certain characters from fields and the 'TRIM' function is used to remove blank characters.

SAS MACROS, %INCLUDE statement and the LINK statement are used mainly for the execution of redundant data checks. Writing one piece of code and calling it repeatedly (passing different parameters as needed) greatly reduces the amount of coding. GO–TO statements, the IF–THEN–ELSE statement, and various types of DO–LOOPS are used to reduce run time and make the programs run more efficiently. These statements are used to bypass small sections of code which need conditional execution.

Larger sections of code can be skipped by the use of macros and global variables. In this application, for example, there are three general groupings of chemical compounds called fractions. Certain edits are done on all three fractions, and some are done on one specific fraction only. Data packages received may contain only one or two of the three possible fractions. Code bypasses are used to save processing and avoid the program executing code on edits for which there are no data. By checking the data to see which fractions are present and storing this flag in a global variable, decisions to skip entire edits can be made. The following example of SAS code shows how this is done.

```
DATA CHECK;
SET FILE9(KEEP=ONEVAR) END=EDF;
IF EOF THEN DO;
   IF ONEVAR NE ' ' THEN DO;
      CALL SYMPUT('GLOBVAR','YES');
   END;
END; RUN;
```

The above example of code reads a file to see if any data are present, if there are data present then 'YES' is assigned to the global SAS variable 'GLOBVAR'. Later, the editing program executes the following macro to process the desired section of code.

```
%MACRO FILE9;
   IF &GLOBVAR = YES %THEN %DO;
```

```
        %INCLUDE LIB(FILE9);
    END;
```

This macro checks the global variable 'GLOBVAR' and if it is set to 'YES' (which means there is data present to be edited) brings in the code necessary to edit this data (in this example, file 9 data) and executes it. Since the code for editing file 9 is over 500 lines, skipping over this code when there is no file 9 data can save considerable processing time.

Processing missing data can be difficult unless the software being used has features to deal with missing variables. SAS has built-in mechanisms to process missing and blank data fields. These mechanisms free the programmer from writing the code to handle missing data problems. Most SAS functions and procedures automatically handle missing data fields.

Identifying the Nature and Location of the Error

Identifying the exact nature of an error and determining in which file the observation containing the error can be found is a critical issue. Reliable error reports and error resolution can be performed only if error identification is unambiguous. The major SAS features used in identifying the nature and location of an error include:

- Data manipulation features
- SAS format tables

The nature of an error is determined as precisely as possible during the edit using field comparisons and double-checking of values to related fields. As much information as possible is gathered about each error. This information is stored on a permanent error file for subsequent reporting and data correction purposes.

In the current application, a 40 character text field containing a description for each 2 character error code is stored in a format table. The error code written to the error record is 'decoded' or 'formatted' to this 40 character text field in the error report.

Error Resolution

Records or data elements generating error messages are examined during the error resolution process. When the discrepancy that triggered the error message is resolved, the record or data element is updated and the resolution re-checked. Eventually, the error message is resolved by correcting or deleting the data offending the specification, overriding or ignoring the specification for that particular case or sample, or flagging the data value as suspicious and overriding the specification. Copies of all file updates and data flags, the final version of the specifications and the originally received data are kept for documentation. An evaluation of these ancillary files can produce an audit of the quality assurance and control process.

Major activities involved in error resolution include:

- Identification and resolution of errors.
- Correction of the error records.
- Application of corrections to the data.

Primary SAS features used in this activity are format tables, conditional output, temporary work files, SAS report feature, update procedure and full screen file editing.

Identification and Resolution of Key Errors

Through the use of properly designed error reports, errors can be identified and resolved. In certain types of edits, a single error can generate multiple error records. It is important that the error reports unambiguously identify the underlying or primary error so that a single correction will correct the associated errors. Sometimes this will be impossible. A single error source may not be identified, in which case the errors are corrected and the edits rerun. Several iterations of the edit–correction process may be necessary to correct all errors.

The SAS features used to identify and resolve errors are:

- SAS report writing
- Data manipulation features

In this application, error reports are generated using the SAS report writing procedure (see example error report in Appendix E). The SAS print procedure can completely format a detailed report with a minimum of coding. Various SAS data manipulation features are used to extract and 'prepare' the data before they are passed to the print procedure. These error reports are used to locate the specific error in the hardcopy data forms where a decision can be made about its correction and resolution.

Correction of the Error Records

Correction of the error records requires user input through on–line and batch processing. Specific error records must be located and presented to the user for updates. The user–input corrections are stored on the error records until the data files are updated.

The major SAS feature used in making corrections to the error records is SAS full screen file editing.

In the current application, errors are being resolved, but the corrected values are not being entered on the files. Due to the large volume of data being received, the manpower necessary to execute this error correction process is not yet available. When the correction process is implemented, error reports will be displayed to the user through the use of SAS Full Screen editing. This SAS facility allows quick access to SAS files through full–screen terminals. One advantage of using this facility is that SAS will automatically build the file display screen for the user. Updates made to this file through full screen editing are input directly to the permanent error file for storage.

Application of the Corrections to the Data

Applying corrections to the data involves finding the file and observation on which the error exists and replacing the incorrect field with the user input correction. This is done by building into the error record the necessary keys to locate the observation which generated the error. The SAS features used to apply the corrections to the data include:

- SAS file sorting and merging
- SAS update procedure

As stated above, corrections are currently not being applied to the data files. When this process is implemented, the corrected fields on the error records will be applied to the data with the use of the SAS 'UPDATE' procedure. This procedure will automatically update the value of any number of variables on one file with the values of corresponding variables on another file. The error records must first be 'prepared' for the update using some data manipulation, sorts, and merges.

Conclusions

Given the nature of the data and various operational requirements, SAS is an effective system language for designing a data quality assurance process. In the area of scientific and technical data collection and processing, a greater variety and scope of data occurs than in business applications. Data is often collected from multiple sources, on different forms or represented in variable or relational structures. Each new group of data may have different data problems and require the modification or design of new error detection routines or reports. Such systems tend to be dynamic and changes in data and data formats are frequent.

At present over 1,500 data packages have been processed using the described approach. The system appears to be meeting all user requirements. In addition to the QA/QC processing discussed in this paper, the system supports data aquisition, electronic data transfer, file management and data archival, all of which are performed by non-programming personnel. The system also has links to other data base management systems including Natural/ADABAS and FOCUS and has been linked into previously existing financial system written in SAS.

The QA/QC support system has been operating in production mode since late 1987. The final phase, data correction, will go into production in late 1988. All modifications to this system, including the release of enhancements, have been made without interruption of production processing. One by-product of the system has been the availability of the data to various users. A large quantity of raw analytical results data has now been accumulated. Since the data are maintained in

SAS files, statistical analyses by users is greatly facilitated.

SAS has proven to be an appropriate a language for supporting generalized error identification and resolution tasks. Once a quality assurance and control system is structured, knowledgeable users can design and review data checks. Statements may be added, modified, or deleted easily and quickly. The same general system features and structure can be extended to other data and data acquisition efforts. If the system becomes more static and routine, patching in a procedural language such as PL/1 or C can easily be done to optimize code and increase operational efficiency. The techniques and approach discussed will be used for future applications which require extensive data quality control and editing procedures prior to storage and utilization.⌑

Appendix A : Five SAS tables stored in one text file

```
TABLE VALUE-----  LABEL-----------------------------------  COMMENT----

TABLE #1: ERRORS

TABLE ERROR CODE  ERROR TEXT
 01        10     ANALYSIS DATE
 01        20     ANALYSIS DATE-BLANK
 01        30     ANALYSIS DATE-BLANK FOR CLOUMN 2
 01        31     ANALYSIS DATE- FOR COLUMN 2
 01        40     ANALYSIS DATE- HEADER
 01        50     ANALYSIS DATE-CONT. CALIBRATION

TABLE #2: $SURRQC - SURROGATE RECOVERY ADVISORY QC LIMITS

TABLE SURR. CODE  - 2A -] 2B ]- 2C -] 2D ]- 2E-]- 2F -
 02   S1          088110081117035114023120024154020150
 02   S2          086115074121043116030115
 02   S3          076114070121033141018137
 02   S4                      010094024113
 02   S5                      021100025121
 02   S6                      010123019122

TABLE #3: $MSMSDQC - MATRIX SPIKE/MATRIX SPIKE RECOVERY QC LIMITS

TABLE CAS_NO      RPD REC-REC RPD REC-REC ORD
 03   75354       014 061 145 022 059 172  01
 03   79016       014 071 120 024 062 137  02
 03   71432       011 076 127 021 066 142  03
 03   108883      013 076 125 021 059 139  04
 03   108907      013 075 130 021 060 133  05

TABLE #4: $CMPDNAME - COMPOUND NAMES

VOLATILE COMPOUNDS
TABLE CAS_NO      COMPOUND NAME---------------------------]
 04   74873       CHLOROMETHANE
 04   74839       BROMOMETHANE
 04   75014       VINYL CHLORIDE
 04   75003       CHLOROETHANE
 04   75092       METHYLENE CHLORIDE

TABLE #5: $CCCSPCC - THIS IS FOR FORM 6 MIN RRF, MAX % RSD
                     AND FOR FORM 7 MIN RRF50, MAX % D

TABLE CAS_NO      RRFLIM  RSDLIM  RRF50LIM  DLIM
 07   74873        0.30             0.30
 07   75014                30.0               25.0
 07   75354                30.0               25.0
 07   75343        0.30             0.30
 07   67663                30.0               25.0
 07   78875                30.0               25.0
 07   75252        0.250            0.250
 07   79345        0.30             0.30
```

Appendix B : SAS program to load SAS format tables

```
DATA TAB(KEEP=TABLE VALUE LABEL);
INFILE TABLES;
LENGTH TABLE 2. VALUE $10. LABEL $40.;
KEEP TABLE VALUE LABEL;
INPUT @2 TAB_ID $CHAR2. @7 VALUE $CHAR10. @19 LABEL $CHAR40.;
IF INDEXC(TAB_ID,'123456789') GT 0 THEN DO;
   TABLE = TAB_ID;
   OUTPUT;
END;

PROC SORT DATA=TAB;
BY TABLE;

DATA _NULL_;
SET TAB END=EOF;
BY TABLE;
FILE TEMP1;
IF FIRST.TABLE AND TABLE EQ 1 THEN DO;
   PUT @1 'PROC FORMAT DDNAME=FORMAT;';
   PUT @1 'VALUE ERRORS';
END;
IF FIRST.TABLE AND TABLE EQ 2 THEN DO;
   PUT @1 'PROC FORMAT DDNAME=FORMAT;';
   PUT @1 'VALUE $SURRQC';
END;
IF FIRST.TABLE AND TABLE EQ 3 THEN DO;
   PUT @1 'PROC FORMAT DDNAME=FORMAT;';
   PUT @1 'VALUE $MSMSDQC';
END;
IF FIRST.TABLE AND TABLE EQ 4 THEN DO;
   PUT @1 'PROC FORMAT DDNAME=FORMAT;';
   PUT @1 'VALUE $CMPDNAME';
END;
IF FIRST.TABLE AND TABLE EQ 5 THEN DO;
   PUT @1 'PROC FORMAT DDNAME=FORMAT;';
   PUT @1 'VALUE $CCCSPCC';
END;
IF FIRST.TABLE AND TABLE EQ 6 THEN DO;
   PUT @1 'PROC FORMAT DDNAME=FORMAT;';
   PUT @1 'VALUE $CRQL';
END;
IF FIRST.TABLE AND TABLE EQ 7 THEN DO;
   PUT @1 'PROC FORMAT DDNAME=FORMAT;';
   PUT @1 'VALUE $CMPDTAB';
END;
PUT @3 VALUE $CHAR10. @15 '="' @17 LABEL $CHAR40. '"';
IF LAST.TABLE THEN PUT ';';
IF EOF THEN PUT ';';
RUN;

OPTIONS DQUOTE;
%INCLUDE TEMP1;
RUN;
```

Appendix C : Data collection forms - form 1a

1A
VOLATILE ORGANICS ANALYSIS DATA SHEET

EPA SAMPLE NO.

Lab Name:__________ Contract:__________

Lab Code: ______ Case No.: ______ SAS No.: ______ SDG No.: ______

Matrix: (soil/water)______ Lab Sample ID: ______

Sample wt/vol: ______(g/mL)____ Lab File ID: ______

Level: (low/med) _____ Date Received: ______

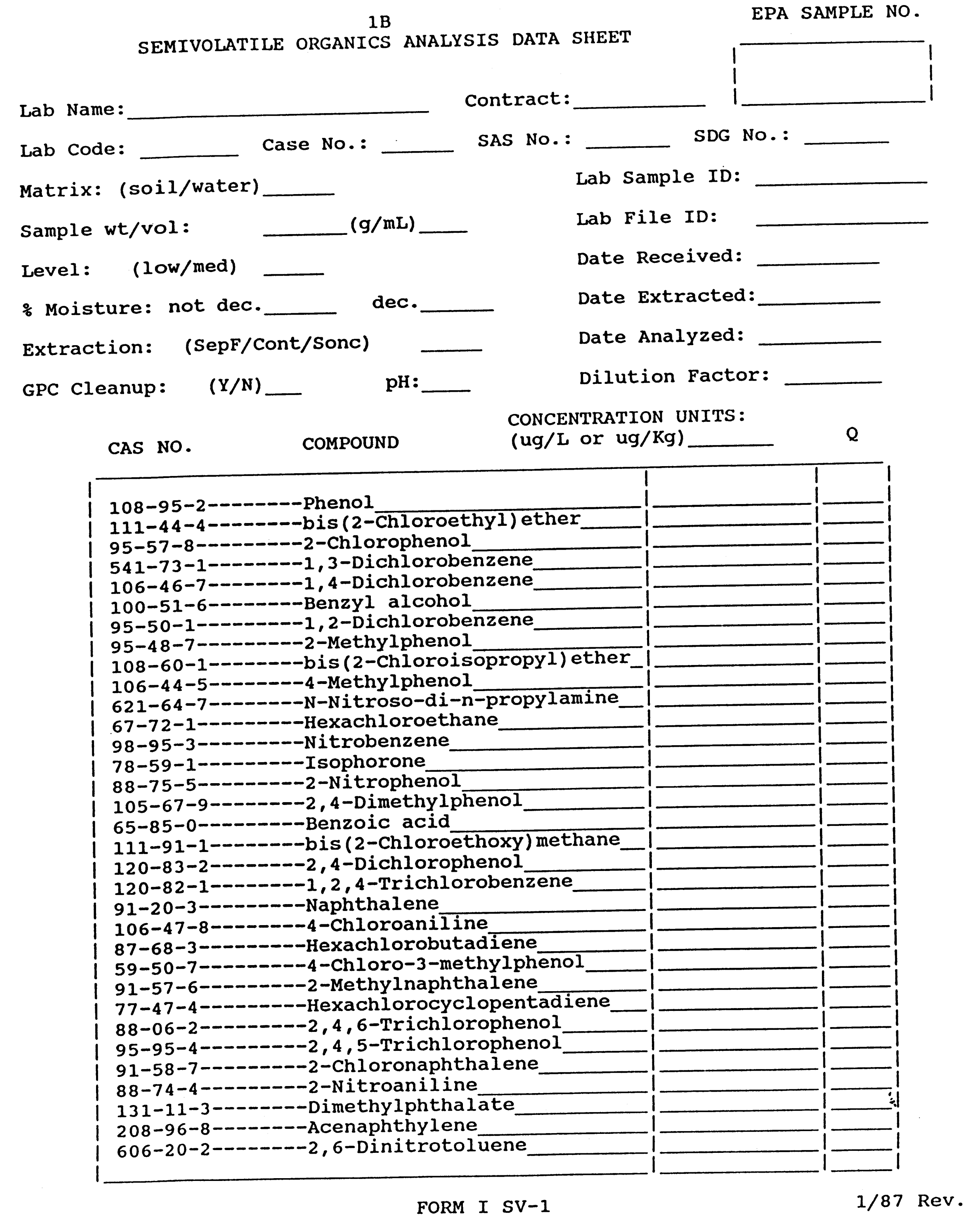
% Moisture: not dec.______ Date Analyzed: ______

Column: (pack/cap) _____ Dilution Factor: ______

CAS NO.	COMPOUND	CONCENTRATION UNITS: (ug/L or ug/Kg)_______	Q
74-87-3	Chloromethane		
74-83-9	Bromomethane		
75-01-4	Vinyl Chloride		
75-00-3	Chloroethane		
75-09-2	Methylene Chloride		
67-64-1	Acetone		
75-15-0	Carbon Disulfide		
75-35-4	1,1-Dichloroethene		
75-34-3	1,1-Dichloroethane		
540-59-0	1,2-Dichloroethene (total)		
67-66-3	Chloroform		
107-06-2	1,2-Dichloroethane		
78-93-3	2-Butanone		
71-55-6	1,1,1-Trichloroethane		
56-23-5	Carbon Tetrachloride		
108-05-4	Vinyl Acetate		
75-27-4	Bromodichloromethane		
78-87-5	1,2-Dichloropropane		
10061-01-5	cis-1,3-Dichloropropene		
79-01-6	Trichloroethene		
124-48-1	Dibromochloromethane		
79-00-5	1,1,2-Trichloroethane		
71-43-2	Benzene		
10061-02-6	trans-1,3-Dichloropropene		
75-25-2	Bromoform		
108-10-1	4-Methyl-2-Pentanone		
591-78-6	2-Hexanone		
127-18-4	Tetrachloroethene		
79-34-5	1,1,2,2-Tetrachloroethane		
108-88-3	Toluene		
108-90-7	Chlorobenzene		
100-41-4	Ethylbenzene		
100-42-5	Styrene		
1330-20-7	Xylene (total)		

FORM I VOA 1/87 Rev.

Appendix C : Data collection forms - form 1b



1B
SEMIVOLATILE ORGANICS ANALYSIS DATA SHEET

EPA SAMPLE NO. ____________

Lab Name: ______________ Contract: __________

Lab Code: ________ Case No.: ______ SAS No.: _______ SDG No.: _______

Matrix: (soil/water) ______ Lab Sample ID: ____________

Sample wt/vol: _______ (g/mL) ____ Lab File ID: ____________

Level: (low/med) _____ Date Received: __________

% Moisture: not dec. ______ dec. ______ Date Extracted: __________

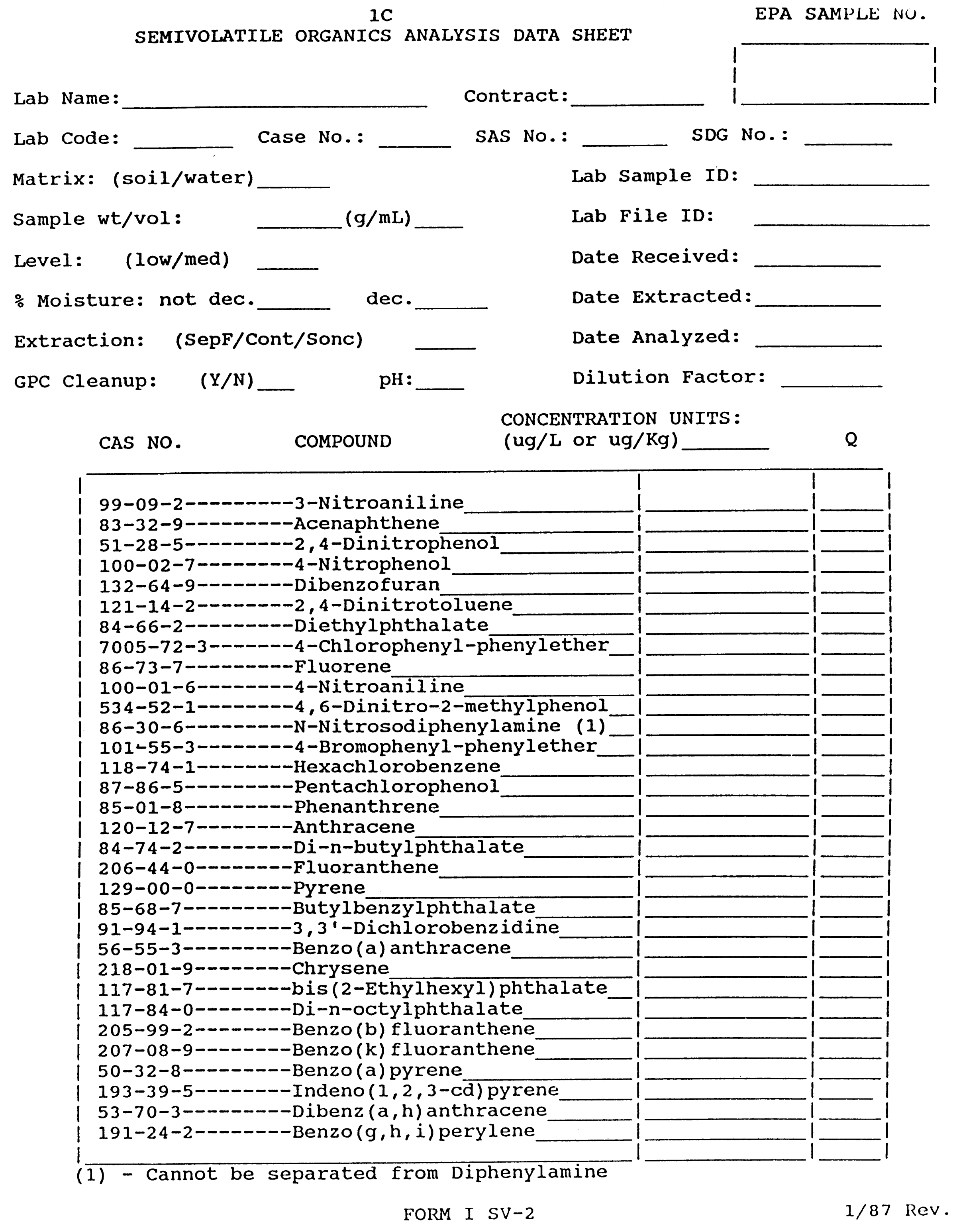
Extraction: (SepF/Cont/Sonc) _____ Date Analyzed: __________

GPC Cleanup: (Y/N) ___ pH: ____ Dilution Factor: ________

CAS NO.	COMPOUND	CONCENTRATION UNITS: (ug/L or ug/Kg) _______	Q
108-95-2	Phenol		
111-44-4	bis(2-Chloroethyl)ether		
95-57-8	2-Chlorophenol		
541-73-1	1,3-Dichlorobenzene		
106-46-7	1,4-Dichlorobenzene		
100-51-6	Benzyl alcohol		
95-50-1	1,2-Dichlorobenzene		
95-48-7	2-Methylphenol		
108-60-1	bis(2-Chloroisopropyl)ether		
106-44-5	4-Methylphenol		
621-64-7	N-Nitroso-di-n-propylamine		
67-72-1	Hexachloroethane		
98-95-3	Nitrobenzene		
78-59-1	Isophorone		
88-75-5	2-Nitrophenol		
105-67-9	2,4-Dimethylphenol		
65-85-0	Benzoic acid		
111-91-1	bis(2-Chloroethoxy)methane		
120-83-2	2,4-Dichlorophenol		
120-82-1	1,2,4-Trichlorobenzene		
91-20-3	Naphthalene		
106-47-8	4-Chloroaniline		
87-68-3	Hexachlorobutadiene		
59-50-7	4-Chloro-3-methylphenol		
91-57-6	2-Methylnaphthalene		
77-47-4	Hexachlorocyclopentadiene		
88-06-2	2,4,6-Trichlorophenol		
95-95-4	2,4,5-Trichlorophenol		
91-58-7	2-Chloronaphthalene		
88-74-4	2-Nitroaniline		
131-11-3	Dimethylphthalate		
208-96-8	Acenaphthylene		
606-20-2	2,6-Dinitrotoluene		

FORM I SV-1 1/87 Rev.

Appendix C : Data collection forms - form 1c



1C
SEMIVOLATILE ORGANICS ANALYSIS DATA SHEET

EPA SAMPLE NO.
____

Lab Name:__________ Contract:__________

Lab Code: ______ Case No.: ______ SAS No.: ______ SDG No.: ______

Matrix: (soil/water)______ Lab Sample ID: __________

Sample wt/vol: ______(g/mL)____ Lab File ID: __________

Level: (low/med) ______ Date Received: __________

% Moisture: not dec.______ dec.______ Date Extracted:__________

Extraction: (SepF/Cont/Sonc) ______ Date Analyzed: __________

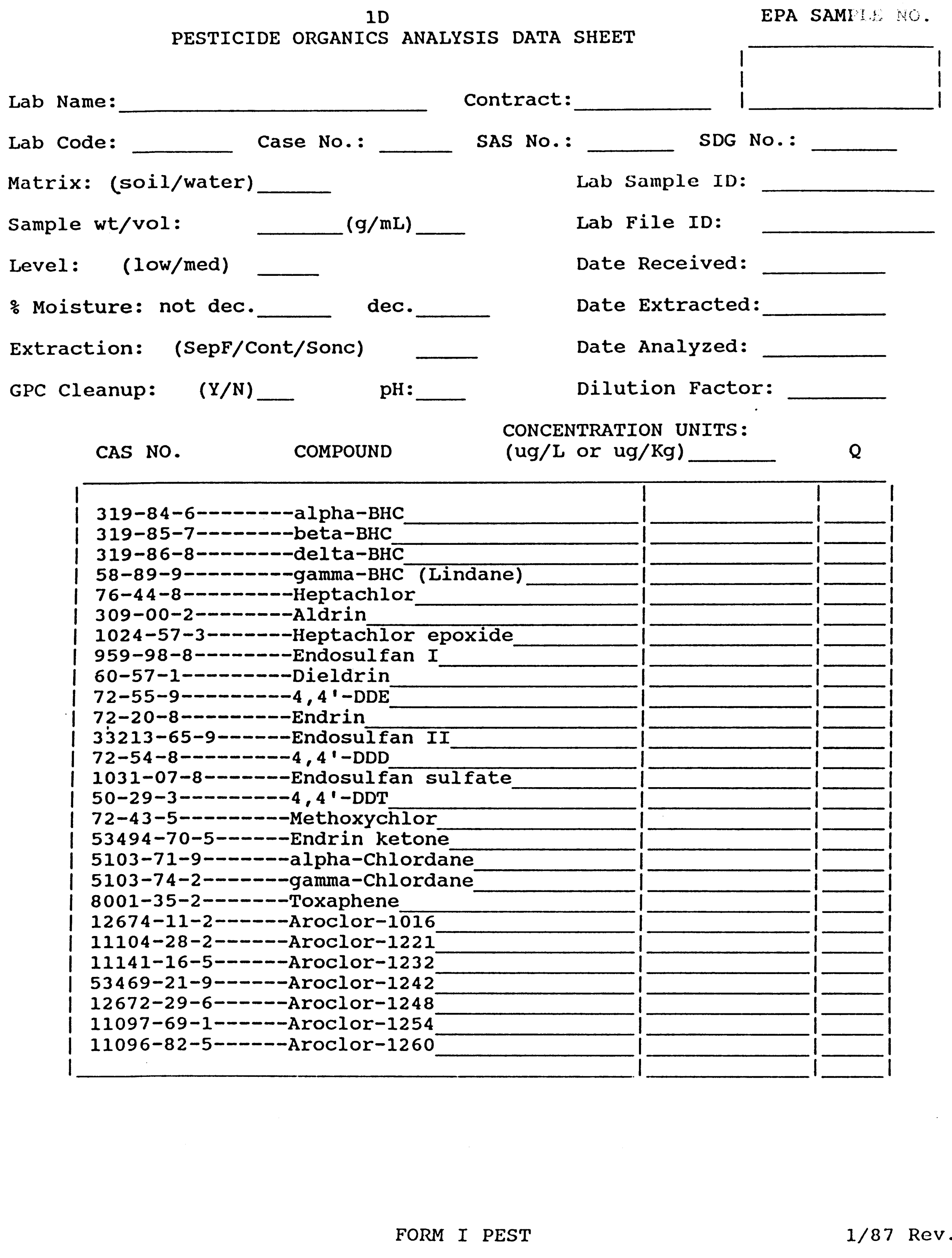
GPC Cleanup: (Y/N)___ pH:____ Dilution Factor: ________

CAS NO.	COMPOUND	CONCENTRATION UNITS: (ug/L or ug/Kg)_______	Q
99-09-2	3-Nitroaniline		
83-32-9	Acenaphthene		
51-28-5	2,4-Dinitrophenol		
100-02-7	4-Nitrophenol		
132-64-9	Dibenzofuran		
121-14-2	2,4-Dinitrotoluene		
84-66-2	Diethylphthalate		
7005-72-3	4-Chlorophenyl-phenylether		
86-73-7	Fluorene		
100-01-6	4-Nitroaniline		
534-52-1	4,6-Dinitro-2-methylphenol		
86-30-6	N-Nitrosodiphenylamine (1)		
101-55-3	4-Bromophenyl-phenylether		
118-74-1	Hexachlorobenzene		
87-86-5	Pentachlorophenol		
85-01-8	Phenanthrene		
120-12-7	Anthracene		
84-74-2	Di-n-butylphthalate		
206-44-0	Fluoranthene		
129-00-0	Pyrene		
85-68-7	Butylbenzylphthalate		
91-94-1	3,3'-Dichlorobenzidine		
56-55-3	Benzo(a)anthracene		
218-01-9	Chrysene		
117-81-7	bis(2-Ethylhexyl)phthalate		
117-84-0	Di-n-octylphthalate		
205-99-2	Benzo(b)fluoranthene		
207-08-9	Benzo(k)fluoranthene		
50-32-8	Benzo(a)pyrene		
193-39-5	Indeno(1,2,3-cd)pyrene		
53-70-3	Dibenz(a,h)anthracene		
191-24-2	Benzo(g,h,i)perylene		

(1) - Cannot be separated from Diphenylamine

FORM I SV-2 1/87 Rev.

Appendix C : Data collection forms – form 1d



1D
PESTICIDE ORGANICS ANALYSIS DATA SHEET

EPA SAMPLE NO.

Lab Name:________________ Contract:__________

Lab Code: ________ Case No.: ______ SAS No.: _______ SDG No.: _______

Matrix: (soil/water)______ Lab Sample ID: ______________

Sample wt/vol: _______(g/mL)____ Lab File ID: ______________

Level: (low/med) _____ Date Received: __________

% Moisture: not dec.______ dec.______ Date Extracted:__________

Extraction: (SepF/Cont/Sonc) _____ Date Analyzed: __________

GPC Cleanup: (Y/N)___ pH:____ Dilution Factor: ________

CAS NO.	COMPOUND	CONCENTRATION UNITS: (ug/L or ug/Kg)_______	Q
319-84-6	alpha-BHC		
319-85-7	beta-BHC		
319-86-8	delta-BHC		
58-89-9	gamma-BHC (Lindane)		
76-44-8	Heptachlor		
309-00-2	Aldrin		
1024-57-3	Heptachlor epoxide		
959-98-8	Endosulfan I		
60-57-1	Dieldrin		
72-55-9	4,4'-DDE		
72-20-8	Endrin		
33213-65-9	Endosulfan II		
72-54-8	4,4'-DDD		
1031-07-8	Endosulfan sulfate		
50-29-3	4,4'-DDT		
72-43-5	Methoxychlor		
53494-70-5	Endrin ketone		
5103-71-9	alpha-Chlordane		
5103-74-2	gamma-Chlordane		
8001-35-2	Toxaphene		
12674-11-2	Aroclor-1016		
11104-28-2	Aroclor-1221		
11141-16-5	Aroclor-1232		
53469-21-9	Aroclor-1242		
12672-29-6	Aroclor-1248		
11097-69-1	Aroclor-1254		
11096-82-5	Aroclor-1260		

FORM I PEST 1/87 Rev.

Appendix C : Data collection forms - form 1e

1E
VOLATILE ORGANICS ANALYSIS DATA SHEET
TENTATIVELY IDENTIFIED COMPOUNDS

EPA SAMPLE NO.

Lab Name: ____________ Contract: ____________

Lab Code: ______ Case No.: ______ SAS No.: ______ SDG No.: ______

Matrix: (soil/water) ______ Lab Sample ID: ______

Sample wt/vol: ______ (g/mL) ______ Lab File ID: ______

Level: (low/med) ______ Date Received: ______

% Moisture: not dec. ______ Date Analyzed: ______

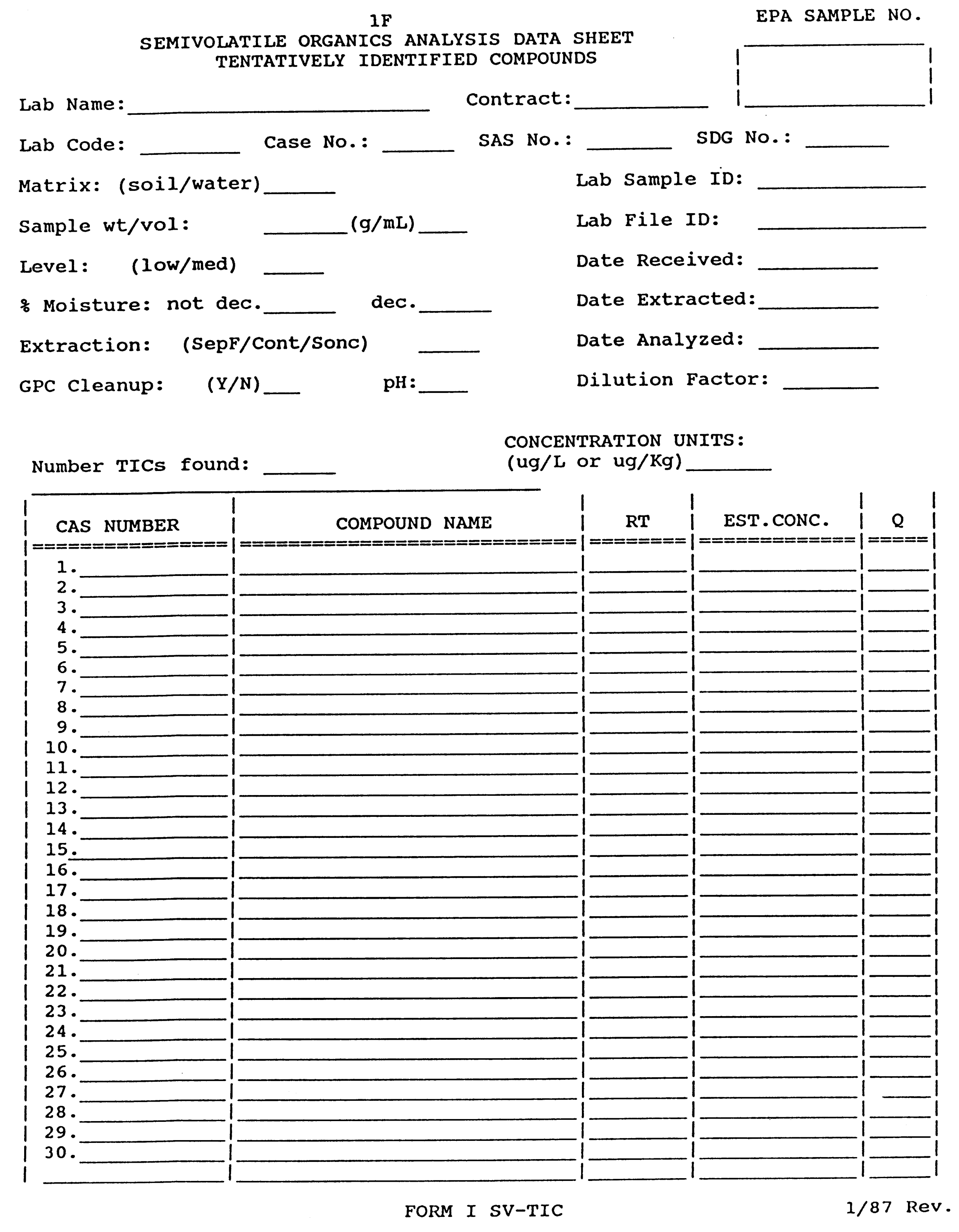
Column: (pack/cap) ______ Dilution Factor: ______

CONCENTRATION UNITS:
Number TICs found: ______ (ug/L or ug/Kg) ______

CAS NUMBER	COMPOUND NAME	RT	EST. CONC.	Q
1.				
2.				
3.				
4.				
5.				
6.				
7.				
8.				
9.				
10.				
11.				
12.				
13.				
14.				
15.				
16.				
17.				
18.				
19.				
20.				
21.				
22.				
23.				
24.				
25.				
26.				
27.				
28.				
29.				
30.				

FORM I VOA-TIC 1/87 Rev.

Appendix C : Data collection forms – form 1f



1F
SEMIVOLATILE ORGANICS ANALYSIS DATA SHEET
TENTATIVELY IDENTIFIED COMPOUNDS

EPA SAMPLE NO.

Lab Name:____________________ Contract:__________

Lab Code: ________ Case No.: ______ SAS No.: _______ SDG No.: _______

Matrix: (soil/water)______ Lab Sample ID: ______________

Sample wt/vol: _______(g/mL)____ Lab File ID: ______________

Level: (low/med) _____ Date Received: __________

% Moisture: not dec.______ dec.______ Date Extracted:__________

Extraction: (SepF/Cont/Sonc) _____ Date Analyzed: __________

GPC Cleanup: (Y/N)___ pH:____ Dilution Factor: ________

CONCENTRATION UNITS:
(ug/L or ug/Kg)_______

Number TICs found: ______

CAS NUMBER	COMPOUND NAME	RT	EST.CONC.	Q
1.				
2.				
3.				
4.				
5.				
6.				
7.				
8.				
9.				
10.				
11.				
12.				
13.				
14.				
15.				
16.				
17.				
18.				
19.				
20.				
21.				
22.				
23.				
24.				
25.				
26.				
27.				
28.				
29.				
30.				

FORM I SV-TIC 1/87 Rev.

Appendix D : Example sas files

FILE 1
HEADER INFORMATION

#	VARIABLE	TYPE	LENGTH
14	ADATEF1	NUM	8
17	CO_UNITS	CHAR	5
15	COLUMN1	CHAR	4
16	DILUTION	NUM	8
24	DRD	NUM	8
19	EXTDATE1	NUM	8
20	EXTRACT1	CHAR	4
10	FILE_ID1	CHAR	14
1	FORM	CHAR	3
21	GPC	CHAR	1
11	LEVEL	CHAR	3
6	MATRIX	CHAR	5
23	NO_TICS	CHAR	2
13	P_MOIST	NUM	8
18	P_MOISTD	NUM	8
22	PH_	NUM	8
12	RECDATE	NUM	8
7	SAMP_ID1	CHAR	12
8	SAMP_WT	NUM	8
3	SAMPLE	CHAR	12
4	SAS_NO	CHAR	6
5	SDG_NO	CHAR	5
2	SUFFIX	CHAR	2
9	WT_UNITS	CHAR	2

FILE 2
RESULTS FILE

#	VARIABLE	TYPE	LENGTH
5	CAS_NO	CHAR	10
8	DRD	NUM	8
1	FORM	CHAR	3
7	QUAL	CHAR	5
6	RESULT1	NUM	8
3	SAMPLE	CHAR	12
4	SDG_NO	CHAR	5
2	SUFFIX	CHAR	2

FILE 3
TIC RESULTS

#	VARIABLE	TYPE	LENGTH
5	CAS_NO	CHAR	10
9	COMPOUND	CHAR	28
11	DRD	NUM	8
1	FORM	CHAR	3
6	QUAL	CHAR	5
7	RESULT2	NUM	8
10	RT	NUM	8
3	SAMPLE	CHAR	12
4	SDG_NO	CHAR	5
8	SEQUENCE	CHAR	2
2	SUFFIX	CHAR	2

FILE 4
SDG INFORMATION

#	VARIABLE	TYPE	LENGTH
5	CASE_NO	CHAR	5
2	CONTRACT	CHAR	11
6	DRD	NUM	8
3	LAB_CODE	CHAR	6
1	LAB_NAME	CHAR	25
7	LOADDATE	NUM	8
4	SDG_NO	CHAR	5

FILE 5
TUNING INFORMATION

#	VARIABLE	TYPE	LENGTH
46	ADATEF5	NUM	8
47	ATIMEF5	NUM	8
7	COLUMN5	CHAR	4
49	DRD	NUM	8
45	FILE_ID5	CHAR	14
1	FORM	CHAR	3
11	IN_IDF5	CHAR	10
10	INJ_DATE	NUM	8
12	INJ_TIME	NUM	8
6	LEVEL	CHAR	3
5	MATRIX	CHAR	5
32	ME127	NUM	8
17	ME173	NUM	8
18	ME173M	NUM	8
19	ME174	NUM	8
20	ME175	NUM	8
21	ME175M	NUM	8
22	ME176	NUM	8
23	ME176M	NUM	8
24	ME177	NUM	8
25	ME177M	NUM	8
33	ME197	NUM	8
34	ME198	NUM	8
35	ME199	NUM	8
36	ME275	NUM	8
37	ME365	NUM	8
38	ME441	NUM	8
39	ME442	NUM	8
40	ME443	NUM	8
41	ME443M	NUM	8
13	ME50	NUM	8
26	ME51	NUM	8
27	ME68	NUM	8
28	ME68M	NUM	8
29	ME69	NUM	8
30	ME70	NUM	8
31	ME70M	NUM	8
14	ME75	NUM	8
15	ME95	NUM	8
16	ME96	NUM	8
8	PAGE	CHAR	1
48	PAGETOT	CHAR	1
44	SAMP_ID5	CHAR	12
43	SAMPLE	CHAR	12
3	SAS_NO	CHAR	6
4	SDG_NO	CHAR	5
42	SEQUENCE	CHAR	2
2	SUFFIX	CHAR	2
9	TFILE_ID	CHAR	14

FILE 6
CALIBRATION INFORMATION

1	AVG_RRF6	NUM	8
26	BHRCTO	NUM	8
24	CAS_NO	CHAR	10
8	COLUMN6	CHAR	4
25	DRD	NUM	8
2	FORM	CHAR	3
12	F6F_ID1	CHAR	14
13	F6F_ID2	CHAR	14
14	F6F_ID3	CHAR	14
15	F6F_ID4	CHAR	14
16	F6F_ID5	CHAR	14
10	ICDATE16	NUM	8
11	ICDATE26	NUM	8
9	IN_IDF6	CHAR	10
7	LEVEL	CHAR	3
6	MATRIX	CHAR	5
27	PKRCTO	NUM	8
17	RRF1	NUM	8
18	RRF2	NUM	8
19	RRF3	NUM	8
20	RRF4	NUM	8
21	RRF5	NUM	8
23	RSD	NUM	8
4	SAS_NO	CHAR	6
5	SDG_NO	CHAR	5
1	SEQUENCE	CHAR	2
3	SUFFIX	CHAR	2
28	VHRCTO	NUM	8

Appendix E : Error report

```
                              O A R Q   E R R O R   R E P O R T          13:47 TUESDAY, MAY 24, 1988    1
                                          F O R
                          LAB: LAB1          SDG: EW341      FORMAT:'A'

---------------------------------------------------- SAMPLE=A FORM =6A ----------------------------------------------------

OBS  FORM    SEQUENCE NUMBER           ERROR  ERROR                       CURRENT VALUE  CORRECT VALUE  SECONDARY   COMPARISON
     SUFFIX  OR COMPOUND NAME          NUMBER                                                           IDENTIFIER  BETWEEN
                                                                                                                    FORMS

  1   AA                                430   INSTRUMENT ID                              VG#2                       5A VS 1A
  2   AA                                903   MS/MSD MORE FORM3 NEEDED    MISSING        FORM3???                    1B
  3   AA     VINYL CHLORIDE             140   COLUMN
  4          =>=>=>=>=>=>=>=>                 NUMBER OF ADDITIONAL        RECORDS OF     THIS TYPE        NOT       DISPLAYED    34
  5   AA     TRANS-1,3-DICHLOROPROPENE  172   COMPOUND MISSPELLED
  6   AA     CHLOROMETHANE              360   FILE ID                     MISSING        RRF20
  7   AA     CHLOROMETHANE              410   INITIAL CALIBRATION DATE    MISSING
  8          =>=>=>=>=>=>=>=>                 NUMBER OF ADDITIONAL        RECORDS OF     THIS TYPE        NOT       DISPLAYED   182
  9   AB     CHLOROMETHANE              410   INITIAL CALIBRATION DATE    MISSING
 10   AB     VINYL CHLORIDE             430   INSTRUMENT ID               MISSING
 11   AB     2-HEXANONE                 70    AVERAGE RRF                 .307           .549                       7A VS 6A
 12   AA            133027              90    CAS NUMBER                  133            027            NOT VALID
 13   AC     BIS(2-CHLOROETHYL)ETHER    1070  RESULT                      780            1600
 14   AB              2                 1220  SAMPLE NO.                  MISSING        EW341RE
 15   AB            133027              90    CAS NUMBER                  133            027            NOT VALID
 16   AA     CHLOROETHANE               70    AVERAGE RRF                 0.684          0.606                         7A  VS. 6A
 17   AC            133027              90    CAS NUMBER                  133            027            NOT VALID
 18   AC                                130   CONCENTRATION UNITS         UG/KG          UG/L
 20   AC     3                          650   M/E176                      <> LIMIT
 21   AC                                380   FORM NUMBER
 22   AC     4                          140   COLUMN                      PAC            PACK
 23   AC                                1201  SAMPLE ID-BLANK MISSIN G
 24   AC     GAMMA-BHC                  890   MS PERCENT OUT                             *


---------------------------------------------------- SAMPLE=B FORM =6A ----------------------------------------------------

OBS  FORM    SEQUENCE NUMBER           ERROR  ERROR                       CURRENT VALUE  CORRECT VALUE  SECONDARY   COMPARISON
     SUFFIX  OR COMPOUND NAME          NUMBER                                                           IDENTIFIER  BETWEEN
                                                                                                                    FORMS

  1   AA     TRICHLOROETHANE            960   MSD %RPD                    10             -11
```