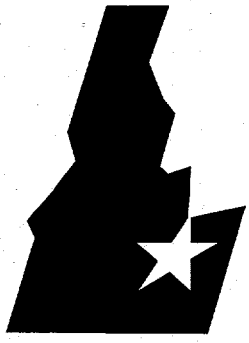

Idaho
National
Engineering
Laboratory

INEL-95/0454

September 1995

Version 1.00 Programmer's Tools Used in Constructing the INEL RML/Analytical Radiochemistry Sample Tracking Database and its User Interface

D. A. Femec

Lockheed
Idaho Technologies Company

INEL-95/0454

Version 1.00 Programmer's Tools Used in Constructing the INEL RML/Analytical Radiochemistry Sample Tracking Database and its User Interface

D. A. Femec

Published September 1995

Idaho National Engineering Laboratory
Nuclear Engineering Department
Lockheed Idaho Technologies Company
Idaho Falls, Idaho 83415-7111

Prepared for the
U.S. Department of Energy
Under DOE Idaho Field Office
Contract DE-AC07-94ID13223

ABSTRACT

This report describes two code-generating tools used to speed design and implementation of relational databases and user interfaces: CREATE_SCHEMA and BUILD_SCREEN. CREATE_SCHEMA produces the SQL commands that actually create and define the database. BUILD_SCREEN takes templates for data entry screens and generates the screen management system routine calls to display the desired screen. Both tools also generate the related FORTRAN declaration statements and precompiled SQL calls. Included with this report is the source code for a number of FORTRAN routines and functions used by the user interface. This code is broadly applicable to a number of different databases.

DISCLAIMER

DISCLAIMER

Portions of this document may be illegible electronic image products. Images are produced from the best available original document.

CONTENTS

1. INTRODUCTION 1

1.1 Purpose 1

1.2 Overview 1

1.3 Definitions, Acronyms, and Abbreviations 1

1.4 Registered Trademarks and Copyrights 1

2. DESIGN OF THE DATABASE 2

2.1 Basic Description 2

2.1.1 Tables and their Fields and Indexes for Sorting 2

2.1.2 Domains and Attributes 2

2.1.3 Storage Areas and Snapshot Files 2

2.2 Specifics 3

2.2.1 Names of Tables, Storage Areas, and Snapshot Files 3

2.2.2 Names and Descriptions of Domains 3

2.2.3 Backing Up the Database 4

2.2.4 Removal of Inactive Entries from the Database 5

3. DESIGN OF THE USER INTERFACE 6

3.1 Basic Description 6

3.2 Specifics 6

3.2.1 Data Storage Variables and Arrays 6

3.2.1.1 Storage of Data 6

3.2.1.2 Conversion of Character Data to Numerical Data 7

3.2.1.3 Error Checking 7

3.2.1.3.1 "a" for Anything 7

3.2.1.3.2 "d" for Date 7

3.2.1.3.3 "l" for Length 9

3.2.1.3.4 "t" for Time 9

3.2.1.3.5 "u" for Units of Time 9

3.2.1.3.6 "?" for Questions 9

3.2.2 Communication between the User Interface and the Database 9

3.2.2.1 Forms of SQL Used 9

3.2.2.2 Commencing a Database Transaction 9

3.2.2.3 Add a Sample Entry 10

3.2.2.4 Search for a Sample Entry 11

3.2.2.5 Due Date Checks 12

3.2.2.6 Update a Sample Entry 13

4. TOOLS USED IN DESIGNING THE DATABASE 15

4.1 CREATE_SCHEMA 15

4.1.1 Purpose of the Routine 15

4.1.2 Flow Description of the Routine 15

4.1.2.1 Creating the Domains 15

4.1.2.2 Creating the Tables 15

4.1.2.3 Creating the Schema 16

4.1.2.4 Associating Attributes with Data Fields 16

4.1.3 Input Files 17

4.1.3.1 ENTITY.PRN 17

4.1.3.2 SCREEN.PRN 17

4.1.4 Output Files 17

5. TOOLS USED IN DESIGNING THE USER INTERFACE 19

5.1 BUILD_SCREEN 19

5.1.1 Purpose of the Routine 19

5.1.2 Flow Description of the Routine 19

6. SOME SOURCE CODES 21

6.1 For Use in Displaying Screens 21

6.1.1 DISPLY 21

6.1.1.1 Purpose of the Routine 21

6.1.1.2 Flow Description of the Routine 21

6.1.2 TRMNTR 22

6.1.2.1 Purpose of the Routine 22

6.1.2.2 Flow Description of the Routine 22

6.1.3 CHOOSE 22

6.1.3.1 Purpose of the Routine 22

6.1.3.2 Flow Description of the Routine 23

6.1.4 KEYPAD 23

6.1.4.1 Purpose of the Routine 23

6.1.4.2 Flow Description of the Routine 23

6.2 For Use in Data Type Conversions 23

6.2.1 Character String-to-Numerical Value Conversions: RVALUE 23

6.2.1.1 Purpose of the Function 23

6.2.1.2 Flow Description of the Function 23

6.2.2 Numerical-to-Character String Representation Conversions: CVALUE . . 24

6.2.2.1 Purpose of the Function 24

6.2.2.2 Flow Description of the Function 24

6.2.3 Conversions between mmddyy Dates and Integer Date Counts 25

6.2.3.1 Purpose of the Functions 25

6.2.3.2 Flow Description of the Routine 25

6.3 For Use in Constructing Searches 25

6.3.1 SRCBLD 25

6.3.1.1 Purpose of the Routine 25

6.3.1.2 Flow Description of the Routine 25

6.3.2 APSTRG 26

6.3.2.1 Purpose of the Routine 26

6.3.2.2 Flow Description of the Routine 26

7. REFERENCES 27

APPENDIXES A-1

A. CREATE_SCHEMA A-1

B. LOTUS® 1-2-3® PRINT FILES B-1

B.1. Entity-Sorted List B-1

B.2. Screen-Sorted List B-6

C. EXECUTION OF CREATE_SCHEMA C-1

D. SQL INSTRUCTION SET D-1

E. BUILD_SCREEN E-1

F. SCREEN TEMPLATE FOR THE GROSS ALPHA-BETA ANALYSES F-1

G. BUILD_SCREEN OUTPUT G-1

H. ARRAY AND SCALAR DECLARATIONS H-1

I. DISPLY I-1

J. TRMNTR J-1

K. CHOOSE K-1

L. KEYPAD L-1

M. RVALUE M-1

N. CVALUE N-1

O. DAYCNT AND CHRDAT O-1

P. SRCBLD P-1

Q. APSTRG Q-1

TABLES

Table 1. Arrays and Scalars Used in Coding the User Interface. 8

1. INTRODUCTION

1.1 Purpose

This report presents the software tools developed for the construction of the RML/Analytical Radiochemistry sample tracking database and portions of the code for its user interface. Chapters 2 and 3 are taken directly from Reference 1 and are included for completeness and to aid in understanding the tools; keep in mind that notations in these chapters to Reference 2 are to latter portions of this document.

1.2 Overview

Two types of tools have been developed: those for design and constructing the database proper and those for preparing code for the user interface. The starting points for the CREATE_SCHEMA tool are data files prepared using a spreadsheet program; these files name and describe the various data fields necessary for each database table. This tool then constructs an instruction listing for creating the database, as well as FORTRAN declaration statements specifying the entities and attributes that correspond to the various data entry fields and some FORTRAN code for accessing the database. For the tool BUILD_SCREEN, the input is a programmer-prepared ASCII text file displaying how the data screen should look. From this file, BUILD_SCREEN prepares declaration statements for use in preparing the FORTRAN code for the VAX VMS screen management system routine calls used by the user interface.

1.3 Definitions, Acronyms, and Abbreviations

Terminology related to Digital Equipment Corporation's VMS operating system, DECnet communication protocol, DECwindows Motif graphical user interface, FORTRAN compiler, GKS (Graphical Kernel System) plotting routines, Rdb relational database software, and SQL Fortran precompiler, is desirable. Otherwise, this document explicitly defines technical terms as they are presented and avoids unnecessary technical jargon for the sake of the general reader.

1.4 Registered Trademarks and Copyrights

2. DESIGN OF E DATABASE

2.1 Basic Description

This database is a relational, multifile database. It is created and accessed using the Digital Equipment Corporation's Rdb/VMS implementation of the industry standard Structured Query Language (SQL). There is a separate file for each type of analyses and for basic sample information, tracking information, and special instructions. The directory [RDB_DATABASES.SAMPLE_TRACKING] on the user disk of the RML's VAX 6000 mainframe computer contains the database files. The weekly user disk backup includes the contents of these files (see Specifics, section 2.2).

The following subsections present a brief overview of the design of the database. The corresponding subsections under Specifics (section 2.2) present more detail. The presentation assumes the reader has some familiarity with relational database terminology.

2.1.1 Tables and their Fields and Indexes for Sorting

A table (or entity) contains a number of named fields for data storage. The names of the tables are as descriptive as possible while keeping the length of the name within the SQL-standard thirty-character limit.

Each field maps to only one data type (or domain). Appropriate fields serve as primary keys (for uniquely identifying each entry in a table) and foreign keys (for connecting entries between tables). Fields can have their own programmer-specified constraints (in addition to those of the corresponding domain) and comments. For example, null entries can be forbidden for a given field, such as the field that serves as the primary key.

A single field or a collection of fields serves as the primary key index for sorting each table. The name of this index is of the form <storage area name>_PRIMARY_KEY_INDEX. Its size depends on the fields which compose it. The comment for this index usually states the table for which it is the sorting key.

2.1.2 Domains and Attributes

A minimal set of data types, or domains, describes the space requirements for data in the database. Associated with each data type are a default value, constraints on the acceptable values, and a comment, all of which are programmer-specified.

Attributes are the data-containing fields within the database tables. Their names are as descriptive as possible while keeping the length of the name within the SQL-standard thirty-character limit. Each attribute's data type maps to only one domain.

2.1.3 Storage Areas and Snapshot Files

A separate storage area file contains each table and its definitions and constraints. The name of a storage area file is of the form <storage area name>.RDA. Its initial size is determined by the amount of data to be contained in a single entry to the table times an estimate of the number of entries that need to be available in the working database. The storage areas (and snapshot) files are automatically increased

in size as needed by Rdb/VMS. (<storage area name> is a concatenation of the first two letters of each underscore-separated part of the table's name (and so can be computer-generated). For example, <storage area name> for the table SAMPLE_INFORMATION is SAIN. The name of the table's snapshot file is of the form <storage area name>.SNP.)

2.2 Specifics

A listing of the SQL instruction set for creating this multifile database appears in Appendix A. The CREATE_SCHEMA tool generated this instruction set. (The software tools used to design the database are described in Reference 2. The examples presented in that report make use of this database.) The Lotus® 1-2-3® print files,[a] containing the entity- (table-) and screen-sorted attribute description lists, that serve as input to CREATE_SCHEMA appear in Appendix B. A recording of the terminal session from the execution of CREATE_SCHEMA, specifying the programmer's input for this database, appears in Appendix C.

The schema for this database is SAMPLE_TRACKING and so the name of the root file is SAMPLE_TRACKING.RDB. The root file contains the complete set of domain and table definitions.

2.2.1 Names of Tables, Storage Areas, and Snapshot Files

The following are the table names, with the respective storage area names in parentheses:

- SAMPLE_INFORMATION (SAIN)

- for alpha analyses: ALPHA_URANIUM (ALUR), ALPHA_THORIUM (ALTH), ALPHA_PLUTONIUM (ALPL), ALPHA_AM241_SANS_PU239 (ALAMSAPU), ALPHA_AM241_WITH_PU239 (ALAMWIPU), ALPHA_TOTAL_SPECTROMETRIC (ALTOSP), and ALPHA_OTHER (ALOT)

- for gross alpha-beta analyses: GROSS_ALPHA_BETA_AIR_FILTERS (GRALBEAIFI) and GROSS_ALPHA_BETA_AIR_OTHER (GRALBEOT)

- for beta analyses: BETA_STRONTIUM_90 (BEST90), BETA_STRONTIUM_89_AND_90 (BEST89AN90), BETA_TOTAL_STRONTIUM (BETOST), BETA_TRITIUM (BETR), and BETA_OTHER (BEOT)

- for gamma analyses: GAMMA_SCREEN (GASC), GAMMA_FULL_ISOTOPIC (GAFUIS), and GAMMA_OTHER (GAOT)

- SPECIAL_INSTRUCTIONS (SPIN) and

- for possessors: TRACKING (TR).

2.2.2 Names and Descriptions of Domains

The following lists the domains and their respective descriptions. Many of the names fully

---

a. The use of Lotus® 1-2-3® in constructing this database does not constitute an endorsement of Lotus® 1-2-3®. Other spreadsheet programs could also be used to generate these output files.

describe the data type of the domain. For example, read "at most X alphanumerics" as the description when "X_CHARACTERS" is the name of the domain.

- CHARACTER_TIME_HH_MM (four or five numbers-plus-colon long and of the form hh:mm; for recording times-of-day)

- CHARGE_NUMBERS (exactly nine alphanumerics long)

- FIFTY_CHARACTERS

- FLAGS (one character long; for Yes/No data)

- FORTY_CHARACTERS

- IDENTIFICATION_CODES (exactly twelve alphanumerics long and of the numerical form mmddyyhhmmss; for the sample tracking database identification code)

- INTEGER_DATE_COUNT (date count starting at January 1, 1990; for use in "Due Date Check")

- INTEGER_NUMBERS

- NAMES (at most twenty-five alphanumerics long)

- PHONE_NUMBERS (at most twelve alphanumerics long)

- REAL_NUMBERS

- RML_SPECTRAL_ID_CODES (exactly fourteen alphanumerics long and of the form <save area abbreviation of two characters in length>$<spectral identification code of seven alphanumerics in length>, for the location and filename of spectral data; see Reference 1)

- SIXTEEN_CHARACTERS

- SIXTY_CHARACTERS

- TEN_CHARACTERS

- TWENTY_CHARACTERS

- UNITS (at most five alphanumerics long; for abbreviated units of measure such as H for hour, M for minute, G for gram, and ML for milliliter; note the exclusive use of uppercase letters, consistent with the user interface forcing all letters entered to uppercase)

- USER_NAME (at most thirty alphanumerics long; for tracking who last modified an entry) and

- VMS_DATE_TIME (seventeen or eighteen alphanumerics long and of the form "dd-mmm-yy hh:mm:ss"; for noting when a record was last modified).

2.2.3 Backing Up the Database

DCL BACKUP does not work correctly on the database files (the root file and storage area and snapshot files), owing to their internal structure. These files are all marked for no backup using the DCL command SET FILE /NOBACKUP. This way the weekly backup of the user disk on the VAX 6000 mainframe computer does not include these files.

The weekly backup includes the contents of the database storage area files, though. The routine used by the RML operators to perform the backup contains code to do this (see Appendix D). The routine

(i) verifies the integrity of the database

(ii) performs an RMU BACKUP on the database (this a creates file than can be correctly saved and restored using DCL BACKUP

(iii) performs the DCL BACKUP

(iv) purges all but the two most recent versions of the file generated by RMU BACKUP and

(v) marks these files for no backup.

2.2.4 Removal of Inactive Entries from the Database

An inactive entry is one for which all work has been completed, say, at least six months ago. The inactive entries could be moved to a second, nearly identical version of the database. At a minimum, the inactive database would differ from the active database in that it would contain the date the entry was moved. The goal of removing inactive entries is to keep the active database as small as it needs to be; unneeded entries slow down searches and retrievals. The actual moving of entries can be performed automatically or by hand. It could be included as part of the backup procedure (see Backing Up the Database). Inactive entries could be made accessible to the user interface by allowing the user to specify the active or inactive database as the one to use. This capability does not exist with version 1.00 of the sample tracking database.

3. DESIGN OF THE USER INTERFACE

3.1 Basic Description

Except for informational messages, the interface makes relating to the database transparent to the user. After selecting the desired option (Add a Sample Entry, Update a Sample Entry, . . .), the user fills in screen-displayed forms. The interface then constructs and executes the appropriate SQL calls. Screen management system routines are used for constructing the screen-displayed forms, reading data from them, and flagging entry errors and missing-yet-required information.

Owing to the case-sensitive nature of SQL searches, all alphabetic input to the interface program is forced to uppercase. Numerical input to data fields can be in the form of integer, real, or exponential numbers; the interface chooses the most complete and concise representation for displaying numerical data retrieved from the database.

3.2 Specifics

The software tools described in Reference 2 were used in preparing portions of code, such as declaration statements. They were also used to generate most of the dynamic SQL calls and related FORTRAN declaration and data-type conversion statements for the user interface. The examples presented in that report make use of this database.

The following subsections provide more detail on how the user interface stores data, converts data types, and performs error checking on entered data. It also describes the types of SQL calls the user interface uses and how, using SQL calls, it communicates with the database in performing insertions (for adding a sample entry), cursor declarations and fetches (for searching for a sample entry and due date checks), and modifications (for updating a sample entry). The routines and functions mentioned in the following subsections are described in more retail in Reference 2.

3.2.1 Data Storage Variables and Arrays

The array and scalar declaration statements (and some source code) related to the screen management routines are contained in the include file FLDDEF.INC (see Appendix E). These statements were generated using the tools BUILD_SCREEN and CREATE_SCHEMA; the screen templates submitted to BUILD_SCREEN are presented in Appendix F (see Reference 2).

3.2.1.1 Storage of Data. For ease of use with screen management routines, all information for a given screen is saved in a character array (composed of eighty-character-long elements) assigned to that screen. Since SQL calls cannot make use of arrays (and variable names over six characters long are discouraged in Rdb/VMS), the arrays are equivalenced to a set of scalar variables that are of the required length for the appropriate domains. A character array's name is of the form fidat<first letter of the screen's name>; for example, for the Main screen the array's name is fidatM.

BUILD_SCREEN creates a set of scalar variables using a generic naming scheme that only identifies the screen. For example, scalar variables for the main screen's data fields are named fldM<n>, where <n> is the sequential number of the field within the screen. For clarity, the names of the scalar variables were changed to include some relation to the appropriate attribute.

The new names of the scalar variables begin with f<first letter of the screen's name>. The

remaining four letters, at most, of each name are constructed in a fashion similar to how storage area names are constructed (see Storage Areas and Snapshot Files, section 2.1.3) but using only the first letter of each underscore-separated part of the respective attribute. For example, for the main screen's sample tracking identification code (attribute name SAMPLE_TRACKING_ID) the scalar variable is named fMSTI (Main screen, then SAMPLE, then TRACKING, then ID). (Some hand modifications were required to eliminate duplicate names).

3.2.1.2 Conversion of Character Data to Numerical Data. If the database requires a number for a given attribute, the appropriate translation of character-to-integer or character-to-real is performed. (The result is stored in a scalar variable the name of which begins with an "i" instead of an "f" for integer data and with an "r" instead of an "f" for real data). The reverse translation is performed for numerical data retrieved from the database and to be displayed via screen management routines. These conversions are handled by the FORTRAN functions RVALUE (for character string-to-real number conversions) and CVALUE (for real number-to-character string conversions). The codes for these functions, again, are presented in Reference 2.

The arrays containing labels for the fields have names of the form field<first letter of the screen's name>. The arrays containing the names of the entities and attribute corresponding to the screen fields have names of the form atrbt<first letter of the screen's name>. There are no scalar equivalents for these arrays. Other arrays and scalars used in the screen management calls are similarly constructed; Table 1 lists all the arrays associated with any screen. (Note that from here on and in the table "<first letter of the screen's name>" is abbreviated "<>.")

Some of these arrays will be discussed in more detail later, in the following subsection on error checking for example. The implementation of these arrays, and the coding for the error checking, is found in the FORTRAN routine DISPLY (see Reference 2).

3.2.1.3 Error Checking. The user interface checks for three types of errors. First, is the data entered short enough to fit into the appropriate field? Each data field has a maximum number of alphanumerics that can be entered into it, mxnoc<>. Second, is the data required? If the field is rendered in bold print, an entry must be made into the data field before the user can leave that screen. Also, an entry can be required in a field because an entry has been made in a related field: if an entry appears in the "Date Completed" field, for example, an entry would be required in the "Results report citation" field. Third, is the data entered of the type specified by the fityp<> array? While data entered into a field is read into a FORTRAN CHARACTER variable, a real number may be needed for the corresponding attribute in the database. The following data types, as implemented in the FORTRAN routine DISPLY, are presently available.

3.2.1.3.1 "a" for Anything - Any combination of alphanumerics, spaces included and up to the specified maximum length, is allowed.

3.2.1.3.2 "d" for Date - A date in the integer form mmddyy is allowed. The entry will be converted into an integer date count (that is, how many days have elapsed between the specified date and December 31, 1989) for storage in the database. Integer date counts make due date checks easier to perform, for instance. Conversions are accomplished using the FORTRAN functions DAYCNT (character mmddyy-to-integer date count) and CHRDAT (integer date count-to-character mmddyy).

An entry of this type must be composed of exactly six integers (leading and trailing zeros must be included). The first two digits, mm, represent the month and must fall in the range 01 to 12 (inclusive). The second two digits, dd, represent the day of the month. The last two digits, yy, represent the

Table 1. Arrays and Scalars Used in Coding the User Interface.

Array Name	Description	Scalar Equivalent?	Letter for Equivalent
field<>	contents of the description field	N	-
fidat<>	contains error-free data (or is null) from the data field	Y	f
atrbt<>	table (entity) and attribute names corresponding to the screen field	N	-
varbl<>	contains the names of scalar variables equivalenced to fidat<> and to the respective if-null indicator	N	-
fityp<>	allowed data type for the data field ("a" for alphanumeric, e.g.)	N	-
fdlen<>	length of the description field	N	-
fdrow<>	row number for display of the description and data fields	N	-
fdcol<>	starting column number for display of the description field	N	-
ficol<>	starting column number for display of the data field	N	-
mxnoc<>	maximum number of alphanumerics allowed in the data field (length of data field)	Y	l (for length)
fdrnd<>	rendition of the description field (normal if optional, bold if required, reverse video if fixed)	N	-
firnd<>	rendition of the data field	N	-
ifnul<>	if-null indicator: -1 if the data field is empty (null), 0 otherwise	Y	n (for null)
ifrqd<>	entry into the current data field required if data field number ifrqd<> is not empty	N	-
idito<>	"ditto" transfers contents of data field number idito<> into the current data field	N	-
ifmod<>	if-modified indicator: true if the data field's value has been changed (during Update), false otherwise	Y	m (for modified)

year and are interpreted as follows: if yy is between 90 and 99, the year is 19yy; if yy is between 00 and

79, the year is 20yy. Entries between 80 and 89 are not allowed as they should be interpreted as 20yy, yet the resulting integer date count would be too large to fit into a FORTRAN INTEGER*4 variable and hence into the respective database attribute.

3.2.1.3.3 "l" for Length - Here the entry must be exactly mxnoc<> alphanumerics long. There are no restriction as to the alphanumerics used. One instance in which "l" is used is the charge number to be billed for performing the analyses. All charge numbers are exactly nine alphanumerics long, so a shorter or longer entry is invalid.

3.2.1.3.4 "t" for Time - The format of the entry allowed here is either "h:mm" or "hh:mm." h or hh represents the hour as an integer. A leading zero is not necessary because the colon can be used to delineate the hour portion of the entry from the minute portion. mm represents the number of minutes.

3.2.1.3.5 "u" for Units of Time - In instances such as the units for the duration of the spectral acquisition, an entry of either H for hours or M for minutes is allowed. "u" covers only units of time (duration); it does not cover sample size units, for example.

3.2.1.3.6 "?" for Questions needing a Yes-or-No Answer - The only allowed entries here are "Y" for yes and "N" for no. This data type is used for flags such as whether a certain type of analysis is to be performed.

3.2.2 Communication between the User Interface and the Database

SQL calls are used to transfer data between the user interface and the database. These calls are either hard-coded into the user interface (precompiled SQL) or are constructed by it (dynamic SQL).

3.2.2.1 Forms of SQL Used. Both precompiled SQL and dynamic SQL calls are used. Precompiled SQL calls are completely known by the user interface before they are executed. Such calls include inserting all attributes of an entry into the database. An EXEC SQL preface marks these calls as precompiled SQL statements rather than as statements in the host program's language.

Dynamic SQL calls can only be composed during execution of the user interface. Instances of these include requesting the tables needed for the current transaction. The instruction set for a dynamic SQL call is constructed as a FORTRAN CHARACTER variable and is executed from the main routine using EXEC SQL EXECUTE IMMEDIATE. To minimize the length of the character string, a FORTRAN routine APSTRG (APpend character STRinG) fills the character variable with each additional portion of the SQL call passed to it.

Modular SQL, in which calls are made from the main routine to SQL routines constructed in a single and separate file, is not used. The main routine of the user interface is larger than might seem prudent for ease of code development because of the Rdb/VMS requirement that all precompiled SQL statements reside in the same routine.

3.2.2.2 Commencing a Database Transaction. Communications with the database occur within transactions. The transaction definition (SET TRANSACTION) specifies the tables of the database the user needs and the type of access required. For example, for adding an entry to the database, the sample coordinator needs exclusive write access to all of the tables for the requested analyses. No one else should be able to access these tables in any fashion until the addition is completed. The construction of the SET TRANSACTION call depends on the application (for example, see Add a Sample Entry,

section 3.2.2.3).

At present, all SET TRANSACTION calls will wait at most sixty seconds for the requested tables to become available. To minimize conflicts (lock conditions resulting from two users wanting the same table at the same time and in incompatible fashions), database transactions last only as long as necessary. Any database transaction starts only as soon as all the requisite information is available. Again using the example of adding an entry, the database transaction does not begin until all of the appropriate screens are filled by the user. And, accordingly, each transaction is closed - either committed, that is, saved, or rolled back, that is, undone - upon completion.

3.2.2.3 Add a Sample Entry. The addition (insertion) of a sample entry to the database occurs via the SQL INSERT command. Here the main routine constructs the SET TRANSACTION command (using the FORTRAN routine APSTRG mentioned above in Forms of SQL Used), and then executes it as a dynamic SQL call (EXECUTE IMMEDIATE). Again, dynamic SQL is used since all of the tables that will be needed are generally not known until it is time for the addition to occur. All tables needed are included in a single SET TRANSACTION command because violations of constraints are better managed within a single transaction than over a series of transactions. A constraint, for example, may require that an entry for sample identification code exists in the sample information table before an entry for that same identification code can be made into an analysis table. A typical sequence of calls to APSTRG, followed by execution of the SET TRANSACTION call, would be

```
      call apstrg(comnd1,lcmnd1,('set transaction read write wait 60'//
     . ' reserving SAMPLE_INFORMATION for exclusive write,'),.true.)
      if(ifgros(1))then
         if(ifgros(2))call apstrg(comnd1,lcmnd1,
     .    ' GROSS_ALPHA_BETA_AIR_FILTERS for exclusive write,',.false.)

                                  . . .

         endif

                                  . . .

      call apstrg(comnd1,lcmnd1,
     . (' SPECIAL_INSTRUCTIONS for exclusive write,'//
     . ' TRACKING for exclusive write'),.false.)

                                  . . .

      exec sql execute immediate :comnd1

                                  . . . .
```

The next SQL calls are precompiled SQL calls and insert the data provided by the sample coordinator into the database:

```
      exec sql insert into SAMPLE_INFORMATION values(
     .    :fMSTI:nMSTI,:iMDSE:nMDSE,:iMDAWC:nMDAWC,:fMCSI:nMCSI,
     .    :iMRNBD:nMRNBD,:fMCPN:nMCPN,:fMWCN:nMWCN,:fMCSN:nMCSN,
     .    :fMST:nMST,:rMSS:nMSS,:fMSSU:nMSSU,:iMSCD:nMSCD,
     .    :fMSCT:nMSCT,:fMASI:nMASI,:fMSHS:nMSHS,:rMSHSA:nMSHSA,
     .    :fMSSN:nMSSN,:fMSSPN:nMSSPN,:fMSSA:nMSSA,:fMTCN:nMTCN,
     .    :fMTCPN:nMTCPN,:fMTCA:nMTCA,:fMSRTN:nMSRTN,:fMSRTP:nMSRTP,
     .    :fMSRTA:nMSRTA,:fMSPBN:nMSPBN,:fMSPBP:nMSPBP,:fMSPBA:nMSPBA,
     .    :fMNGAB:nMNGAB,:fMNA:nMNA,:fMNB:nMNB,:fMNG:nMNG,
```

```
.    :nxinst:ifnnxi,:nxpssr:ifnnxp,:fGNGS:nGNGS,:fGNGFI:nGNGFI,
.    :fGNGO:nGNGO,:fCNGAF:nCNGAF,:fCNGAO:nCNGAO,:fBNBSr:nBNBSr,
.    :fBNBSC:nBNBSC,:fBNBST:nBNBST,:fBNBT:nBNBT,:fBNBO:nBNBO,
.    :fANAU:nANAU,:fANATh:nANATh,:fANAPu:nANAPu,:fANASP:nANASP,
.    :fANAWP:nANAWP,:fANATS:nANATS,:fANAO:nANAO,USER,:today:notnul,
.    :fMCOCN:nMCOCN)
```

The values are passed in the form :<scalar equivalent of data field>:<scalar equivalent of if-null indicator>. The colon preceding each variable name tells the SQL precompiler that what follows is a variable in the user interface's native language, not an SQL variable or key word.

What about fields into which no entry was made? Recall that there is an if-null indicator array. If a field is left blank, that field's if-null indicator is set (to a value of -1). When this if-null indicator is passed to the database (with a value of -1), the attribute in the database entry is marked NULL. (The if-null indicator could be used to specify which attributes are to be passed to a table. Dynamic SQL would be used in place of precompiled SQL in this case. Pointers to variables are required in place of variables for dynamic SQL, though.)

3.2.2.4 Search for a Sample Entry. This is another instance where dynamic SQL calls could be used; this time they are, in conjunction with the APSTRG routine and the atrbt<> and fidat<> arrays. To perform a search, the user fills in all the fields necessary to describe the entry or entries desired. The if-null indicator is used to indicate the fields to be included in the search command. The user is to leave blank those fields for which no information is known or for which the information may vary among the desired entries.

Recall that the atrbt<> arrays contain the name of the table (entity) and attribute that correspond to a given screen field. The search command is constructed using the atrbt<> arrays and the values contained in the fidat<> arrays. If a data field contains an entry (and so its if-null indicator is not set), the appropriate elements from atrbt<> and the entry are used in constructing the dynamic SQL search call via APSTRG. These portions of the search command are constructed in the FORTRAN routine SRCBLD (SEARCH BUILD).

Searches are conducted using cursors and fetches from cursors. A cursor can be thought of as a virtual table containing only the specified entries (EXEC SQL DECLARE . . . CURSOR for <statement name>; EXEC SQL PREPARE <statement name> from <search command>). Data from a cursor can be retrieved into native language variables (EXEC SQL FETCH . . . INTO . . .). A typical portion of the main routine's code for constructing the search command and executing the search and retrieval is

```
 call apstrg(comnd1,lcmnd1,
. ('select distinct SAMPLE_INFORMATION.SAMPLE_TRACKING_ID,'//
. ' SAMPLE_INFORMATION.CUSTOMER_SAMPLE_ID,'//
. ' SAMPLE_INFORMATION.CUSTOMER_SAMPLE_NAME'//
. ' from SAMPLE_INFORMATION'),.true.)
 call apstrg(comnd2,lcmnd2,' where',.true.)

                          . . .

 call srcbld(comnd1,lcmnd1,comnd2,lcmnd2,numfdM,fidatM,mxnocM,
. fitypM,atrbtM)
 if(ifgama(1))call srcbld(comnd1,lcmnd1,comnd2,lcmnd2,numfdG,
. fidatG,mxnocG,fitypG,atrbtG)

                          . . .
```

```
      call apstrg(comnd1,lcmnd1,(comnd2(1:lcmnd2)//
     . ' order by SAMPLE_INFORMATION.SAMPLE_TRACKING_ID asc'),.false.)

                                    . . .

      exec sql declare SEARCH_cursor cursor for SEARCH_statement
      exec sql prepare SEARCH_statement from comnd1
      exec sql open SEARCH_cursor
      exec sql whenever not found goto 704

                                    . . .

  702 . . .
      exec sql fetch SEARCH_cursor into :smplid,:custid,:csname

                                    . . .

      goto 702
  704 . . .
      exec sql close SEARCH_cursor

                                    . . . .
```

Note that two character strings are input to the routine APSTRG. For each non-null data field, a call to SRCBLD transfers the name of the respective entity alone to the first string (if it does not already appear there) and, to the second string, the entity (table) name, the attribute name, and the value contained in the data field. The second string is appended to the first string prior to the search. Finally, the search is performed as the cursor is declared (defined), opened (much as a file is opened), and read from (via FETCH) until the last entry is found (watched for by WHENEVER NOT FOUND . . .). The data retrieved can be displayed as it is found and stored for future reference. After all the data is retrieved from the cursor, the cursor is closed.

3.2.2.5 Due Date Checks. Due date checks are a form of search where the same, specific attributes in the various tables are always interrogated, namely the date-completed fields. The sample coordinator states how many days ahead should be checked for analyses and reports which will be due. The user interface calculates the integer date count for the current day, and adds to it the number of days ahead to be checked. A search is then preformed for all entries having due dates the integer date count for which is less than or equal to the above sum (IDUCNT) and for which the specified work has not been completed (that is, for which the DATE_ALL_WORK_COMPLETED field is null). The coding for one such search is

```
      exec sql declare BEST90_due_cursor cursor for
     .   select
     .      BETA_STRONTIUM_90.SAMPLE_TRACKING_ID,
     .      SAMPLE_INFORMATION.CUSTOMER_SAMPLE_ID,
     .      SAMPLE_INFORMATION.CUSTOMER_SAMPLE_NAME,
     .      BETA_STRONTIUM_90.RESULTS_NEEDED_BY_DATE
     .   from BETA_STRONTIUM_90, SAMPLE_INFORMATION
     .   where BETA_STRONTIUM_90.RESULTS_NEEDED_BY_DATE<=:iducnt and
     .      BETA_STRONTIUM_90.DATE_ALL_WORK_COMPLETED is NULL and
     .      BETA_STRONTIUM_90.SAMPLE_TRACKING_ID=
     .       SAMPLE_INFORMATION.SAMPLE_TRACKING_ID
     .   order by BETA_STRONTIUM_90.SAMPLE_TRACKING_ID asc

                                    . . .
```

```
    exec sql open BEST90_due_cursor

                                . . .

    exec sql whenever not found goto 836
834 exec sql fetch BEST90_due_cursor into :smplid,:custid,:csname,
  . :duedat

                                . . .

    goto 834
836 . . .
    exec sql close BEST90_due_cursor

                                . . . .
```

3.2.2.6 Update a Sample Entry. This final mode of communication uses the techniques described above. First, data for the specified entry is fetched from the database via a cursor:

```
        exec sql declare ALUR_cursor cursor for
.          select
.             ALPHA_URANIUM.SAMPLE_TRACKING_ID,
.             SAMPLE_INFORMATION.CUSTOMER_SAMPLE_NAME,
.             ALPHA_URANIUM.RESULTS_NEEDED_BY_DATE,
.             ALPHA_URANIUM.DATE_ALL_WORK_COMPLETED,
.             ALPHA_URANIUM.RESULTS_REPORT_CITATION,
.             ALPHA_URANIUM.ANY_SPECIAL_INSTRUCTIONS
.          from ALPHA_URANIUM, SAMPLE_INFORMATION
.          where ALPHA_URANIUM.SAMPLE_TRACKING_ID=:smplid and
.             ALPHA_URANIUM.SAMPLE_TRACKING_ID=
.              SAMPLE_INFORMATION.SAMPLE_TRACKING_ID
.          order by ALPHA_URANIUM.SAMPLE_TRACKING_ID desc

                                . . .

        exec sql open ALUR_cursor

                                . . .

        exec sql fetch ALUR_cursor into
.          :fASTI:nASTI,:fACSN:nACSN,:iARNB1:nARNB1,:iADAW1:nADAW1,
.          :fARRC1:nARRC1,:fAASI1:nAASI1

                                . . .

        exec sql close ALUR_cursor

                                . . . .
```

The retrieved data is then displayed for the sample coordinator to update. Once the updating of the screens is completed, the entire database entry is updated (modified) if any changes have been made within the screens.

The if-modified indicator plays a role different from that the if-null indicator plays in searching for a sample entry. If any of the if-modified indicators for a table's fields are true, the entire entry is updated if it previously existed in the database. If the entry is new to the database, the entry is added to the database:

```
i   falph(2).and.(mARNB1.or.mADAW1.or.mARRC1.or.mAASI1))then

                                . . .

    if(ifalpO(2))then    !Use update to change the entry.
         exec sql update ALPHA_URANIUM set
.                 RESULTS_NEEDED_BY_DATE=:iARNB1:nARNB1,
.                 DATE_ALL_WORK_COMPLETED=:iADAW1:nADAW1,
.                 RESULTS_REPORT_CITATION=:fARRC1:nARRC1,
.                 ANY_SPECIAL_INSTRUCTIONS=:fAASI1:nAASI1,
.                 USER_CREATING_OR_MODIFYING=USER,
.                 DATE_CREATED_OR_MODIFIED=:today:notnul
.             where SAMPLE_TRACKING_ID=:smplid
      else    !Use insert since it was not there before.
         exec sql insert into ALPHA_URANIUM values(
.           :fASTI:nASTI,:iARNB1:nARNB1,:iADAW1:nADAW1,
.           :fARRC1:nARRC1,:fAASI1:nAASI1,USER,:today:notnul)
      endif
    endif

                                . . . .
```

4. TOOLS USED IN DESIGNING THE DATABASE

4.1 CREATE_SCHEMA

4.1.1 Purpose of the Routine

CREATE_SCHEMA generates the SQL commands for creating the database and its associated definitions (the schema). It prepares FORTRAN declaration statements for variables that allow a data entry field to be related to the correct entity and attribute. It also generates the precompiled SQL code (for FORTRAN) necessary to access the database in a number of ways, such as UPDATE and FETCH, INSERT and CURSOR.

CREATE_SCHEMA asks questions of the programmer as it runs, and so it should be executed interactively. To do so, type

@CREATE_SCHEMA

at the DCL prompt. There are no command line parameters to be specified. Appendix C records the interactive execution of CREATE_SCHEMA used to create the sample tracking database's schema.

4.1.2 Flow Description of the Routine

The DCL code for CREATE_SCHEMA is presented in Appendix A and is summarized here. There are no declaration statements in DCL; numbers can be stored only as integers, and character strings are distinguished from numbers by enclosing the value in quotation marks. After initializing some counters, CREATE_SCHEMA opens the two Lotus® 1-2-3® print files, ENTITY.PRN and SCREEN.PRN; these files are presented in Appendix B and described in detail below under Input Files, section 4.1.3.

After obtaining the name of the schema to be created from the programmer, CREATE_SCHEMA opens a number of temporary output files and writes appropriate comments to them. These files will later be concatenated and rearranged to yield the final output files, described below under Output Files, section 4.1.4.

4.1.2.1 Creating the Domains. The first set of SQL commands to be generated create the various domains (or data types) for use within the database. The statements to create two domains, USER_NAME and VMS_DATE_TIME, are automatically written to file by CREATE_SCHEMA. The descriptions of the rest of the domains are provided interactively by the programmer. CREATE_SCHEMA reads through the ENTITY.PRN file until it finds the alphabetic list of unique domains. It then reads one of these at a time, and presents each to the programmer for definition. The programmer can assign the domain to one of four generic data types: character, date, integer, or real. Default values and comments can also be assigned. CREATE_SCHEMA keeps track of the names of the domains and their respective sizes in bytes for later use. From all of this information, CREATE_SCHEMA writes to file the appropriate SQL statements.

4.1.2.2 Creating the Tables. The second set of SQL commands that CREATE_SCHEMA generates create the various entities or tables described in ENTITY.PRN. Starting over at the top of ENTITY.PRN, CREATE_SCHEMA reads the attribute listings for a given entity one at a time. For each entity (or table), an abbreviated name is constructed for later use. The abbreviation is constructed as described above in Storage Areas and Snapshot Files, section 2.1.3. For each new table encountered,

CREATE_SCHEMA writes the first part of a number of precompiled SQL calls (cursor declarations, fetches from the database, inserts into the database, . . .).

Each attribute listing specifies the name of the attribute, what type of key, if any, the attribute is, and the domain to which the attribute is to be associated. From the domain, for instance, CREATE_SCHEMA determines how many more bytes of storage space per record will be needed as each attribute is added. (Remember that CREATE_SCHEMA recorded the number of bytes needed for each domain earlier.) If the attribute is labeled as the primary key for the table, CREATE_SCHEMA records this; if multiple attributes make up a table's primary key, CREATE_SCHEMA properly constructs the necessary key description, including the total number of bytes needed for the primary key.

Special handling is needed for those attributes which will contain integer or real numbers in the database. The screen management routines that present the data for the user to see can only make use of characters strings, and so character representations need to be created for numerical data. In these cases, CREATE_SCHEMA generates the appropriate integer or real FORTRAN declaration statements and the code or function calls necessary to convert between numeric and character representations.

Once the end of a table (entity) is reached, CREATE_SCHEMA calculates the size of the storage area for the table and its snapshot file. FINISHED_WITH_TABLE is the subroutine that handles these calculations and the generation of the appropriate SQL commands to create the table. The subroutine also completes the precompiled SQL calls that have been growing with each new attribute.

4.1.2.3 Creating the Schema. After all of the statements for creating all of the tables have been generated, CREATE_SCHEMA now has enough information to write the first part of the statements that actually create the schema. The main piece of needed information that was unknown until all of the tables were dealt with is the size of the largest table. Associated with the schema is a buffer, the size of which depends on the size of the largest table.

In should be clear at this point why a number of temporary output files will be put together to arrive at the final output files. Consider just the statement for creating the schema. Rather than forcing a second or even third pass through the print files, CREATE_SCHEMA writes to a file called CREATE_SCHEMA_SECOND.SQL the statements for creating the tables as the tables are found. The first part of the statements cannot be written until later, and so they are written to another file, CREATE_SCHEMA_FIRST.SQL. The final output file, CREATE_<name of the schema>.SQL, is created by concatenating CREATE_SCHEMA_SECOND.SQL to the end of CREATE_SCHEMA_FIRST.SQL.

A similar approach is taken in dealing with the precompiled SQL statements and the associated representation conversions, and is then taken a step further. In the instance of the precompiled SQL statements that handle the cursor declarations, the contents of one of the files needs to be reordered. Fragments of the precompiled SQL statement are intermingled with the internal write statements that are used to effect the number-to-character representation conversions, and so the file is read through and the appropriate pieces culled out to new temporary files. These new temporary files are then used in the final concatenation in place of their parent file.

4.1.2.4 Associating Attributes with Data Fields. The last portion of CREATE_SCHEMA makes use of the SCREEN.PRN Lotus® 1-2-3® print file. It generates the FORTRAN declaration statements that associate the data fields within a screen with the appropriate attributes. It also generates the declaration statements that associate the data fields with the scalar variables that contain the value entered into that field (see Storage of Data, section 3.2.1.1).

Each attribute listing includes a screen (identified using the one-letter abbreviations used by BUILD_SCREEN, section 5.1) and the number of the data field within that screen with which the attribute is to be associated. This information, along with the domain, allows CREATE_SCHEMA to generate the proper declaration statements.

4.1.3 Input Files

4.1.3.1 ENTITY.PRN. Appendix B.1 contains a listing of the ENTITY.PRN file used to create the schema for the sample tracking database. Attribute lists for different entities are separated from each other by a 30-character-long dashed line. Because duplicate "keys" (see below) have not been removed from the listings, a change in entity name does not necessarily signal the beginning of a new entity.

For the majority of the file there are six columns that contain information about each attribute. The first column, labeled "Entity," contains the name of the entity (or table) in which the attribute can be found. The second column, labeled "Attribute," contains the name of the attribute. (Note that there is actually a one-character-wide column preceding "Attribute." This column contains only a period and was added for readability.)

The third column is labeled "Key" and in many instances is left blank. The only notations made here have to do with whether the attribute is the table's primary key (and so is marked with a p) or part of the table's primary key (marked p<n>, n an integer). A duplicate "key" is marked with a d; this indicates that this attribute is actually from another table and should not be included in size calculations for the current table. The fourth column, labeled "Screens," contains the one-letter abbreviation for the screen in which the attribute appears. The fifth column, labeled "#," gives the sequential number of the data field with which the attribute is associated.

The sixth column contains the domain, the data type for the attribute. At the beginning of the file there are actually two columns labeled "Domain." The second one contains the alphabetically ordered listing of all of the unique domains. The sorting is performed within Lotus® 1-2-3®.

4.1.3.2 SCREEN.PRN. Appendix B.2 contains a listing of the SCREEN.PRN file that is related to the above ENTITY.PRN file. Here the attributes are sorted on the basis of the screen with which they are associated, in order of increasing data field number. A change in the contents of the column "Screen" signals the start of a new screen. The columns are identical to those in ENTITY.PRN, except for the absence of the second "Domain" column; CREATE_SCHEMA, having read ENTITY.PRN before reading SCREEN.PRN, already knows the unique set of domains it will encounter.

The creation of this screen-sorted listing logically precedes the creation of the above entity-sorted listing. Within Lotus® 1-2-3®, the screen-sorted listing can be sorted by entity and attribute to give the entity-sorted listing.

4.1.4 Output Files

One output file from the CREATE_SCHEMA tool, CREATE_<name of the schema>.SQL, contains the SQL commands necessary to construct the specified database. A listing of the instruction set used to create the sample tracking database is found in Appendix D.

The remaining output files contain precompiled SQL statements to be included in the FORTRAN code for the user interface. Examples of code similar to that generated can be found above in Design of the User Interface, section 3. The main difference is in the naming of the scalar variables; the default

names have been replaced with abbreviations that reflect the name of the associated attribute (see Storage of Data, section 3.2.1.1).

5. TOOLS USED IN DESIGNING THE USER INTERFACE

5.1 BUILD_SCREEN

5.1.1 Purpose of the Routine

The data screens could be constructed using a series of VAX VMS screen management system routine calls, each of which contains the complete description for an individual data field. Or the calls could be executed in a loop, the descriptions for the fields coming from the appropriate arrays. Preferring to keep data separate from code within the user interface program, the latter method is used here. BUILD_SCREEN constructs the needed arrays, along with the associated declaration statements. BUILD_SCREEN is run once for each data screen, and a separate output file is created for each data screen. The output files for all of the data screens can then be concatenated together into a single field definitions file (FLDDEF.INC) that is INCLUDEd in the source code for the user interface.

BUILD_SCREEN does not ask any questions of the programmer, and so it can be executed as a spawned process. To execute it as a spawned process, enter, for example,

SPAWN/NOWAIT @BUILD_SCREEN MAIN M Y

at the DCL prompt (the user-specified parameters that follow @BUILD_SCREEN are described below under Flow Description of the Routine, section 5.1.2). To execute BUILD_SCREEN interactively, simply enter

@BUILD_SCREEN MAIN M Y

at the DCL prompt.

5.1.2 Flow Description of the Routine

The DCL code for BUILD_SCREEN appears in Appendix E. Again, there are no declaration statements in DCL; numbers can be stored only as integers, and character strings are distinguished from numbers by enclosing the value in quotation marks.

Up to three user-specified parameters can be included in the command line when invoking BUILD_SCREEN. The first, p1, identifies the input file. An input file should have a name of the form <p1>_SCREEN.DAT; <p1> could be MAIN for the main data screen, for instance. The second, p2, is a one-letter abbreviation for the data screen - M for main. This letter is used in constructing the names of the arrays to ensure that unique names are created for any data screen. The third parameter, p3, is optional, and tells BUILD_SCREEN whether to include some declarations needed for the user interface but that are not directly related to any screen.

First, the input file is opened. This file contains a picture of how the data screen should look, minus the field renditions (normal text, bold, . . .). The programmer constructs the input file using a full screen editor; an example is presented in Appendix F. The numbers across the top and down the left-hand side are part of the file and are included for reference. There are 20 lines and 78 columns available for data fields; the remaining two columns and two of the remaining three lines are used to draw a box around the screen. The twenty-first line is used to display some reminders about common keystrokes.

Six temporary output files are then opened. FIELDS.DAT will contain the FORTRAN code for

filling the arrays that will describe the data fields; declaration statements are not used to do this so that all of the information for a given field can be grouped together in one place. To SCRDEC.DAT will be written the FORTRAN declaration statements for the arrays, with sizes appropriate for the number of fields in the data screen. A short bit of code for setting initial default values for the field renditions will be written to DEFFLD.DAT. The last three files, CHARAC.DAT, LOGICA.DAT, and EQUIVA.DAT, will contain the scalar versions of the contents of the data field description, the logical-like if-null-indicator arrays, and the equivalence statements that connect the scalar variables to the appropriate array elements, respectively. Appropriate comments are written to these last three files immediately after they are opened.

After initializing counters for the line number currently being scanned for fields (LINMBR) and for the number of data fields found so far (NUMFLD), the input file is read one line at a time. The first line of the input file contains column numbers and so is not scanned for data fields. As each subsequent line is read, LINMBR is incremented by one. If a blank line is encountered in the input file, it is treated as a numbered line but not scanned for fields.

Each non-null line is scanned, one character at a time, for fields. The beginning of a field is signaled by an alphanumeric character (FIRSTC) must appear in column one, or in a later column (at most up to column 63 - MAXNFC) but preceded by a blank. The field's alphanumeric descriptor is expected to precede a series of underscores; the underscores indicate the maximum-allowed length for the data to be entered.

The descriptor, FIELD, is all of the characters from the first one detected up to but not including the first underscore, minus any trailing blanks. The column in which a new data field description begins is saved in FDSCOL. From FDSCOL, the line is scanned until an underscore is encountered; the column in which the first underscore appears is saved in FINCOL. The length of the data entry field, MAXNOC, is the number of underscores found. With a completely described data field in hand, the number of data fields found, NUMFLD, is incremented by one, and the appropriate field-specific statements are written to the output files FIELDS.DAT, CHARAC.DAT, EQUIVA.DAT, and LOGICA.DAT.

Starting at the column after the last underscore, BUILD_SCREEN looks for another data field in the same line. If none are found, the next line is read and scanning starts again at column one. Once all of the lines have been scanned, the FORTRAN declaration statements for the various arrays can be written, along with the bit of code to initialize the field renditions. The six temporary output files are then concatenated into a single file, <p1>_SCREEN.FOR. Appendix G contains the single output file that results from the processing of the input file found in Appendix F. The temporary output files are deleted before the routine exits to the DCL prompt. The only hand-editing that needs to be performed on the output file involves specifying the field renditions to be used for each field.

Once BUILD_SCREEN is executed for all of the data screens, the individual output files can be concatenated together and rearranged into a single field definitions file, call it FLDDEF.INC. Appendix H contains the FLDDEF.INC file for the user interface to the sample tracking database. This file is INCLUDEd in the source code for the user interface. Keeping these definitions in a separate include file reduces the size of the file containing the rest of the source code for the user interface. For the sample tracking database, FLDDEF.INC takes up more lines than the rest of the source code.

6. SOME SOURCE CODES WITH BROAD APPLICATION IN DATABASE USER INTERFACES

6.1 For Use in Displaying Screens

6.1.1 DISPLY

6.1.1.1 Purpose of the Routine. The source code for the FORTRAN subroutine DISPLY is presented in Appendix I. This routine makes use of VAX VMS screen management (SMG) system routines to display the various screens of the user interface on the user's monitor. In particular, these system routines allow the user interface to place text in the precise location desired on the user's monitor. Further, the text can be rendered in a number of ways: normal, bold, reverse video, and blinking.

The routine handles the initial data entry functions. It also confirms that the appropriate type of data has been entered into each field. It will flag those fields that are in error and keep the user from moving on to another screen until the errors are corrected.

6.1.1.2 Flow Description of the Routine. The calling routine passes to DISPLY all of the information needed to describe the fields to be displayed on the user's monitor. This includes what text is to be printed, the rendition in which it is to be printed, the type of data to be entered into each field, and any special relationships that exist between data fields. Declaration statements for the variables to contain this information follow, along with the necessary screen management system routine include files.

The first step is to assure that the virtual keyboard (where the user's input is coming from) is established correctly. This is done by first deleting any existing virtual keyboard (SMG$DELETE_VIRTUAL_KEYBOARD), and then creating a new one (SMG$CREATE_VIRTUAL_KEYBOARD). Once this is done the various fields can be displayed.

Calls to SMG$PUT_CHARS place the fields in their proper locations. The renditions used are field-dependent: if an entry in this field is required, render the field in bold; if the entry is already fixed and cannot be changed, render the field in reverse video. (These renditions are specified by hand-editing the output file from BUILD_SCREEN. See Flow Description of the Routine, section 5.1.2.) All data input fields are rendered as underlined (FINPUT); this makes the screen look like a fill-in-the-blank form. Additional calls to SMG$PUT_CHARS add comments and key stroke reminders.

Beginning at the field IBEGIN, DISPLY waits for user input. This may be directives for moving the cursor (such as the arrow keys), or data entered into a data field. All input is accepted via SMG$READ_STRING and then interpreted appropriately. DISPLY directly handles the special function keys PF1 (for aborting execution), PF2 or Help (for displaying the special function key mapping via the KEYPAD routine - see Keypad, section 6.1.4), and "Insert Here" to ditto data from a connected field into the current field.

Other special function keys are acted upon after the contents of the current data field are checked against the requirements for the kind of information needed in that field. Dates should be given as mmddyy (all integers); numbers can be real, integer, or exponential; and so on. If the type of information given is not correct, DISPLY marks the field as being in error by setting its attribute as blinking in addition to the original setting. The user must correct the errant data before DISPLY will move on to another field. Once the error is corrected, DISPLY returns the rendition of the field to its original setting.

Once valid data is entered into a field, DISPLY calls the routine TRMNTR (see TRMNTR, sect 6.1.2) for any further action on the keystroke used to terminate the data field entry. If the user sign the interface that there are no more data fields to be filled in (by striking "Do" or ctrl-Z), DISPLY sca through all of the fields on the screen for fields left blank that need to be filled in. Fields that need t be filled in are treated as if they contained errant data (see above), and the user must again strike "Do" or ctrl-Z after having filled in the field correctly. After all of the required fields are filled in and any errant data corrected, DISPLY renews the virtual keyboard (as explained above) before returning to the calling routine.

If the user has requested that the cursor move to another field, TRMNTR returns the number of the next possibly accessible field. DISPLY then starts at one field before the field returned by TRMNTR and moves ahead one field at a time until an accessible (i.e., non-fixed) field is found. Note that when the initial field IBEGIN is passed from the calling routine, DISPLY decrements it by one before starting to look for the next accessible field.

6.1.2 TRMNTR

6.1.2.1 Purpose of the Routine. The source code for this FORTRAN subroutine is presente in Appendix J. TRMNTR acts on the terminator received by SMG$READ_STRING; recall th SMG$READ_STRING is used in DISPLY to read user input to a data screen.

6.1.2.2 Flow Description of the Routine. The calling routine passes to TRMNTR most of the information needed to describe the fields to be displayed on the user's monitor. This includes what text is to be printed, the rendition in which it is to be printed, and where it is to be printed. Declaration statements for the variables to contain this information follow, along with the necessary screen management system routine include files.

Some of the actions taken by TRMNTR have no immediate impact. For example, if the user strikes PF4 (Advance) or PF5 (Reverse), TRMNTR remembers that the user has specified a direction to be used with any subsequent "Next" keypad keys (all of which move the cursor to the next field). TRMNTR acts on most of the other terminators to specify the next field to which DISPLY should try to move. For example, striking the up-arrow key causes TRMNTR to scan backwards through the locations of the fields until it reaches one which is on a line above the current field. TRMNTR is set up to treat the data screen as a sphere; for example, striking the down-arrow key from the bottom line initially moves the cursor to the top line.

Other actions performed by TRMNTR include those special cases DISPLY handles directly (the main routine also calls TRMNTR, for example). It can also erase the contents of a data field when directed to do so by either "Remove" or one of the "Delete" keypad keys. And if it does not recognize the terminator, TRMNTR does not move the cursor. Note that the tab key is not a terminator. The VMS default set of terminators recognized by the screen management routines does not include the tab key. The programmer can construct a set of terminators that includes the tab key for use in the SMG$READ_-STRING calls.

6.1.3 CHOOSE

6.1.3.1 Purpose of the Routine. The source code for the FORTRAN subroutine CHOOSE is presented in Appendix K. This routine makes use of VAX VMS screen management system routine to overlay a second screen on top of the data entry screen. This overlaid screen provides the user wit some choices for a field (or series of fields) to be filled in. For this user interface, CHOOSE is used to

display a list of known sample possessors to speed the updating of sample entries. It performs as a scaled-down version of DISPLY, calling TRMNTR as is needed.

6.1.3.2 Flow Description of the Routine. The calling routine passes to CHOOSE all of the information needed to describe the fields to be displayed in the overlaid screen. Declaration statements for the variables to contain this information follow, along with the necessary screen management system routine include files.

After creating (SMG$CREATE_VIRTUAL_DISPLAY) and pasting (SMG$PASTE_VIRTUAL_-DISPLAY) the overlaid screen, CHOOSE renews the virtual keyboard as does DISPLY. It then displays the list, and waits for the user to either select the current item in the list or move to another. To select an item in the list, the user need only put some character in the corresponding field. Once the user strikes "Do" or ctrl-Z, CHOOSE determines which item, if any, the user selected by searching for the first field filled in. The number of the item selected is returned to the calling routine; the calling routine then fills in the field(s) appropriately.

Before returning to the calling routine, CHOOSE removes the overlaid screen (SMG$ERASE_DIS-PLAY, followed by SMG$UNPASTE_VIRTUAL_DISPLAY) and, once again, renews the virtual keyboard.

6.1.4 KEYPAD

6.1.4.1 Purpose of the Routine. The source code for the FORTRAN subroutine KEYPAD is presented in Appendix L. This routine makes use of VAX VMS screen management system routines to display the mappings of the special function and keypad keys. This mapping is presented as an overlaid screen, much as CHOOSE does.

6.1.4.2 Flow Description of the Routine. The calling routine passes no arguments to KEYPAD. Declaration statements for the variables to contain the information needed to describe the overlaid screen follow, along with the necessary screen management system routine include files. KEYPAD then creates and pastes into place the overlaid screen.

As with DISPLY, TRMNTR, and CHOOSE, KEYPAD renews the virtual keypad. KEYPAD removes the overlaid screen as soon as any key is struck. TRMNTR is not called, nor does KEYPAD handle any special function keys itself. After removing the overlaid screen, KEYPAD renews the virtual keyboard and returns to the calling routine.

6.2 For Use in Data Type Conversions

6.2.1 Character String-to-Numerical Value Conversions: RVALUE

6.2.1.1 Purpose of the Function. The source code for the FORTRAN real function RVALUE is presented in Appendix M. This function attempts to convert the data represented in the character string passed to it into a real number. The converted data is for insertion into the database. The data can represent an integer, real, or exponential number.

6.2.1.2 Flow Description of the Function. After declaring the few variables needed, RVALUE determines whether the data represents an integer by looking for a decimal point. If it does not

find a decimal point, the characters passed to it are assumed to represent an integer. Since DISPLY has already confirmed that the corresponding field does contain something that looks like a number, a failure of the internal read used to effect the conversion is assumed to be fatal to further execution. Once the internal read into an integer variable is performed, RVALUE converts the integer value to a real value, then exits.

If a decimal point is found, an internal read is performed to obtain the real number directly. Note that the format used for the internal read works for real numbers in exponential notation as well as for real numbers without an exponent.

6.2.2 Numerical-to-Character String Representation Conversions: CVALUE

6.2.2.1 Purpose of the Function. The source code for this FORTRAN function, the functional inverse of RVALUE, is presented in Appendix N. As the inverse of RVALUE, CVALUE attempts to convert the real number passed to it into a character representation; the conversion of integer values is performed in the calling routine since only an internal write is required. The character representation is for use in displaying the data from the database using screen management system routines.

A separate routine was prepared to handle real numbers so that the conversion effected (using FORTRAN F or E format) would generate the most complete character representation that will fit in the allowed field. For example, consider a field that is only eight characters long. The number to be presented in that field is 0.000000123. Using an F format, the character representation would be 0.000000 - likely not a very good representation. If the leading zero were dropped, the F format would only allow for .0000001. But if an E format was used, the representation would be 1.23E07, a much more complete representation.

6.2.2.2 Flow Description of the Function. Following the few variable declarations needed, CVALUE first tries to represent the number using an F format. The first F format attempted places as many digits as possible after the decimal point. As in the above example, this is generally the best F format to begin with.

If an error occurs during the internal write, it likely means that there are not enough digits before the decimal point to represent the number. CVALUE then effectively moves one character from after the decimal point to before it by respecifying the F format. If an error again occurs during the internal write, another character is moved. This will continue until a successful internal write is performed or there are no more characters to be moved (in which case an E format will surely be needed).

If an internal write using an F format is successful, CVALUE then calls RVALUE to obtain the real number represented by the conversion. (It is possible that the real number represented by the conversion does not have the same value as the original real number.) CVALUE then attempts to generate the best E format representation in a fashion identical to that used for generating the best F format representation. Once a successful internal write is achieved using an E format representation, the real number corresponding to the representation is again obtained using RVALUE (i.e., a back translation is performed).

Finally CVALUE decides which representation should be returned to the calling routine. If one of the conversions failed (i.e., none of the internal writes succeeded), then CVALUE returns the result of the other conversion. If both conversions fail, the character representation returned is 0.0. If both conversions succeeded, CVALUE compares the results of the two back translations to the original value.

The back-translated value closest to the original value is the one returned to the calling routine.

6.2.3 Conversions between mmddyy Dates and Integer Date Counts

6.2.3.1 Purpose of the Functions. The source code for the FORTRAN functions DAYCNT and CHRDAT are presented in Appendix O. These functions are used to convert dates of the integer form mmddyy to the day count from January 1, 1990 (day count 1) and vise versa.

6.2.3.2 Flow Description of the Routine. A small number of variables are first declared in both functions. Then the appropriate mappings are performed.

DAYCNT uses the following FORTRAN expression to directly relate an mmddyy date to a day count:

$$daycnt = (year-1990)*365 + int((year-1989)/4) + edamnt(mm) + dd,$$

where the first term gives the number of days from fully elapsed years since 1990, the second term corrects for leap years, the second term represents an array (EDAMNT) that contains the number of days fully elapsed up to the mm-th month, and the fourth term is simply the date of the month. If yy is less than 90, the date is assumed to be in the 2000s; otherwise the date is assumed to be in the 1900s, and year is determined appropriately. The leap year correction is valid for the next 100+ years; recall that years evenly divisible by 100 are not leap years unless they are also evenly divisible by 400, which the year 2000 is.

CHRDAT, the inverse of DAYCNT, does not directly obtain the mmddyy date from the day count. First it makes a rough guess at the year by dividing the day count sent to it by 365. Then, based on the rough guess, CHRDAT corrects for the number of leap years that have elapsed since 1990 to determine the correct year. With the correct year in hand, CHRDAT calculates how many days remain after all of the fully elapsed years are subtracted from the total. The month is determined by subtracting the number of days in each month from the remnant; once a negative value is obtained, CHRDAT knows that the correct month is the prior one. The final remnant is the date of the month.

6.3 For Use in Constructing Searches

6.3.1 SRCBLD

6.3.1.1 Purpose of the Routine. The source code for the FORTRAN subroutine SRCBLD is presented in Appendix P. Given the variability of searches, and that the user interface cannot know a user-specified search before it is constructed, the appropriate SQL call needs to be prepared on the fly and executed as dynamic SQL. SRCBLD constructs the SQL call, compiling the attribute and domain names and the associated values into a character string (via APSTRG) with as few extra blanks as possible. The current limit for the length of the character string is 6000 characters, split evenly between the "WHERE" clause and the rest of the cursor declaration call.

6.3.1.2 Flow Description of the Routine. The calling routine passes to SRCBLD the values of all of the fields for a data entry screen, along with the array containing the name of the appropriate attribute(s) and domains (ATRBUT). These variables are all defined in the declaration statements, along with the two pieces of the character string.

SRCBLD then runs through all of the fields to find out which ones have been filled in. Only those that have been filled in are included in the SQL call that performs the search. Once a filled-in field is found, SRCBLD adds the name of the appropriate attribute to the "FROM" clause of the SQL call. The "FROM" clause is found at the end of the first piece of the character string. For the sample tracking database, the domain SAMPLE_TRACKING_ID serves as the primary or foreign key for the various database files. To ensure that the search finds only those entries for which all of the specified information applies fully to each entry, SRCBLD automatically includes the condition that all of the SAMPLE_TRACKING_IDs from a single successful hit match. Again, the actual tacking on of the attribute name to the existing string takes place in the subroutine APSTRG (see APSTRG, section 6.3.2).

The attribute, domain, and field value are then all added to the second piece of the character string in the form <attribute>.<domain>=<field value>. This version of SRCBLD only allows for exact equalities; wildcards, ranges, and "sort of like"s have not yet been implemented.

6.3.2 APSTRG

6.3.2.1 Purpose of the Routine. The source code for this FORTRAN subroutine is presented in Appendix Q. APSTRG "appends" new character strings to an existing one. "Appends" is not totally accurate, though; APSTRG does not perform a concatenation. Rather, APSTRG keeps track of how many characters are in the existing string, and then puts the contents of the new string in after that point. This requires that the existing string have enough room for all of the new strings to be included within it.

6.3.2.2 Flow Description of the Routine. The calling routine passes to APSTRG the existing character string, the number of characters in the existing string, the new string to be added, and a flag that tells APSTRG whether the existing string should be started afresh and so contain only the new string upon return to the calling routine. These variables are all defined in the declaration statements.

If the existing string is to be started afresh, APSTRG starts by resetting the number of characters filled in in the existing string to zero. APSTRG then assumes that all of the characters passed in the new string are to be included in the existing one. Since the length of the new string can vary, APSTRG determines this length using the FORTRAN intrinsic function LEN. The new string is then copied into the existing one, the length of the new string added to the prior length of the existing string, and then APSTRG returns to the calling routine.

7. REFERENCES

1. D. A. Femec, *User's and Reference Guide to the INEL RML/Analytical Radiochemistry Sample Tracking Database Version 1.00*, INEL-95/0455, September 1995.

2. This document.

APPENDIXES

A. CREATE_SCHEMA - A ROUTINE FOR CONSTRUCTING AN INSTRUCTION SET FOR CREATING A DATABASE

```
$ ! Routine to generate an .SQL file for creating a schema from a Lotus
$ ! entity.prn file.  The storage areas for the tables are calculated,
$ ! and the tables, domains, and cursors are defined.  The attributes
$ ! marked with a "d" (for duplicate) in the Key field of the entity.prn
$ ! file are to be included in the cursors but excluded from the tables.
$ !
$ ! Version 1 completed March 26, 1991 by D. A. Femec.
$ !
$ ! Set the maximum page size at zero to start, and define a string of
$ ! thirty blanks to work with later in tabbing.
$ !
$ maximum_page_size = 0
$ thirty_blanks = "                              "
$ number_of_integers = 0
$ number_of_reals = 0
$ !
$ ! See that entity.prn exists, opening it if it does.
$ !
$ open entity entity.prn /read
$ open screen screen.prn /read
$ !
$ ! Obtain the name of the schema to be created.
$ !
$ get_schema_name:
$    write sys$output " "
$    inquire schema "Enter the name of the schema to be created"
$    if schema .eqs. "" then goto get_schema_name
$ !
$ ! Open the .SQL and other code files.
$ !
$ open sql_first create_schema_first.sql /write
$ open sql_second create_schema_second.sql /write
$ write sql_second "!"
$ write sql_second "! Create the storage areas."
$ write sql_second "!"
$ open sql_third create_schema_third.sql /write
$ open cursor_first cursor_first.for /write
$ open cursor_second cursor_second.for /write
$ open cursor_third cursor_third.for /write
$ open cursor_fourth cursor_fourth.for /write
$ open insert_first insert_first.for /write
$ open insert_second insert_second.for /write
$ open insert_third insert_third.for /write
$ open insert_fourth insert_fourth.for /write
$ open insert_fifth insert_fifth.for /write
$ write insert_fifth "C"
$ write insert_fifth "C    Database table and attribute names for due " -
   + "date checking."
$ write insert_fifth "C"
$ write insert_fifth "      data datatr/"
$ open insert_sixth insert_sixth.for /write
$ write insert_sixth "C"
$ write insert_sixth "C    Program variable (value and null indicator) "-
   + "names for due date checking."
$ write insert_sixth "C"
$ write insert_sixth "      data datvar/"
$ open delete_tables delete.for /write
$ open update_tables update.for /write
$ !
$ ! Get the domains, including the defaults, constraints, and comments.
$ !
$ write sql_third "!"
$ write sql_third "! Create the domains, including defaults, " -
   + "constraints, and comments."
```

```
$ write sql_third "!"
$ write sql_third "create domain USER_NAME char(30)"
$ write sql_third "    default NULL;"
$ write sql_third "
$ write sql_third "comment on domain USER_NAME"
$ write sql_third "    is 'Domain for user name.';"
$ write sql_third " "
$ write sql_third "create domain VMS_DATE_TIME char(18)"
$ write sql_third "    default NULL;"
$ write sql_third "
$ write sql_third "comment on domain VMS_DATE_TIME"
$ write sql_third "    is 'Domain for VMS date (dd-mmm-yy) and time (hh:mm:ss).';"
$ write sql_third " "
$ domains = ""
$ domain_types = ""
$ domain_sizes_bytes = ""
$ number_of_domains = 0
$ create_domains:
$    read entity line /end_of_file=premature_eof
$    if f$extract(108,1,line) .nes. "*" then goto create_domains
$    read entity line /end_of_file=premature_eof
$    read entity line /end_of_file=premature_eof
$    read entity line /end_of_file=premature_eof
$    domain = f$edit(f$extract(108,30,line),"trim,upcase")
$ get_domain_type:
$    write sys$output " "
$    write sys$output "For domain ''domain':"
$    write sys$output " "
$    inquire domain_type -
      "   Enter the data type for this domain (char/date/int/real)"
$    if domain_type .nes. "C" then goto date_check
$ get_char_length:
$    write sys$output " "
$    inquire char_length "      Enter the length of the character domain"
$    if char_length .le. 0 then goto get_char_length
$    domain_type = "char(" + f$string(char_length) + ")"
$    if domain_sizes_bytes .nes. "" then domain_sizes_bytes = -
      domain_sizes_bytes + "/"
$    domain_sizes_bytes = domain_sizes_bytes + f$string(char_length)
$    goto print_domain
$ date_check:
$    if domain_type .nes. "D" then goto int_check
$    domain_type = "date"
$    if domain_sizes_bytes .nes. "" then domain_sizes_bytes = -
      domain_sizes_bytes + "/"
$    domain_sizes_bytes = domain_sizes_bytes + "8"
$    goto print_domain
$ int_check:
$    if domain_type .nes. "I" then goto real_check
$    domain_type = "integer"
$    if domain_sizes_bytes .nes. "" then domain_sizes_bytes = -
      domain_sizes_bytes + "/"
$    domain_sizes_bytes = domain_sizes_bytes + "4"
$    goto print_domain
$ real_check:
$    if domain_type .nes. "R" then goto get_domain_type
$    domain_type = "real"
$    if domain_sizes_bytes .nes. "" then domain_sizes_bytes = -
      domain_sizes_bytes + "/"
$    domain_sizes_bytes = domain_sizes_bytes + "4"
$ print_domain:
$    if domains .nes. "" then domains = domains + "/"
$    domains = domains + domain
$    if domain_types .nes. "" then domain_types = domain_types + "/"
$    domain_types = domain_types + domain_type
$    number_of_domains = number_of_domains + 1
$    write sql_third "create domain ''domain' ''domain_type'"
$    write sys$output " "
$    inquire default "   Enter the default value (return says NULL)"
$    if (default .nes. "") .and. ((f$extract(0,4,domain_type) .eqs. "char") -
      .or. (f$extract(0,4,domain_type) .eqs. "date")) then -
      default = "'" + default + "'"
$    if (default .nes. "") .and. (f$extract(0,4,domain_type) .eqs. "real") -
```

```
      then default = default + "E0"
$     if default .eqs. "" then default = "NULL"
$     write sql_third "   default ''default';"
$     write sql_third " "
$ get_comment:
$     write sys$output " "
$     inquire domain_comment "   Enter any desired comment"
$     if domain_comment .eqs. "" then goto next_domain
$     write sql_third "comment on domain ''domain'"
$     write sql_third "   is " + "'" + domain_comment + "';"
$     write sql_third " "
$ next_domain:
$     read entity line /end_of_file=create_tables
$     domain = f$edit(f$extract(108,30,line),"trim,upcase")
$     if domain .nes. "" then goto get_domain_type
$ !
$ ! Begin rereading the entity file and construct the appropriate
$ ! statements for creating tables.
$ !
$ create_tables:
$     close entity
$     open entity entity.prn /read
$     if number_of_domains .eq. 0 then goto no_domains
$ read_next_line:
$     read entity line /end_of_file=premature_eof
$     if f$extract(0,6,line) .nes. "Entity" then goto read_next_line
$     read entity line /end_of_file=premature_eof
$     table = f$edit(f$extract(0,30,line),"trim,upcase")
$     write sql_third "!"
$     write sql_third "! Create tables."
$     write sql_third "!"
$ !
$ ! Prepare to create a new table.
$ !
$ prepare_new_table:
$ !
$ ! Construct an abbreviated table name for the primary key index since
$ ! the full table name is probably too long to allow a 30-alphanumeric
$ ! index name to be constructed.
$ !
$     abbreviated_table = f$extract(0,2,table)
$     under_bar = 0
$ next_under_bar:
$     under_bar = under_bar + 1
$     fragment = f$element(under_bar,"_",table)
$     if fragment .eqs. "_" then goto abbreviated_table_complete
$     abbreviated_table = abbreviated_table + f$extract(0,2,fragment)
$     goto next_under_bar
$ abbreviated_table_complete:
$     write sql_third "create table ''table' ("
$     write cursor_first "C"
$     write cursor_first "C    Declare a cursor for the contents of " -
       + "table ''table'."
$     write cursor_first "C"
$     write cursor_first "         exec sql declare " -
       + "''abbreviated_table'_cursor cursor for"
$     write cursor_first "      .      select"
$     write cursor_third "C"
$     write cursor_third "C    Declare a cursor for the due dates in " -
       + "table ''table'."
$     write cursor_third "C"
$     write cursor_third "         exec sql declare " -
       + "''abbreviated_table'_due_cursor cursor for"
$     write cursor_third "      .      select ''table'.SAMPLE_TRACKING_ID"
$     write cursor_third "      .      from ''table'"
$     write cursor_third "      .      where"
$     write cursor_second "C"
$     write cursor_second "C    Open the cursor for the contents of " -
       + "table ''table'."
$     write cursor_second "C"
$     write cursor_second "         exec sql open ''abbreviated_table'_cursor"
$     write cursor_second "C"
$     write cursor_second "C    Close the cursor for the contents of " -
```

```
     + "table ''table'."
$    write cursor_second "C"
$    write cursor_second "         exec sql close ''abbrev  ted_table'_cursor"
$    write cursor_second "C"
$    write cursor_second "C    Fetch a row from cursor for the table ''table'."
$    write cursor_second "C    Perform the requisite internal writes " - .
     + "to transfer from noncharacter fields."
$    write cursor_second "C"
$    write cursor_second "         exec sql fetch ''abbreviated_table'_cursor" -
     + " into"
$    write cursor_fourth "C"
$    write cursor_fourth "C    Open the cursor for the due dates in " -
     + "table ''table'."
$    write cursor_fourth "C"
$    write cursor_fourth "         exec sql open ''abbreviated_table'_due_cursor"
$    write cursor_fourth "C"
$    write cursor_fourth "C    Close the cursor for the due dates in " -
     + "table ''table'."
$    write cursor_fourth "C"
$    write cursor_fourth "         exec sql close ''abbreviated_table'_due_cursor"
$    write cursor_fourth "C"
$    write cursor_fourth "C    Fetch a row from cursor for the table ''table'."
$    write cursor_fourth "C    Perform the requisite internal writes " -
     + "to transfer from noncharacter fields."
$    write cursor_fourth "C"
$    write cursor_fourth "         exec sql fetch ''abbreviated_table'_due_cursor" -
     + " into"
$    write insert_second "C"
$    write insert_second "C    Perform the requisite internal reads " -
     + "to transfer into noncharacter fields."
$    write insert_second "C    Insert the row into table ''table'."
$    write insert_second "C"
$    write insert_second "         exec sql insert into ''table' values("
$    write delete_tables "C"
$    write delete_tables "C    Delete the entry from table ''table'."
$    write delete_tables "C"
$    write delete_tables "         exec sql delete from ''table'"
$    write update_tables "C"
$    write update_tables "C    Update the entry in table ''table'."
$    write update_tables "C"
$    write update_tables "            exec sql update ''table' set"
$    primary_key = ""
$    primary_key_found = "F"
$    foreign_key_found = "F"
$    primary_key_size_byt  = 0
$    primary_key_size_at   butes = 0
$    column_constraint =   "
$    row_size_bytes = 48   !Account for USER_NAME (30 bytes) and
$    number_of_attributes = 2   !VMS_DATE_TIME (18 bytes).
$    cursor_where_first = ""
$    cursor_where_second = ""
$    cursor_where_third = ""
$    cursor_order_by_first = ""
$    cursor_order_by_second = ""
$    cursor_order_by_third = ""
$    cursor_from = ""
$    write sys$output " "
$    write sys$output "For table ''table':"
$    write sys$output " "
$    inquire table_comment "   Enter any desired comment"
$ get_attribute:
$    attribute = f$edit(f$extract(31,30,line),"trim,upcase")
$    table_current = f$edit(f$extract(0,30,line),"trim,upcase")
$    if (f$locate(table_current,cursor_from) .eqs. f$length(cursor_from)) -
     .and. (cursor_from .nes. "") then cursor_from = cursor_from -
     + ", " + table_current
$    if cursor_from .eqs. "" then cursor_from = table_current
$    domain = f$edit(f$extract(78,30,line),"trim,upcase")
$    element = f$edit(f$extract(66,1,line),"upcase") + "(" -
     + f$edit(f$extract(75,2,line),"trim") + ")"
$    serial = f$edit(f$extract(66,1,line),"upcase") -
     + f$edit(f$extract(75,2,line),"trim")
$    domain_number = 0
```

```
$ check_next_domain:
$    if domain .eqs. f$element(domain_number,"/",domains) then goto domain_found
$    domain_number = domain_number + 1
$    if f$element(domain_number,"/",domains) .eqs. "/" then goto no_domain_found
$    goto check_next_domain
$ domain_found:
$ !
$ ! If this is a duplicate attribute, skip the following so as to not
$ ! include the attribute in the table, and read the next line.
$ !
$    duplicate_attribute = f$extract(61,1,line) .eqs. "d"
$    if duplicate_attribute then goto next_line
$    row_size_bytes = row_size_bytes -
      + f$integer(f$element(domain_number,"/",domain_sizes_bytes))
$    number_of_attributes = number_of_attributes + 1
$ !
$ ! Check if this attribute has been specified as a primary key (or as
$ ! part of one).
$ !
$    if f$locate("p",f$extract(61,5,line)) .eq. 5 then goto check_foreign_key
$    if primary_key_found then primary_key = primary_key + ", "
$    primary_key = primary_key + attribute
$    primary_key_size_bytes = primary_key_size_bytes -
      + f$integer(f$element(domain_number,"/",domain_sizes_bytes))
$    primary_key_size_attributes = primary_key_size_attributes + 1
$    column_constraint = f$extract(0,(31-f$length(domain)),thirty_blanks) -
      + " not null,"
$    primary_key_found = "T"
$    if (cursor_where_first .nes. "") .and. (cursor_where_second .nes. "") then -
      cursor_where_third = table + "." + attribute + "=:..."
$    if (cursor_where_first .nes. "") .and. (cursor_where_second .eqs. "") then -
      cursor_where_second = table + "." + attribute + "=:..."
$    if cursor_where_first .eqs. "" then cursor_where_first = table + "." -
      + attribute + "=:..."
$    if (cursor_order_by_first .nes. "") .and. -
      (cursor_order_by_second .nes. "") then cursor_order_by_second -
      = cursor_order_by_second + ","
$    if (cursor_order_by_first .nes. "") .and. -
      (cursor_order_by_second .nes. "") then cursor_order_by_third = table -
      + "." + attribute + " desc"
$    if (cursor_order_by_first .nes. "") .and. -
      (cursor_order_by_second .eqs. "") then cursor_order_by_first -
      = cursor_order_by_first + ","
$    if (cursor_order_by_first .nes. "") .and. -
      (cursor_order_by_second .eqs. "") then cursor_order_by_second = table -
      + "." + attribute + " desc"
$    if cursor_order_by_first .eqs. "" then cursor_order_by_first = table -
      + "." + attribute + " desc"
$ !
$ ! Check if this attribute has been specified as the foreign key.
$ ! Obtain the appropriate reference table and column if it has.
$ !
$ check_foreign_key:
$    if foreign_key_found then goto next_line
$    if f$locate("f",f$extract(61,5,line)) .eq. 5 then goto next_line
$    write sys$output " "
$    write sys$output "   For foreign key ''attribute':"
$    write sys$output " "
$    inquire reference_table "      Enter the reference table"
$    write sys$output " "
$    inquire reference_column "      Enter the reference column"
$    foreign_key = attribute
$    column_constraint = f$extract(0,(31-f$length(domain)),thirty_blanks) -
      + " not null,"
$    foreign_key_found = "T"
$ !
$ ! Read the next line from the entity.prn file.
$ !
$ next_line:
$    read entity line /end_of_file=done
$    if f$edit(f$extract(0,108,line),"trim") .eqs. "" then goto next_line
$    if f$extract(0,1,line) .eqs. "-" then goto dashed_line
$ print_attribute:
```

```
$ !
$ ! Print the cursor declaration and open/fetch statements.
$ !
$    write cursor_first "     .          ''table_current''.''attribute',"
$    if f$extract(0,1,f$element(domain_number,"/",domain_types)) .nes. "i" -
      then goto check_if_real
$    if domain .eqs. "INTEGER_DATE_COUNT" then write cursor_third -
      "     .          ''table_current''.''attribute',"
$    number_of_integers = number_of_integers + 1
$    write cursor_second -
      "     .      :intg''number_of_integers':ifn''serial',"
$    write cursor_second "         if(ifn''serial'.eq.-1)intg''number_of_integers'=0"
$    write cursor_second "         write(fld''serial'," -
      + "fmt='(i<mxnoc''element'>)')intg''number_of_integers'"
$    if domain .eqs. "INTEGER_DATE_COUNT" then write cursor_fourth -
      "     .      :intg''number_of_integers':ifn''serial',"
$    if duplicate_attribute then goto get_attribute
$    write insert_second -
      "     .      :intg''number_of_integers':ifn''serial',"
$    write insert_second "         if(ifn''serial' eq.-1)then
$    write insert_second "              intg''number_of_integers'=0"
$    write insert_second "            else"
$    write insert_second "              read(fld  erial'," -
      + "fmt='(i<mxnoc''element'>)')intg''number_  ntegers'"
$    write insert_second "            endif"
$    write update_tables -
      "     .          ''attribute'=:intg''number_of_integers':ifn''serial',"
$    goto print_attribute_to_sql
$ check_if_real:
$    if f$extract(0,1,f$element(domain_number,"/",domain_types)) .nes. "r" -
      then goto character_of_sorts
$    number_of_reals = number_of_reals + 1
$    write cursor_second "     .      :real''number_of_reals'" -
      + ":ifn''serial',"
$    write cursor_second "         if(ifn''serial'.eq.-1)real''number_of_reals'=0.0"
$    write cursor_second "         write(fld''serial'," -
      + "fmt='(f<mxnoc''element'-2>.1)')real''number_of_reals'"
$    if duplicate_attribute then goto get_attribute
$    write insert_second -
      "     .      :real''number_of_reals':ifn''serial',"
$    write insert_second "         if(ifn''serial'.eq.-1)then
$    write insert_second "              real''number_of_reals'=0.0"
$    write insert_second "            else"
$    write insert_second "              read(fld''serial'," -
      + "fmt='(f<mxnoc''element'-2>.1)')real''number_of_reals'"
$    write insert_second "            endif"
$    write update_tables "     .          ''attribute'=:real''number_of_  '" -
      + ":ifn''serial',"
$    goto print_attribute_to_sql
$ character_of_sorts:
$    write cursor_second -
      "     .      :fld''serial':ifn''serial',"
$    if duplicate_attribute then goto get_attribute
$    write insert_second -
      "     .      :fld''serial':ifn''serial',"
$    write update_tables -
      "     .          ''attribute'=:fld''serial':ifn''serial',"
$ print_attribute_to_sql:
$    write sql_third "     ''attribute' ''f$extract(0, -
      (31-f$length(attribute)),thirty_blanks)' ''domain''column_constraint'"
$    column_constraint = ","
$    goto get_attribute
$ dashed_line:
$    read entity line /end_of_file=done
$    if f$edit(f$extract(0,108,line),"trim") .eqs. "" then goto dashed_line
$    table_next = f$edit(f$extract(0,30,line),"trim,upcase")
$ !
$ ! Fi  sh "creation of table" instructions.
$ !
$    gosub finis  _with_table
$    write sql_  cond " "
$ !
$ ! Prepare to create a new table.
```

```
$ !
$    goto prepare_new_table
$ !
$ ! Normal termination.
$ !
$ done:
$ !
$ ! Finish create schema instruction.
$ !
$    gosub finished_with_table
$    write sql_second "   ;"
$    write sql_second " "
$ !
$ ! Write the create schema command - waited until now so that the
$ ! buffer size (maximum page size from the storage areas) would be
$ ! known.
$ !
$    buffer_size = 3 * maximum_page_size
$    write sql_first "! SQL commands for creating ''schema' schema."
$    write sql_first "!"
$    write sql_first "! Define the data and snapshot file location " -
      + "logical names."
$    write sql_first "!"
$    write sql_first "$ define/system/nolog/translation_attributes=concealed " -
      + "rdb_data $user:[femec.database.tracking.]"
$    write sql_first "$ define/system/nolog/translation_attributes=concealed " -
      + "rdb_snap $user:[femec.database.tracking.]"
$    write sql_first "!"
$    write sql_first "! Create the schema."
$    write sql_first "!"
$    write sql_first "create schema"
$    write sql_first "   filename ""''schema'"""
$    write sql_first "   buffer size is ''buffer_size'"
$    write sql_first " "
$ !
$ ! Commit the schema creation then exit.
$ !
$    write sql_third "commit;"
$    write sql_third " "
$    write sql_third "exit;"
$ !
$ ! Include declaration statements for real and integer attribute storage.
$ !
$    write insert_first "C"
$    write insert_first "C    Declaration statements."
$    write insert_first "C"
$ real_number = 1
$    write insert_first "      real*4 "
$ next_real:
$    write insert_first "     . real''real_number',"
$    real_number = real_number + 1
$    if real_number .lt. number_of_reals then goto next_real
$    write insert_first "     . real''real_number'"
$ integer_number = 1
$    write insert_first "      integer*4 "
$ next_integer:
$    write insert_first "     . intg''integer_number',"
$    integer_number = integer_number + 1
$    if integer_number .lt. number_of_integers then goto next_integer
$    write insert_first "     . intg''integer_number'"
$    type sys$input

   Execution has successfully completed.  The create_schema.sql and cursor
and insert files will be closed and retained.  The entity and screen
files will also be closed and retained.

   Please wait while certain sections of the cursor and insert files are
rearranged; this may take a few minutes.

$    close sql_first
$    close sql_second
$    close sql_third
$    copy/concatenate create_schema_first.sql + create_schema_second.sql -
```

```
    + create_schema_third.sql create_'schema.sql
$ !
$ ! Reorder the contents of cursor_second so that the internal write
$ ! statements occur after the respective fetch.
$ !
$    close cursor_first
$    close cursor_second
$    close cursor_third
$    close cursor_fourth
$    open cursor_second cursor_second.for /read
$    open cursor_second_a cursor_second.for /write
$    open cursor_second_b cursor_second_b.for /write
$    read cursor_second line /end_of_file=end_of_cursor_second
$ echo_comments_to_cursor_second:
$    write cursor_second_a line
$    read cursor_second line /end_of_file=end_of_cursor_second
$    write cursor_second_a line
$    read cursor_second line /end_of_file=end_of_cursor_second
$    write cursor_second_a line
$    read cursor_second line /end_of_file=end_of_cursor_second
$    write cursor_second_a line
$    read cursor_second line /end_of_file=end_of_cursor_second
$    write cursor_second_a line
$    read cursor_second line /end_of_file=end_of_cursor_second
$    write cursor_second_a line
$    read cursor_second line /end_of_file=end_of_cursor_second
$    write cursor_second_a line
$    read cursor_second line /end_of_file=end_of_cursor_second
$    write cursor_second_a line
$    read cursor_second line /end_of_file=end_of_cursor_second
$    write cursor_second_a line
$    read cursor_second line /end_of_file=end_of_cursor_second
$    write cursor_second_a line
$    read cursor_second line /end_of_file=end_of_cursor_second
$    write cursor_second_a line
$    read cursor_second line /end_of_file=end_of_cursor_second
$    write cursor_second_a line
$ read_from_cursor_second:
$    read cursor_second line /end_of_file=end_of_cursor_second
$    if f$extract(0,1,line) .eqs. "C" then goto new_cursor
$    if f$extract(9,6,line) .nes. "if(ifn" then write cursor_second_a line
$    if f$extract(9,6,line) .nes. "if(ifn" then goto read_from_cursor_second
$    write cursor_second_b line
$    read cursor_second line /end_of_file=end_of_cursor_second
$    write cursor_second_b line
$    goto read_from_cursor_second
$ new_cursor:
$    close cursor_second_a
$    close cursor_second_b
$    copy/concatenate cursor_second.for + cursor_second_b.for -
      cursor_second.for
$    delete cursor_second_b.for;
$    open cursor_second_a cursor_second.for /append
$    open cursor_second_b cursor_second_b.for /write
$    goto echo_comments_to_cursor_second
$ end_of_cursor_second:
$    close cursor_second
$    close cursor_second_a
$    close cursor_second_b
$    copy/concatenate cursor_second.for + cursor_second_b.for -
      cursor_second.for
$    delete cursor_second_b.for;
$    purge cursor_second.for
$    copy/concatenate cursor_first.for + cursor_second.for -
      + cursor_third.for + cursor_fourth.for cursor.for
$    type sys$input
  Cursor file rearranged.

$ !
$ ! Reorder the contents of insert_second so that the internal read
$ ! statements occur before the respective insert.
$ !
$    close insert_first
```

```
$    close insert_second
$    open insert_second insert_second.for /read
$    open insert_second_a insert_second.for /write
$    open insert_second_b insert_second_b.for /write
$    read insert_second line /end_of_file=end_of_insert_second
$ echo_comments_to_insert_second:
$    write insert_second_a line
$    read insert_second line /end_of_file=end_of_insert_second
$    write insert_second_a line
$    read insert_second line /end_of_file=end_of_insert_second
$    write insert_second_a line
$    read insert_second line /end_of_file=end_of_insert_second
$    write insert_second_a line
$ read_from_insert_second:
$    read insert_second line /end_of_file=end_of_insert_second
$    if f$extract(0,1,line) .eqs. "C" then goto new_insert
$    if f$extract(9,6,line) .nes. "if(ifn" then write insert_second_b line
$    if f$extract(9,6,line) .nes. "if(ifn" then goto read_from_insert_second
$    write insert_second_a line
$    read insert_second line /end_of_file=end_of_insert_second
$    write insert_second_a line
$    read insert_second line /end_of_file=end_of_insert_second
$    write insert_second_a line
$    read insert_second line /end_of_file=end_of_insert_second
$    write insert_second_a line
$    read insert_second line /end_of_file=end_of_insert_second
$    write insert_second_a line
$    goto read_from_insert_second
$ new_insert:
$    close insert_second_a
$    close insert_second_b
$    copy/concatenate insert_second.for + insert_second_b.for -
      insert_second.for
$    delete insert_second_b.for;
$    open insert_second_a insert_second.for /append
$    open insert_second_b insert_second_b.for /write
$    goto echo_comments_to_insert_second
$ end_of_insert_second:
$    close insert_second
$    close insert_second_a
$    close insert_second_b
$    type sys$input
   Insert file rearranged.

$ !
$ ! Now get table and attribute names and the associated fields ordered
$ ! by screen.
$ !
$    number_of_integers = 0
$    number_of_reals = 0
$ looking_for_entity:
$    read screen line /end_of_file=premature_eof
$    if f$extract(0,6,line) .nes. "Entity" then goto looking_for_entity
$    read screen line /end_of_file=premature_eof
$ new_screen:
$    screen_ltr = f$edit(f$extract(66,1,line),"upcase")
$    write insert_third "C"
$    write insert_third "C    Database table and attribute names for screen fields."
$    write insert_third "C"
$    write insert_third "      data atrbt''screen_ltr'/"
$    write insert_fourth "C"
$    write insert_fourth "C    Program variable names (value and null " -
      + "indicator) for screen fields."
$    write insert_fourth "C"
$    write insert_fourth "      data varbl''screen_ltr'/"
$ next_variable_descriptor:
$    attribute = f$edit(f$extract(31,30,line),"trim,upcase")
$    table = f$edit(f$extract(0,30,line),"trim,upcase")
$    serial = f$edit(f$extract(66,1,line),"upcase") -
      + f$edit(f$extract(75,2,line),"trim")
$    domain = f$edit(f$extract(78,30,line),"trim,upcase")
$    domain_number = 0
$ get_domain_number:
```

```
$    if domain .eqs. f$element(domain_number,"/",domains) then -
      goto get_next_line
$    domain_number = domain_number + 1
$    if f$element(domain_number,"/",domains) .eqs. "/" then goto no_domain_found
$    goto get_domain_number
$ get_next_line:
$    read screen line /end_of_file=done_with_screen
$    if f$edit(f$extract(0,108,line),"trim") .eqs. "" then goto get_next_line
$    if f$edit(f$extract(66,1,line),"upcase") .nes. screen_ltr then -
      goto next_screen
$    write insert_third "     . '" + table + "','" + attribute + "',"
$    if f$extract(0,1,f$element(domain_number,"/",domain_types)) .nes. "i" -
      then goto is_it_real
$    number_of_integers = number_of_integers + 1
$    write insert_fourth "     . ':intg''number_of_integers':ifn''serial'',"
$    if domain .nes. "INTEGER_DATE_COUNT" then goto next_variable_descriptor
$    write insert_fifth "     . '" + table + "','" + attribute + "',"
$    write insert_sixth "     . ':intg''number_of_integers':ifn''serial'',"
$    goto next_variable_descriptor
$ is_it_real:
$    if f$extract(0,1,f$element(d     _number,"/",domain_types)) .nes. "r" -
      then goto must_be_character
$    number_of_reals = number_of_     + 1
$    write insert_fourth "     .     l''number_of_reals':ifn''serial'',"
$    goto next_variable_descriptor
$ must_be_character:
$    write insert_fourth "     . ':fld''serial':ifn''serial'',"
$    goto next_variable_descriptor
$ next_screen:
$    write insert_third "     . '" + table + "','" + attribute + "/"
$    if f$extract(0,1,f$element(domain_number,"/",domain_types)) .nes. "i" -
      then goto is_this_real.
$    number_of_integers = number_of_integers + 1
$    write insert_fourth "     . ':intg''number_of_integers':ifn''serial''/"
$    if domain .nes. "INTEGER_DATE_COUNT" then goto new_screen
$    write insert_fifth "     . '" + table + "','" + attribute + "',"
$    write insert_sixth "     . ':intg''number_of_integers':ifn''serial'',"
$    goto new_screen
$ is_this_real:
$    if f$extract(0,1,f$element(domain_number,"/",domain_types)) .nes. "r" -
      then goto this_is_character
$    number_of_reals = number_of_reals + 1
$    write insert_fourth "     . ':real''number_of_reals':ifn''serial''/"
$    goto new_screen
$ this_is_character:
$    write insert_fourth "     . ':fld''serial':ifn''serial''/"
$    goto new_screen
$ done_with_screen:
$    write insert_third "     . '" + "''table'" + "','" + "''attribute''/"
$    if f$extract(0,1,f$element(domain_number,"/",domain_types)) .nes. "i" -
      then goto is_this_real_last
$    number_of_integers = number_of_integers + 1
$    write insert_fourth "     . ':intg''number_of_integers':ifn''serial''/"
$    if domain .nes. "INTEGER_DATE_COUNT" then goto close_inserts
$    write insert_fifth "     . '" + table + "','" + attribute + "'/"
$    write insert_sixth "     . ':intg''number_of_integers':ifn''serial''/"
$    goto close_inserts
$ is_this_real_last:
$    if f$extract(0,1,f$element(domain_number,"/",domain_types)) .nes. "r" -
      then goto this_is_character_last
$    number_of_reals = number_of_reals + 1
$    write insert_fourth "     . ':real''number_of_reals':ifn''serial''/"
$    goto close_inserts
$ this_is_character_last:
$    write insert_fourth "     . ':fld''serial':ifn''serial''/"
$ close_inserts:
$    close insert_third
$    close insert_fourth
$    close insert_fifth
$    close insert_sixth
$    copy/concatenate insert_second.for + insert_second_b.for -
      insert_second.for
$    delete insert_second_b.for;
```

```
$    purge insert_second.for
$    copy/concatenate insert_first.for + insert_second.for + insert_third.for -
      + insert_fourth.for + insert_fifth.for + insert_sixth.for insert.for
$    goto close_then_exit
$ !
$ ! Abnormal terminations.
$ !
$ !    Premature end-of-file detected.
$ !
$ premature_eof:
$    type sys$input
```

A premature end-of-file was detected. The create_schema.sql file being created will be deleted. The entity input file will be closed but not deleted.

```
$    goto abort
$ !
$ !    No domains entered.
$ !
$ no_domains:
$    type sys$input
```

No domains were entered, hence no attributes can be created. The create_schema.sql and cursor and insert files being created will be deleted. The entity input file will be closed but not deleted.

```
$    goto abort
$ !
$ !    No matching domain found.
$ !
$ no_domain_found:
$    type sys$input
```

No matching domain was found for the current attribute. The create_schema.sql and cursor and insert files being created will be deleted. The entity input file will be closed but not deleted.

```
$ !
$ ! Abortive closure.
$ !
$ abort:
$    close sql_first
$    close sql_second
$    close sql_third
$    close cursor_first
$    close cursor_second
$    close cursor_third
$    close cursor_fourth
$    close insert_first
$    close insert_second
$    close insert_third
$    close insert_fourth
$    close insert_fifth
$    close insert_sixth
$    close update_tables
$    close delete_tables
$    delete/log create_'schema.sql;
$    delete/log cursor.for;
$    delete/log insert.for;
$    delete/log update.for;
$    delete/log delete.for;
$ !
$ ! Close the input files and delete the temporary output files before exiting.
$ !
$ close_then_exit:
$    close entity
$    close screen
$    close update_tables
$    close delete_tables
$    delete insert_*.for;
$    delete cursor_*.for;
$    delete create_schema_*.sql;
```

```
$    exit
$ !
$ ! "Finished with table" GOSUB definition.
$ !
$ ! Storage area size calculation (see Rdb/VMS MAINT, section 14.4):
$ !    the primary key will be used as the index for sorting;
$ !    node_size = 3 * (primary_key_size_bytes + primary_key_size_attributes
$ !     + 11) + 32;
$ !    row_size_bytes = row size in bytes, obtained by summing over all
$ !     the respective column (attribute) domain sizes;
$ !    row_overhead_bytes = number of no-compression bytes,
$ !     (row_size_bytes + 64)/128, + 2 for version number + number of
$ !     null bytes, (number_of_attributes+4)/8 + 2 for record identifier
$ !     + 3 for control information; will add one extra no-compression
$ !     byte and one extra null byte to avoid none being provided for;
$ !    variable page overhead = system record bytes, 8 + duplicate node
$ !     record line and TSN index bytes, 8 + (data row line and TSN
$ !     index bytes, 8) * number of data rows per page
$ !
$ !    Number of 512-byte block pages for table and sorted index storage area =
$ !     (number_of_data_rows_per_page * (row_size_bytes + row_overhead_bytes +
$ !     8 + primary_key_size_bytes + primary_key_size_attributes + 11)) +
$ !     (fixed page overhead, 40) + (variable page overhead, 16) +
$ !     (node size overhead, 32))/512 + 1
$ !
$ !    Will use one-tenth of the initial size as the storage area extent.
$ !
$ !    Will use ten pages for the snapshot file allocation, and two
$ !    pages for each snapshot file extent.
$ !
$ finished_with_table:
$ !
$ ! Print the cursor declaration and open/fetch statements.
$ !
$    write cursor_first "     .          ''table_current'.''attribute'"
$    if f$extract(0,1,f$element(domain_number,"/",domain_types)) .nes. "i" -
      then goto if_real
$    if domain .eqs. "INTEGER_DATE_COUNT" then write cursor_third -
       "     .          ''table_current'.''attribute'"
$    number_of_integers = number_of_integers + 1
$    write cursor_second -
       "     .       :intg''number_of_integers':ifn''serial'"
$    if .not.duplicate_attribute then write insert_second -
       "     .       :intg''number_of_integers':ifn''serial')"
$    write cursor_second -
       "          if(ifn''serial'.eq.-1)intg''number_of_integers'=0"
$    write cursor_second "          write(fld''serial'," -
      + "fmt='(i<mxnoc''element'>)')intg''number_of_integers'"
$    if domain .eqs. "INTEGER_DATE_COUNT" then write cursor_fourth -
       "     .       :intg''number_of_integers':ifn''serial'"
$    write insert_second "          if(ifn''serial'.eq.-1)then
$    write insert_second "               intg''number_of_integers'=0"
$    write insert_second "            else"
$    write insert_second "               read(fld''serial'," -
      + "fmt='(i<mxnoc''element'>)')intg''number_of_integers'"
$    write insert_second "            endif"
$    write update_tables -
       "     .          ''attribute'=:intg''number_of_integers':ifn''serial',"
$    goto last_attribute_to_sql
$ if_real:
$    if f$extract(0,1,f$element(domain_number,"/",domain_types)) .nes. "r" -
      then goto if_character_of_sorts
$    number_of_reals = number_of_reals + 1
$    write cursor_second "     .       :real''number_of_reals'" -
      + ":ifn''serial'"
$    if .not.duplicate_attribute then write insert_second -
       "     .       :real''number_of_reals':ifn''serial')"
$    write cursor_second "          if(ifn''serial'.eq.-1)real''number_of_reals'=0.0"
$    write cursor_second "          write(fld''serial'," -
      + "fmt='(f<mxnoc''element'-2>.1)')real''number_of_reals'"
$    write insert_second "          if(ifn''serial'.eq.-1)then
$    write insert_second "               real''number_of_reals'=0.0"
$    write insert_second "            else"
```

```
$    write insert_second "                read(fld''serial',"
      + "fmt='(f<mxnoc''element'-2>.1)')real''number_of_reals'"
$    write insert_second "              endif"
$    write update_tables "     .            ''attribute'=:real''number_of_reals'" -
      + ":ifn''serial',"
$    goto last_attribute_to_sql
$ if_character_of_sorts:
$    write cursor_second -
      "     .      :fld''serial':ifn''serial'"
$    if .not.duplicate_attribute then write insert_second -
      "     .      :fld''serial':ifn''serial',"
$    write update_tables -
      "     .            ''attribute'=:fld''serial':ifn''serial',"
$ last_attribute_to_sql:
$    write update_tables -
      "     .            USER_CREATING_OR_MODIFIYING=USER,"
$    write update_tables -
      "     .            DATE_CREATED_OR_MODIFIED=TODAY"
$    write insert_second "     .      USER,TODAY)"
$    if .not.duplicate_attribute then write sql_third -
      "      ''attribute' ''f$extract(0,(31-f$length(attribute)), -
     thirty_blanks)' ''domain''f$extract(0,(f$length(column_constraint)-1), -
     column_constraint)',"
$    write sql_third "      USER_CREATING_OR_MODIFIYING     USER_NAME,"
$    write sql_third "      DATE_CREATED_OR_MODIFIED        VMS_DATE_TIME,"
$    write cursor_first "     .      from ''cursor_from'"
$    write cursor_first "     .      where ''cursor_where_first'"
$    if cursor_where_second .nes. "" then write cursor_first -
      "     .          and ''cursor_where_second'"
$    if cursor_where_third .nes. "" then write cursor_first -
      "     .          and ''cursor_where_third'"
$    write cursor_first "     .      order by ''cursor_order_by_first'"
$    if cursor_order_by_second .nes. "" then write cursor_first -
      "     .          ''cursor_order_by_second'"
$    if cursor_order_by_third .nes. "" then write cursor_first -
      "     .          ''cursor_order_by_third'"
$    write delete_tables "     .      where ''cursor_where_first'"
$    if cursor_where_second .nes. "" then write delete_tables -
      "     .          and ''cursor_where_second'"
$    if cursor_where_third .nes. "" then write delete_tables -
      "     .          and ''cursor_where_third'"
$    write update_tables "     .          where ''cursor_where_first'"
$    if cursor_where_second .nes. "" then write update_tables -
      "     .          and ''cursor_where_second'"
$    if cursor_where_third .nes. "" then write update_tables -
      "     .          and ''cursor_where_third'"
$    column_constraint = ","
$    if .not.foreign_key_found then goto no_foreign_key_found
$    write sql_third "   primary key (''primary_key'),"
$    write sql_third "   foreign key (''foreign_key')"
$    write sql_third "      references ''reference_table' " -
      + "(''reference_column') );"
$    write cursor_third "     .      order by ''table'.SAMPLE_TRACKING_ID desc"
$    goto finish_write
$ no_foreign_key_found:
$    write sql_third "   primary key (''primary_key') );"
$ finish_write:
$    write sql_third " "
$    if table_comment .eqs. "" then goto get_number_of_rows_per_page
$    write sql_third "comment on table ''table'"
$    write sql_third "   is '" + table_comment + "';"
$    write sql_third " "
$ get_number_of_rows_per_page:
$    write sys$output " "
$    inquire number_of_rows_per_page -
      "   Enter the number of rows per page to be clustered together"
$    if number_of_rows_per_page .eqs. "" then number_of_rows_per_page = 3
$    if number_of_rows_per_page .lt. 1 then goto get_number_of_rows_per_page
$ get_number_of_rows_per_table:
$    write sys$output " "
$    inquire number_of_rows_per_table -
      "   Enter the maximum number of rows for this table"
$    if number_of_rows_per_table .eqs. "" then number_of_rows_per_table = 1
```

```
$     if number_of_rows_per_table .lt. 1 then goto get_number_of_rows_per_table
$     row_overhead_bytes = (row_size_bytes + 64)/128 -
       + (number_of_attributes + 4)/8 + 9
$     node_size = 3 * (primary_key_size_bytes + primary_key_size_attributes -
       + 11) + 32
$     write sql_third -
       "create unique index ''abbreviated_table'_PRIMARY_KEY_INDEX"
$     write sql_third "    on ''table' (''primary_key')"
$     write sql_third "    type is sorted"
$     write sql_third "    node size ''node_size'"
$     write sql_third "    store in ''abbreviated_table';"
$     write sql_third " "
$     write sql_third "comment on index ''abbreviated_table'_PRIMARY_KEY_INDEX"
$     write sql_third "    is 'Primary key index for table ''table'.';"
$     write sql_third " "
$     write sql_third "create storage map ''abbreviated_table'_map for ''table'"
$     write sql_third "    store in ''abbreviated_table'"
$     write sql_third -
       "    placement via index ''abbreviated_table'_PRIMARY_KEY_INDEX"
$     write sql_third "    disable compression;"
$     write sql_third " "
$     page_size = (number_of_rows_per_page * (row_size_bytes -
       + row_overhead_bytes + 8 + primary_key_size_bytes -
       + primary_key_size_attributes + 11) + 88)/512 + 1
$     if page_size .gt. maximum_page_size then maximum_page_size = page_size
$     file_allocation_size = (11 * number_of_rows_per_table -
       / number_of_rows_per_page) / 10
$     extent_size = file_allocation_size / 10 + 1
$     write sql_second "create storage area ''abbreviated_table'"
$     write sql_second "    filename ""rdb_data:[database]''abbreviated_table'"""
$     write sql_second "    allocation is ''file_allocation_size' pages"
$     write sql_second "    page size is ''page_size'"
$     write sql_second "    page format is uniform"
$     write sql_second "    extent is ''extent_size'"
$     write sql_second -
       "    snapshot filename ""rdb_snap:[database]''abbreviated_table'.snp"""
$     write sql_second "    snapshot allocation is 10 pages"
$     write sql_second "    snapshot extent is 2 pages"
$     table = table_next
$     return
```

B. LOTUS® 1-2-3® PRINT FILES CONTAINING THE ENTITY- AND SCREEN-SORTED ATTRIBUTE DESCRIPTION LISTS

The series of blank lines which appear regularly throughout these two files are a result of these being Lotus® 1-2-3® print files; the series of blank lines correspond to page throws. The tool CREATE_SCHEMA ignores these page throws.

B.1. Entity-Sorted List

Rdb Entity-Sorted Worksheet - duplicate entries retained for cursor generation - dashes demarcate entities.

Entity	Attribute	Key	Screens	#	Domain	Domain
sample_information	.sample_tracking_id	p	m	1	identification_codes	*
sample_information	.date_sample_entered		m	2	integer_date_count	
sample_information	.date_all_work_completed		m	3	integer_date_count	Domain
sample_information	.customer_sample_id		m	4	identification_codes	character_time_hh_mm
sample_information	.results_needed_by_date		m	5	integer_date_count	charge_numbers
sample_information	.customer_project_name		m	6	forty_characters	fifty_characters
sample_information	.work_charge_number		m	7	charge_numbers	flags
sample_information	.customer_sample_name		m	8	sixty_characters	forty_characters
sample_information	.sample_type		m	9	ten_characters	identification_codes
sample_information	.sample_size		m	10	real_numbers	integer_date_count
sample_information	.sample_size_units		m	11	units	integer_numbers
sample_information	.sample_collection_date		m	12	integer_date_count	names
sample_information	.sample_collection_time		m	13	character_time_hh_mm	phone_numbers
sample_information	.any_special_instructions		m	14	flags	real_numbers
sample_information	.sample_hp_surveyed		m	15	flags	rml_spectral_id_codes
sample_information	.sample_hp_survey_activity		m	16	real_numbers	sixteen_characters
sample_information	.sample_submitter_name		m	17	names	sixty_characters
sample_information	.sample_submitter_phone_number		m	18	phone_numbers	ten_characters
sample_information	.sample_submitter_address		m	19	sixty_characters	twenty_characters
sample_information	.technical_contact_name		m	20	names	units
sample_information	.technical_contact_phone_number		m	21	phone_numbers	
sample_information	.technical_contact_address		m	22	sixty_characters	
sample_information	.send_results_to_name		m	23	names	
sample_information	.send_results_to_phone_number		m	24	phone_numbers	
sample_information	.send_results_to_address		m	25	sixty_characters	
sample_information	.sample_pickup_by_name		m	26	names	
sample_information	.sample_pickup_by_phone_number		m	27	phone_numbers	
sample_information	.sample_pickup_by_address		m	28	sixty_characters	
tracking	.possessor_name	d	m	29	names	
tracking	.possessor_phone_number	d	m	30	phone_numbers	
tracking	.possessor_address	d	m	31	sixty_characters	
sample_information	.chain_of_custody_number		m	32	ten_characters	
sample_information	.need_gross_alpha_beta		m	33	flags	
sample_information	.need_alpha		m	34	flags	
sample_information	.need_beta		m	35	flags	
sample_information	.need_gamma		m	36	flags	
sample_information	.next_special_instruction_line		m	nd	integer_numbers	
sample_information	.next_possessor_sequence_number		m	nd	integer_numbers	
sample_information	.need_gamma_screen		g	3	flags	
sample_information	.need_gamma_full_isotopic		g	11	flags	
sample_information	.need_gamma_other		g	19	flags	
sample_information	.need_gross_alpha_beta_filters		c	3	flags	
sample_information	.need_gross_alpha_beta_other		c	11	flags	
sample_information	.need_beta_strontium_90		b	3	flags	

```
sample_information          .need_beta_strontium_89_and_90      b       11 flags
sample_information          .need_beta_total_strontium          b       19 flags
sample_information          .need_beta_tritium                  b       27 flags
sample_information          .need_beta_other                    b       35 flags
sample_information          .need_alpha_uranium                 a        3 flags
sample_information          .need_alpha_thorium                 a        8 flags
sample_information          .need_alpha_plutonium               a       13 flags
sample_information          .need_alpha_am241_sans_pu238        a       18 flags
sample_information          .need_alpha_am241_with_pu238        a       23 flags
sample_information          .need_alpha_total_spectrometric     a       28 flags
sample_information          .need_alpha_other                   a       33 flags
------------------------------
alpha_uranium               .sample_tracking_id         f/p     a        1 identification_codes
```

```
sample_information          .customer_sample_name       d       a        2 sixty_characters
alpha_uranium               .results_needed_by_date             a        4 integer_date_count
alpha_uranium               .date_all_work_completed            a        5 integer_date_count
alpha_uranium               .results_report_citation            a        6 twenty_characters
alpha_uranium               .any_special_instructions           a        7 flags
------------------------------
alpha_thorium               .sample_tracking_id         f/p     a        1 identification_codes
sample_information          .customer_sample_name       d       a        2 sixty_characters
alpha_thorium               .results_needed_by_date             a        9 integer_date_count
alpha_thorium               .date_all_work_completed            a       10 integer_date_count
alpha_thorium               .results_report_citation            a       11 twenty_characters
alpha_thorium               .any_special_instructions           a       12 flags
------------------------------
alpha_plutonium             .sample_tracking_id         f/p     a        1 identification_codes
sample_information          .customer_sample_name       d       a        2 sixty_characters
alpha_plutonium             .results_needed_by_date             a       14 integer_date_count
alpha_plutonium             .date_all_work_completed            a       15 integer_date_count
alpha_plutonium             .results_report_citation            a       16 twenty_characters
alpha_plutonium             .any_special_instructions           a       17 flags
------------------------------
alpha_am241_sans_pu238      .sample_tracking_id         f/p     a        1 identification_codes
sample_information          .customer_sample_name       d       a        2 sixty_characters
alpha_am241_sans_pu238      .results_needed_by_date             a       19 integer_date_count
alpha_am241_sans_pu238      .date_all_work_completed            a       20 integer_date_count
alpha_am241_sans_pu238      .results_report_citation            a       21 twenty_characters
alpha_am241_sans_pu238      .any_special_instructions           a       22 flags
------------------------------
alpha_am241_with_pu238      .sample_tracking_id         f/p     a        1 identification_codes
sample_information          .customer_sample_name       d       a        2 sixty_characters
alpha_am241_with_pu238      .results_needed_by_date             a       24 integer_date_count
alpha_am241_with_pu238      .date_all_work_completed            a       25 integer_date_count
alpha_am241_with_pu238      .results_report_citation            a       26 twenty_characters
alpha_am241_with_pu238      .any_special_instructions           a       27 flags
------------------------------
alpha_total_spectrometric   .sample_tracking_id         f/p     a        1 identification_codes
sample_information          .customer_sample_name       d       a        2 sixty_characters
alpha_total_spectrometric   .results_needed_by_date             a       29 integer_date_count
alpha_total_spectrometric   .date_all_work_completed            a       30 integer_date_count
alpha_total_spectrometric   .results_report_citation            a       31 twenty_characters
alpha_total_spectrometric   .any_special_instructions           a       32 flags
------------------------------
alpha_other                 .sample_tracking_id         f/p     a        1 identification_codes
sample_information          .customer_sample_name       d       a        2 sixty_characters
alpha_other                 .results_needed_by_date             a       34 integer_date_count
alpha_other                 .date_all_work_completed            a       35 integer_date_count
alpha_other                 .results_report_citation            a       36 twenty_characters
alpha_other                 .any_special_instructions           a       37 flags
```

```
------------------------------
beta_strontium_90            .sample_tracking_id          f/p  b      1 identification_codes
sample_information           .customer_sample_name        d    b      2 sixty_characters
beta_strontium_90            .detector_count_length            b      4 real_numbers
beta_strontium_90            .detector_count_length_units      b      5 units
beta_strontium_90            .results_needed_by_date           b      6 integer_date_count
beta_strontium_90            .results_needed_by_time           b      7 character_time_hh_mm
beta_strontium_90            .date_all_work_completed          b      8 integer_date_count
beta_strontium_90            .results_report_citation          b      9 twenty_characters
beta_strontium_90            .any_special_instructions         b     10 flags
------------------------------
beta_strontium_89_and_90     .sample_tracking_id          f/p  b      1 identification_codes
sample_information           .customer_sample_name        d    b      2 sixty_characters



beta_strontium_89_and_90     .detector_count_length            b     12 real_numbers
beta_strontium_89_and_90     .detector_count_length_units      b     13 units
beta_strontium_89_and_90     .results_needed_by_date           b     14 integer_date_count
beta_strontium_89_and_90     .results_needed_by_time           b     15 character_time_hh_mm
beta_strontium_89_and_90     .date_all_work_completed          b     16 integer_date_count
beta_strontium_89_and_90     .results_report_citation          b     17 twenty_characters
beta_strontium_89_and_90     .any_special_instructions         b     18 flags
------------------------------
beta_total_strontium         .sample_tracking_id          f/p  b      1 identification_codes
sample_information           .customer_sample_name        d    b      2 sixty_characters
beta_total_strontium         .detector_count_length            b     20 real_numbers
beta_total_strontium         .detector_count_length_units      b     21 units
beta_total_strontium         .results_needed_by_date           b     22 integer_date_count
beta_total_strontium         .results_needed_by_time           b     23 character_time_hh_mm
beta_total_strontium         .date_all_work_completed          b     24 integer_date_count
beta_total_strontium         .results_report_citation          b     25 twenty_characters
beta_total_strontium         .any_special_instructions         b     26 flags
------------------------------
beta_tritium                 .sample_tracking_id          f/p  b      1 identification_codes
sample_information           .customer_sample_name        d    b      2 sixty_characters
beta_tritium                 .detector_count_length            b     28 real_numbers
beta_tritium                 .detector_count_length_units      b     29 units
beta_tritium                 .results_needed_by_date           b     30 integer_date_count
beta_tritium                 .results_needed_by_time           b     31 character_time_hh_mm
beta_tritium                 .date_all_work_completed          b     32 integer_date_count
beta_tritium                 .results_report_citation          b     33 twenty_characters
beta_tritium                 .any_special_instructions         b     34 flags
------------------------------
beta_other                   .sample_tracking_id          f/p  b      1 identification_codes
sample_information           .customer_sample_name        d    b      2 sixty_characters
beta_other                   .detector_count_length            b     36 real_numbers
beta_other                   .detector_count_length_units      b     37 units
beta_other                   .need_beta_nickel_63              b     38 flags
beta_other                   .need_beta_iron_55                b     39 flags
beta_other                   .need_beta_sulfur_35              b     40 flags
beta_other                   .need_beta_plutonium_241          b     41 flags
beta_other                   .results_needed_by_date           b     42 integer_date_count
beta_other                   .results_needed_by_time           b     43 character_time_hh_mm
beta_other                   .date_all_work_completed          b     44 integer_date_count
beta_other                   .results_report_citation          b     45 twenty_characters
beta_other                   .any_special_instructions         b     46 flags
------------------------------
gamma_screen                 .sample_tracking_id          f/p  g      1 identification_codes
sample_information           .customer_sample_name        d    g      2 sixty_characters
gamma_screen                 .detector_count_length            g      4 real_numbers
gamma_screen                 .detector_count_length_units      g      5 units
gamma_screen                 .results_needed_by_date           g      6 integer_date_count
```

gamma_screen	.results_needed_by_time		g	7	character_time_hh_mm
gamma_screen	.date_all_work_completed		g	8	integer_date_count
gamma_screen	.date_sample_counted		g	9	integer_date_count
gamma_screen	.rml_spectral_id		g	10	rml_spectral_id_codes
gamma_screen	.is_this_a_spectrum_recount		g	11	flags
gamma_screen	.results_report_citation		g	12	twenty_characters
gamma_screen	.any_special_instructions		g	13	flags
gamma_full_isotopic	.sample_tracking_id	f/p	g	1	identification_codes
sample_information	.customer_sample_name	d	g	2	sixty_characters
gamma_full_isotopic	.detector_count_length		g	15	real_numbers
gamma_full_isotopic	.detector_count_length_units		g	16	units
gamma_full_isotopic	.results_needed_by_date		g	17	integer_date_count
gamma_full_isotopic	.results_needed_by_time		g	18	character_time_hh_mm
gamma_full_isotopic	.date_all_work_completed		g	19	integer_date_count
gamma_full_isotopic	.date_sample_counted		g	20	integer_date_count
gamma_full_isotopic	.rml_spectral_id		g	21	rml_spectral_id_codes
gamma_full_isotopic	.is_this_a_spectrum_recount		g	22	flags
gamma_full_isotopic	.results_report_citation		g	23	twenty_characters
gamma_full_isotopic	.any_special_instructions		g	24	flags
gamma_other	.sample_tracking_id	f/p	g	1	identification_codes
sample_information	.customer_sample_name	d	g	2	sixty_characters
gamma_other	.detector_count_length		g	26	real_numbers
gamma_other	.detector_count_length_units		g	27	units
gamma_other	.results_needed_by_date		g	28	integer_date_count
gamma_other	.results_needed_by_time		g	29	character_time_hh_mm
gamma_other	.date_all_work_completed		g	30	integer_date_count
gamma_other	.date_sample_counted		g	31	integer_date_count
gamma_other	.rml_spectral_id		g	32	rml_spectral_id_codes
gamma_other	.is_this_a_spectrum_recount		g	33	flags
gamma_other	.results_report_citation		g	34	twenty_characters
gamma_other	.any_special_instructions		g	35	flags
gross_alpha_beta_air_filters	.sample_tracking_id	f/p	c	1	identification_codes
sample_information	.customer_sample_name	d	c	2	sixty_characters
gross_alpha_beta_air_filters	.detector_count_length		c	4	real_numbers
gross_alpha_beta_air_filters	.detector_count_length_units		c	5	units
gross_alpha_beta_air_filters	.results_needed_by_date		c	6	integer_date_count
gross_alpha_beta_air_filters	.results_needed_by_time		c	7	character_time_hh_mm
gross_alpha_beta_air_filters	.date_all_work_completed		c	8	integer_date_count
gross_alpha_beta_air_filters	.results_report_citation		c	9	twenty_characters
gross_alpha_beta_air_filters	.any_special_instructions		c	10	flags
gross_alpha_beta_other	.sample_tracking_id	f/p	c	1	identification_codes
sample_information	.customer_sample_name	d	c	2	sixty_characters
gross_alpha_beta_other	.detector_count_length		c	12	real_numbers
gross_alpha_beta_other	.detector_count_length_units		c	13	units
gross_alpha_beta_other	.results_needed_by_date		c	14	integer_date_count
gross_alpha_beta_other	.results_needed_by_time		c	15	character_time_hh_mm
gross_alpha_beta_other	.date_all_work_completed		c	16	integer_date_count
gross_alpha_beta_other	.results_report_citation		c	17	twenty_characters
gross_alpha_beta_other	.any_special_instructions		c	18	flags
special_instructions	.sample_tracking_id	f/p1	i	1	identification_codes
sample_information	.customer_sample_name	d	i	2	sixty_characters
special_instructions	.origin_of_special_instruction	p2	i	3	sixteen_characters
special_instructions	.special_instruction		i	4	fifty_characters
special_instructions	.special_instruction_line	p3	i	nd	integer_numbers

tracking	.sample_tracking_id	f/p1	t	1	identification_codes
sample_information	.customer_sample_name	d	t	2	sixty_characters
tracking	.possessor_name		t	3	names
tracking	.possessor_phone_number		t	4	phone_numbers
tracking	.possessor_address		t	5	sixty_characters
tracking	.possessor_possession_reason		t	6	fifty_characters
tracking	.date_accepted_by_possessor		t	7	integer_date_count
tracking	.date_relinquished_by_possessor		t	8	integer_date_count
tracking	.possessor_sequence_number	p2	t	nd	integer_numbers

B.2. Screen-Sorted List

Worksheet for Sorting Entities and Attributes and Assigning Keys and Relationships.

Base Worksheet - Sorted by Screen

Entity	Attribute	Key	Screens	#	Domain
sample_information	.sample_tracking_id	p	m	1	identification_codes
sample_information	.date_sample_entered		m	2	integer_date_count
sample_information	.date_all_work_completed		m	3	integer_date_count
sample_information	.customer_sample_id		m	4	identification_codes
sample_information	.results_needed_by_date		m	5	integer_date_count
sample_information	.customer_project_name		m	6	forty_characters
sample_information	.work_charge_number		m	7	charge_numbers
sample_information	.customer_sample_name		m	8	sixty_characters
sample_information	.sample_type		m	9	ten_characters
sample_information	.sample_size		m	10	real_numbers
sample_information	.sample_size_units		m	11	units
sample_information	.sample_collection_date		m	12	integer_date_count
sample_information	.sample_collection_time		m	13	character_time_hh_mm
sample_information	.any_special_instructions		m	14	flags
sample_information	.sample_hp_surveyed		m	15	flags
sample_information	.sample_hp_survey_activity		m	16	real_numbers
sample_information	.sample_submitter_name		m	17	names
sample_information	.sample_submitter_phone_number		m	18	phone_numbers
sample_information	.sample_submitter_address		m	19	sixty_characters
sample_information	.technical_contact_name		m	20	names
sample_information	.technical_contact_phone_number		m	21	phone_numbers
sample_information	.technical_contact_address		m	22	sixty_characters
sample_information	.send_results_to_name		m	23	names
sample_information	.send_results_to_phone_number		m	24	phone_numbers
sample_information	.send_results_to_address		m	25	sixty_characters
sample_information	.sample_pickup_by_name		m	26	names
sample_information	.sample_pickup_by_phone_number		m	27	phone_numbers
sample_information	.sample_pickup_by_address		m	28	sixty_characters
tracking	.possessor_name	d	m	29	names
tracking	.possessor_phone_number	d	m	30	phone_numbers
tracking	.possessor_address	d	m	31	sixty_characters
sample_information	.chain_of_custody_number		m	32	ten_characters
sample_information	.need_gross_alpha_beta		m	33	flags
sample_information	.need_alpha		m	34	flags
sample_information	.need_beta		m	35	flags
sample_information	.need_gamma		m	36	flags
sample_information	.next_possessor_sequence_number		m	nd	integer_numbers
sample_information	.next_special_instruction_line		m	nd	integer_numbers
alpha_uranium	.sample_tracking_id	f/p	a	1	identification_codes
sample_information	.customer_sample_name	d	a	2	sixty_characters
sample_information	.need_alpha_uranium		a	3	flags
alpha_uranium	.results_needed_by_date		a	4	integer_date_count
alpha_uranium	.date_all_work_completed		a	5	integer_date_count
alpha_uranium	.results_report_citation		a	6	twenty_characters
alpha_uranium	.any_special_instructions		a	7	flags
sample_information	.need_alpha_thorium		a	8	flags
alpha_thorium	.results_needed_by_date		a	9	integer_date_count
alpha_thorium	.date_all_work_completed		a	10	integer_date_count
alpha_thorium	.results_report_citation		a	11	twenty_characters
alpha_thorium	.any_special_instructions		a	12	flags
sample_information	.need_alpha_plutonium		a	13	flags
alpha_plutonium	.results_needed_by_date		a	14	integer_date_count
alpha_plutonium	.date_all_work_completed		a	15	integer_date_count
alpha_plutonium	.results_report_citation		a	16	twenty_characters
alpha_plutonium	.any_special_instructions		a	17	flags
sample_information	.need_alpha_am241_sans_pu238		a	18	flags
alpha_am241_sans_pu238	.results_needed_by_date		a	19	integer_date_count
alpha_am241_sans_pu238	.date_all_work_completed		a	20	integer_date_count
alpha_am241_sans_pu238	.results_report_citation		a	21	twenty_characters
alpha_am241_sans_pu238	.any_special_instructions		a	22	flags
sample_information	.need_alpha_am241_with_pu238		a	23	flags
alpha_am241_with_pu238	.results_needed_by_date		a	24	integer_date_count
alpha_am241_with_pu238	.date_all_work_completed		a	25	integer_date_count
alpha_am241_with_pu238	.results_report_citation		a	26	twenty_characters
alpha_am241_with_pu238	.any_special_instructions		a	27	flags
sample_information	.need_alpha_total_spectrometric		a	28	flags
alpha_total_spectrometric	.results_needed_by_date		a	29	integer_date_count
alpha_total_spectrometric	.date_all_work_completed		a	30	integer_date_count
alpha_total_spectrometric	.results_report_citation		a	31	twenty_characters

alpha_total_spectrometric	.any_special_instructions		a	32	flags
sample_information	.need_alpha_other		a	33	flags
alpha_other	.results_needed_by_date		a	34	integer_date_count
alpha_other	.date_all_work_completed		a	35	integer_date_count
alpha_other	.results_report_citation		a	36	twenty_characters
alpha_other	.any_special_instructions		a	37	flags
beta_strontium_90	.sample_tracking_id	f/p	b	1	identification_codes
sample_information	.customer_sample_name	d	b	2	sixty_characters
sample_information	.need_beta_strontium_90		b	3	flags
beta_strontium_90	.detector_count_length		b	4	real_numbers
beta_strontium_90	.detector_count_length_units		b	5	units
beta_strontium_90	.results_needed_by_date		b	6	integer_date_count
beta_strontium_90	.results_needed_by_time		b	7	character_time_hh_mm
beta_strontium_90	.date_all_work_completed		b	8	integer_date_count
beta_strontium_90	.results_report_citation		b	9	twenty_characters
beta_strontium_90	.any_special_instructions		b	10	flags
sample_information	.need_beta_strontium_89_and_90		b	11	flags
beta_strontium_89_and_90	.detector_count_length		b	12	real_numbers
beta_strontium_89_and_90	.detector_count_length_units		b	13	units
beta_strontium_89_and_90	.results_needed_by_date		b	14	integer_date_count
beta_strontium_89_and_90	.results_needed_by_time		b	15	character_time_hh_mm
beta_strontium_89_and_90	.date_all_work_completed		b	16	integer_date_count
beta_strontium_89_and_90	.results_report_citation		b	17	twenty_characters
beta_strontium_89_and_90	.any_special_instructions		b	18	flags
sample_information	.need_beta_total_strontium		b	19	flags
beta_total_strontium	.detector_count_length		b	20	real_numbers
beta_total_strontium	.detector_count_length_units		b	21	units
beta_total_strontium	.results_needed_by_date		b	22	integer_date_count
beta_total_strontium	.results_needed_by_time		b	23	character_time_hh_mm
beta_total_strontium	.date_all_work_completed		b	24	integer_date_count
beta_total_strontium	.results_report_citation		b	25	twenty_characters
beta_total_strontium	.any_special_instructions		b	26	flags
sample_information	.need_beta_tritium		b	27	flags
beta_tritium	.detector_count_length		b	28	real_numbers
beta_tritium	.detector_count_length_units		b	29	units
beta_tritium	.results_needed_by_date		b	30	integer_date_count
beta_tritium	.results_needed_by_time		b	31	character_time_hh_mm
beta_tritium	.date_all_work_completed		b	32	integer_date_count
beta_tritium	.results_report_citation		b	33	twenty_characters
beta_tritium	.any_special_instructions		b	34	flags
sample_information	.need_beta_other		b	35	flags
beta_other	.detector_count_length		b	36	real_numbers
beta_other	.detector_count_length_units		b	37	units
beta_other	.need_beta_nickel_63		b	38	flags
beta_other	.need_beta_iron_55		b	39	flags
beta_other	.need_beta_sulfur_35		b	40	flags
beta_other	.need_beta_plutonium_241		b	41	flags
beta_other	.results_needed_by_date		b	42	integer_date_count
beta_other	.results_needed_by_time		b	43	character_time_hh_mm
beta_other	.date_all_work_completed		b	44	integer_date_count
beta_other	.results_report_citation		b	45	twenty_characters
beta_other	.any_special_instructions		b	46	flags
gamma_screen	.sample_tracking_id	f/p	g	1	identification_codes
sample_information	.customer_sample_name	d	g	2	sixty_characters
sample_information	.need_gamma_screen		g	3	flags
gamma_screen	.detector_count_length		g	4	real_numbers
gamma_screen	.detector_count_length_units		g	5	units
gamma_screen	.results_needed_by_date		g	6	integer_date_count
gamma_screen	.results_needed_by_time		g	7	character_time_hh_mm
gamma_screen	.date_all_work_completed		g	8	integer_date_count
gamma_screen	.date_sample_counted		g	9	integer_date_count
gamma_screen	.rml_spectral_id		g	10	rml_spectral_id_codes
gamma_screen	.is_this_a_spectrum_recount		g	11	flags
gamma_screen	.results_report_citation		g	12	twenty_characters
gamma_screen	.any_special_instructions		g	13	flags
sample_information	.need_gamma_full_isotopic		g	14	flags
gamma_full_isotopic	.detector_count_length		g	15	real_numbers
gamma_full_isotopic	.detector_count_length_units		g	16	units
gamma_full_isotopic	.results_needed_by_date		g	17	integer_date_count
gamma_full_isotopic	.results_needed_by_time		g	18	character_time_hh_mm
gamma_full_isotopic	.date_all_work_completed		g	19	integer_date_count
gamma_full_isotopic	.date_sample_counted		g	20	integer_date_count
gamma_full_isotopic	.rml_spectral_id		g	21	rml_spectral_id_codes
gamma_full_isotopic	.is_this_a_spectrum_recount		g	22	flags
gamma_full_isotopic	.results_report_citation		g	23	twenty_characters
gamma_full_isotopic	.any_special_instructions		g	24	flags
sample_information	.need_gamma_other		g	25	flags

gamma_other	.detector_count_length		g	26	real_numbers
gamma_other	.detector_count_length_units		g	27	units
gamma_other	.results_needed_by_date		g	28	integer_date_count
gamma_other	.results_needed_by_time		g	29	character_time_hh_mm
gamma_other	.date_all_work_completed		g	30	integer_date_count
gamma_other	.date_sample_counted		g	31	integer_date_count
gamma_other	.rml_spectral_id		g	32	rml_spectral_id_codes
gamma_other	.is_this_a_spectrum_recount		g	33	flags
gamma_other	.results_report_citation		g	34	twenty_characters
gamma_other	.any_special_instructions		g	35	flags
gross_alpha_beta_air_filters	.sample_tracking_id	f/p	c	1	identification_codes
sample_information	.customer_sample_name	d	c	2	sixty_characters
sample_information	.need_gross_alpha_beta_filters		c	3	flags
gross_alpha_beta_air_filters	.detector_count_length		c	4	real_numbers
gross_alpha_beta_air_filters	.detector_count_length_units		c	5	units
gross_alpha_beta_air_filters	.results_needed_by_date		c	6	integer_date_count
gross_alpha_beta_air_filters	.results_needed_by_time		c	7	character_time_hh_mm
gross_alpha_beta_air_filters	.date_all_work_completed		c	8	integer_date_count
gross_alpha_beta_air_filters	.results_report_citation		c	9	twenty_characters
gross_alpha_beta_air_filters	.any_special_instructions		c	10	flags
sample_information	.need_gross_alpha_beta_other		c	11	flags
gross_alpha_beta_other	.detector_count_length		c	12	real_numbers
gross_alpha_beta_other	.detector_count_length_units		c	13	units
gross_alpha_beta_other	.results_needed_by_date		c	14	integer_date_count
gross_alpha_beta_other	.results_needed_by_time		c	15	character_time_hh_mm
gross_alpha_beta_other	.date_all_work_completed		c	16	integer_date_count
gross_alpha_beta_other	.results_report_citation		c	17	twenty_characters
gross_alpha_beta_other	.any_special_instructions		c	18	flags
special_instructions	.sample_tracking_id	f/p1	i	1	identification_codes
sample_information	.customer_sample_name	d	i	2	sixty_characters
special_instructions	.origin_of_special_instruction	p2	i	3	sixteen_characters
special_instructions	.special_instruction		i	4	fifty_characters
special_instructions	.special_instruction_line	p3	i	nd	integer_numbers
tracking	.sample_tracking_id	f/p1	t	1	identification_codes
sample_information	.customer_sample_name	d	t	2	sixty_characters
tracking	.possessor_name		t	3	names
tracking	.possessor_phone_number		t	4	phone_numbers
tracking	.possessor_address		t	5	sixty_characters
tracking	.possessor_possession_reason		t	6	fifty_characters
tracking	.date_accepted_by_possessor		t	7	integer_date_count
tracking	.date_relinquished_by_possessor		t	8	integer_date_count
tracking	.possessor_sequence_number	p2	t	nd	integer_numbers

C. EXECUTION OF CREATE_SCHEMA - A SAMPLE TERMINAL SESSION

What the user enters appears in bold text in the following.

```
$ @create_schema
Enter the name of the schema to be created:  sample_tracking
For domain CHARACTER_TIME_HH_MM:
   Enter the data type for this domain (char/date/int/real):  c
      Enter the length of the character domain:  5
   Enter the default value (return says NULL):
   Enter any desired comment:  Domain for time in the form hh:mm.
For domain CHARGE_NUMBERS:
   Enter the data type for this domain (char/date/int/real):  c
      Enter the length of the character domain:  9
   Enter the default value (return says NULL):  PF7400000
   Enter any desired comment:  Domain for nine-alphanumeric-long charge
numbers.
For domain FIFTY_CHARACTERS:
   Enter the data type for this domain (char/date/int/real):  c
      Enter the length of the character domain:  50
   Enter the default value (return says NULL):
   Enter any desired comment:  Domain for fifty-alphanumeric-long fields.
For domain FLAGS:
   Enter the data type for this domain (char/date/int/real):  c
      Enter the length of the character domain:  1
   Enter the default value (return says NULL):  N
   Enter any desired comment:  Domain for one-character-long yes/no
flags.
For domain FORTY_CHARACTERS:
   Enter the data type for this domain (char/date/int/real):  c
      Enter the length of the character domain:  40
   Enter the default value (return says NULL):
```

Enter any desired comment: **Domain for forty-alphanumeric-long fields.**

For domain IDENTIFICATION_CODES:

Enter the data type for this domain (char/date/int/real): **c**

Enter the length of the character domain: **12**

Enter the default value (return says NULL):

Enter any desired comment: **Domain for twelve-character-long identification codes.**

For domain INTEGER_DATE_COUNT:

Enter the data type for this domain (char/date/int/real): **i**

Enter the default value (return says NULL):

Enter any desired comment: **Domain for integer*4 date count starting at January 1, 1990.**

For domain INTEGER_NUMBERS:

Enter the data type for this domain (char/date/int/real): **i**

Enter the default value (return says NULL):

Enter any desired comment: **Domain for integer*4 numbers.**

For domain NAMES:

Enter the data type for this domain (char/date/int/real): **c**

Enter the length of the character domain: **25**

Enter the default value (return says NULL):

Enter any desired comment: **Domain for twenty-five-alphanumeric-long names.**

For domain PHONE_NUMBERS:

Enter the data type for this domain (char/date/int/real): **c**

Enter the length of the character domain: **12**

Enter the default value (return says NULL):

Enter any desired comment: **Domain for twelve-alphanumeric-long phone numbers.**

For domain REAL_NUMBERS:

Enter the data type for this domain (char/date/int/real): **r**

Enter the default value (return says NULL):

Enter any desired comment: **Domain for real*4 numbers.**

For domain RML_SPECTRAL_ID_CODES:

Enter the data type for this domain (char/date/int/real): **c**

Enter the length of the character domain: **14**

Enter the default value (return says NULL):

Enter any desired comment: **Domain for fourteen-character-long RML spectral identification codes (<save area>$<spectral id>).**

For domain SIXTEEN_CHARACTERS:

Enter the data type for this domain (char/date/int/real): **c**

Enter the length of the character domain: **16**

Enter the default value (return says NULL):

Enter any desired comment: **Domain for sixteen-alphanumeric-long fields.**

For domain SIXTY_CHARACTERS:

Enter the data type for this domain (char/date/int/real): **c**

Enter the length of the character domain: **60**

Enter the default value (return says NULL):

Enter any desired comment: **Domain for sixty-alphanumeric-long fields.**

For domain TEN_CHARACTERS:

Enter the data type for this domain (char/date/int/real): **c**

Enter the length of the character domain: **10**

Enter the default value (return says NULL):

Enter any desired comment: **Domain for ten-alphanumeric-long fields.**

For domain TWENTY_CHARACTERS:

Enter the data type for this domain (char/date/int/real): **c**

Enter the length of the character domain: **20**

Enter the default value (return says NULL):

Enter any desired comment: **Domain for twenty-alphanumeric-long fields.**

For domain UNITS:

Enter the data type for this domain (char/date/int/real): **c**

Enter the length of the character domain: **5**

Enter the default value (return says NULL):

Enter any desired comment: **Domain for five-alphanumeric-long units.**

For table SAMPLE_INFORMATION:

Enter any desired comment: **Table for sample information.**

Enter the number of rows per page to be clustered together: **1**

Enter the maximum number of rows for this table: **4000**

For table ALPHA_URANIUM:

Enter any desired comment: **Table for requested alpha analyses for Uranium radionuclides.**

For foreign key SAMPLE_TRACKING_ID:

Enter the reference table: **SAMPLE_INFORMATION**

Enter the reference column: **SAMPLE_TRACKING_ID**

Enter the number of rows per page to be clustered together: **1**

Enter the maximum number of rows for this table: **1000**

For table ALPHA_THORIUM:

Enter any desired comment: **Table for requested alpha analyses for Thorium radionuclides.**

For foreign key SAMPLE_TRACKING_ID:

Enter the reference table: **SAMPLE_INFORMATION**

Enter the reference column: **SAMPLE_TRACKING_ID**

Enter the number of rows per page to be clustered together: **1**

Enter the maximum number of rows for this table: **1000**

For table ALPHA_PLUTONIUM:

Enter any desired comment: **Table for requested alpha analyses for Plutonium radionuclides.**

For foreign key SAMPLE_TRACKING_ID:

Enter the reference table: **SAMPLE_INFORMATION**

Enter the reference column: **SAMPLE_TRACKING_ID**

Enter the number of rows per page to be clustered together: **1**

Enter the maximum number of rows for this table: **1000**

For table ALPHA_AM241_SANS_PU238:

Enter any desired comment: **Table for requested alpha analyses for**

Americium-241 separate from Plutonium-238.

For foreign key SAMPLE_TRACKING_ID:

Enter the reference table: **SAMPLE_INFORMATION**

Enter the reference column: **SAMPLE_TRACKING_ID**

Enter the number of rows per page to be clustered together: **1**

Enter the maximum number of rows for this table: **1000**

For table ALPHA_AM241_WITH_PU238:

Enter any desired comment: **Table for requested alpha analyses for Americium-241 combined with Plutonium-238.**

For foreign key SAMPLE_TRACKING_ID:

Enter the reference table: **SAMPLE_INFORMATION**

Enter the reference column: **SAMPLE_TRACKING_ID**

Enter the number of rows per page to be clustered together: **1**

Enter the maximum number of rows for this table: **1000**

For table ALPHA_TOTAL_SPECTROMETRIC:

Enter any desired comment: **Table for requested total spectrometric alpha analyses.**

For foreign key SAMPLE_TRACKING_ID:

Enter the reference table: **SAMPLE_INFORMATION**

Enter the reference column: **SAMPLE_TRACKING_ID**

Enter the number of rows per page to be clustered together: **1**

Enter the maximum number of rows for this table: **1000**

For table ALPHA_OTHER:

Enter any desired comment: **Table for requested alpha analyses for other, unlisted radionuclides.**

For foreign key SAMPLE_TRACKING_ID:

Enter the reference table: **SAMPLE_INFORMATION**

Enter the reference column: **SAMPLE_TRACKING_ID**

Enter the number of rows per page to be clustered together: **1**

Enter the maximum number of rows for this table: **1000**

For table BETA_STRONTIUM_90:

Enter any desired comment: **Table for requested beta analyses for**

Strontium-90.

For foreign key SAMPLE_TRACKING_ID:

Enter the reference table: **SAMPLE_INFORMATION**

Enter the reference column: **SAMPLE_TRACKING_ID**

Enter the number of rows per page to be clustered together: **1**

Enter the maximum number of rows for this table: **1000**

For table BETA_STRONTIUM_89_AND_90:

Enter any desired comment: **Table for requested beta analyses for Strontium-89 and -90.**

For foreign key SAMPLE_TRACKING_ID:

Enter the reference table: **SAMPLE_INFORMATION**

Enter the reference column: **SAMPLE_TRACKING_ID**

Enter the number of rows per page to be clustered together: **1**

Enter the maximum number of rows for this table: **1000**

For table BETA_TOTAL_STRONTIUM:

Enter any desired comment: **Table for requested beta analyses for total Strontium.**

For foreign key SAMPLE_TRACKING_ID:

Enter the reference table: **SAMPLE_INFORMATION**

Enter the reference column: **SAMPLE_TRACKING_ID**

Enter the number of rows per page to be clustered together: **1**

Enter the maximum number of rows for this table: **1000**

For table BETA_TRITIUM:

Enter any desired comment: **Table for requested beta analyses for Tritium.**

For foreign key SAMPLE_TRACKING_ID:

Enter the reference table: **SAMPLE_INFORMATION**

Enter the reference column: **SAMPLE_TRACKING_ID**

Enter the number of rows per page to be clustered together: **1**

Enter the maximum number of rows for this table: **1000**

For table BETA_OTHER:

Enter any desired comment: **Table for requested beta analyses for**

other, unlisted radionuclides.

For foreign key SAMPLE_TRACKING_ID:

Enter the reference table: **SAMPLE_INFORMATION**

Enter the reference column: **SAMPLE_TRACKING_ID**

Enter the number of rows per page to be clustered together: **1**

Enter the maximum number of rows for this table: **1000**

For table GAMMA_SCREEN:

Enter any desired comment: **Table for requested screening and shipping gamma analyses.**

For foreign key SAMPLE_TRACKING_ID:

Enter the reference table: **SAMPLE_INFORMATION**

Enter the reference column: **SAMPLE_TRACKING_ID**

Enter the number of rows per page to be clustered together: **1**

Enter the maximum number of rows for this table: **4000**

For table GAMMA_FULL_ISOTOPIC:

Enter any desired comment: **Table for requested full isotopic gamma analyses.**

For foreign key SAMPLE_TRACKING_ID:

Enter the reference table: **SAMPLE_INFORMATION**

Enter the reference column: **SAMPLE_TRACKING_ID**

Enter the number of rows per page to be clustered together: **1**

Enter the maximum number of rows for this table: **4000**

For table GAMMA_OTHER:

Enter any desired comment: **Table for requested gamma analyses for other, unlisted radionuclides.**

For foreign key SAMPLE_TRACKING_ID:

Enter the reference table: **SAMPLE_INFORMATION**

Enter the reference column: **SAMPLE_TRACKING_ID**

Enter the number of rows per page to be clustered together: **1**

Enter the maximum number of rows for this table: **4000**

For table GROSS_ALPHA_BETA_AIR_FILTERS:

Enter any desired comment: **Table for requested gross alpha-beta**

analyses of air filters.

For foreign key SAMPLE_TRACKING_ID:

Enter the reference table: **SAMPLE_INFORMATION**

Enter the reference column: **SAMPLE_TRACKING_ID**

Enter the number of rows per page to be clustered together: **1**

Enter the maximum number of rows for this table: **1000**

For table GROSS_ALPHA_BETA_OTHER:

Enter any desired comment: **Table for requested gross alpha-beta analyses of other, unlisted samples.**

For foreign key SAMPLE_TRACKING_ID:

Enter the reference table: **SAMPLE_INFORMATION**

Enter the reference column: **SAMPLE_TRACKING_ID**

Enter the number of rows per page to be clustered together: **1**

Enter the maximum number of rows for this table: **1000**

For table SPECIAL_INSTRUCTIONS:

Enter any desired comment: **Table for special instructions.**

For foreign key SAMPLE_TRACKING_ID:

Enter the reference table: **SAMPLE_INFORMATION**

Enter the reference column: **SAMPLE_TRACKING_ID**

Enter the number of rows per page to be clustered together: **15**

Enter the maximum number of rows for this table: **20000**

For table TRACKING:

Enter any desired comment: **Table for tracking samples.**

For foreign key SAMPLE_TRACKING_ID:

Enter the reference table: **SAMPLE_INFORMATION**

Enter the reference column: **SAMPLE_TRACKING_ID**

Enter the number of rows per page to be clustered together: **5**

Enter the maximum number of rows for this table: **20000**

Execution has successfully completed. The create_schema.sql and cursor and insert files will be closed and retained. The entity and screen files will also be closed and retained.

Please wait while certain sections of the cursor and insert files are

```
rearranged; this may take a few minutes.
   Cursor file rearranged.
   Insert file rearranged.
$
```

D. SQL INSTRUCTION SET FOR CREATION OF SAMPLE TRACKING DATABASE

Note the ellipses in the two "define/system" lines below; these would be replaced with the appropriate root directory specifier ($USER:[DATABASES.], for example).

```
! SQL commands for creating sample_tracking schema.
!
! Define the data and snapshot file location logical names.
!
$ define/system/nolog/translation_attributes=concealed rdb_data . . .
$ define/system/nolog/translation_attributes=concealed rdb_snap . . .
!
! Create the schema.
!
create schema
   filename "sample_tracking"
   buffer size is 18

!
! Create the storage areas.
!
create storage area SAIN
   filename "rdb_data:[sample_tracking]SAIN"
   allocation is 4400 pages
   page size is 2
   page format is uniform
   extent is 441
   snapshot filename "rdb_snap:[sample_tracking]SAIN.snp"
   snapshot allocation is 10 pages
   snapshot extent is 2 pages

create storage area ALUR
   filename "rdb_data:[sample_tracking]ALUR"
   allocation is 1100 pages
   page size is 1
   page format is uniform
   extent is 111
   snapshot filename "rdb_snap:[sample_tracking]ALUR.snp"
   snapshot allocation is 10 pages
   snapshot extent is 2 pages

create storage area ALTH
   filename "rdb_data:[sample_tracking]ALTH"
   allocation is 1100 pages
   page size is 1
   page format is uniform
   extent is 111
   snapshot filename "rdb_snap:[sample_tracking]ALTH.snp"
   snapshot allocation is 10 pages
   snapshot extent is 2 pages

create storage area ALPL
   filename "rdb_data:[sample_tracking]ALPL"
   allocation is 1100 pages
   page size is 1
   page format is uniform
   extent is 111
```

```
    snapshot filename "rdb_snap:[sample_tracking]ALPL.snp"
    snapshot allocation is 10 pages
    snapshot extent is 2 pages

create storage area ALAMSAPU
    filename "rdb_data:[sample_tracking]ALAMSAPU"
    allocation is 1100 pages
    page size is 1
    page format is uniform
    extent is 111
    snapshot filename "rdb_snap:[sample_tracking]ALAMSAPU.snp"
    snapshot allocation is 10 pages
    snapshot extent is 2 pages

create storage area ALAMWIPU
    filename "rdb_data:[sample_tracking]ALAMWIPU"
    allocation is 1100 pages
    page size is 1
    page format is uniform
    extent is 111
    snapshot filename "rdb_snap:[sample_tracking]ALAMWIPU.snp"
    snapshot allocation is 10 pages
    snapshot extent is 2 pages

create storage area ALTOSP
    filename "rdb_data:[sample_tracking]ALTOSP"
    allocation is 1100 pages
    page size is 1
    page format is uniform
    extent is 111
    snapshot filename "rdb_snap:[sample_tracking]ALTOSP.snp"
    snapshot allocation is 10 pages
    snapshot extent is 2 pages

create storage area ALOT
    filename "rdb_data:[sample_tracking]ALOT"
    allocation is 1100 pages
    page size is 1
    page format is uniform
    extent is 111
    snapshot filename "rdb_snap:[sample_tracking]ALOT.snp"
    snapshot allocation is 10 pages
    snapshot extent is 2 pages

create storage area BEST90
    filename "rdb_data:[sample_tracking]BEST90"
    allocation is 1100 pages
    page size is 1
    page format is uniform
    extent is 111
    snapshot filename "rdb_snap:[sample_tracking]BEST90.snp"
    snapshot allocation is 10 pages
    snapshot extent is 2 pages

create storage area BEST89AN90
    filename "rdb_data:[sample_tracking]BES 89AN90"
    allocation is 1100 pages
    page size is 1
    page format is uniform
    extent is 111
```

```
    snapshot filename "rdb_snap:[sample_tracking]BEST89AN90.snp"
    snapshot allocation is 10 pages
    snapshot extent is 2 pages

create storage area BETOST
    filename "rdb_data:[sample_tracking]BETOST"
    allocation is 1100 pages
    page size is 1
    page format is uniform
    extent is 111
    snapshot filename "rdb_snap:[sample_tracking]BETOST.snp"
    snapshot allocation is 10 pages
    snapshot extent is 2 pages

create storage area BETR
    filename "rdb_data:[sample_tracking]BETR"
    allocation is 1100 pages
    page size is 1
    page format is uniform
    extent is 111
    snapshot filename "rdb_snap:[sample_tracking]BETR.snp"
    snapshot allocation is 10 pages
    snapshot extent is 2 pages

create storage area BEOT
    filename "rdb_data:[sample_tracking]BEOT"
    allocation is 1100 pages
    page size is 1
    page format is uniform
    extent is 111
    snapshot filename "rdb_snap:[sample_tracking]BEOT.snp"
    snapshot allocation is 10 pages
    snapshot extent is 2 pages

create storage area GASC
    filename "rdb_data:[sample_tracking]GASC"
    allocation is 4400 pages
    page size is 1
    page format is uniform
    extent is 441
    snapshot filename "rdb_snap:[sample_tracking]GASC.snp"
    snapshot allocation is 10 pages
    snapshot extent is 2 pages

create storage area GAFUIS
    filename "rdb_data:[sample_tracking]GAFUIS"
    allocation is 4400 pages
    page size is 1
    page format is uniform
    extent is 441
    snapshot filename "rdb_snap:[sample_tracking]GAFUIS.snp"
    snapshot allocation is 10 pages
    snapshot extent is 2 pages

create storage area GAOT
    filename "rdb_data:[sample_tracking]GAOT"
    allocation is 4400 pages
    page size is 1
    page format is uniform
    extent is 441
```

```
    snapshot filename "rdb_snap:[sample_tracking]GAOT.snp"
    snapshot allocation is 10 pages
    snapshot extent is 2 pages

create storage area GRALBEAIFI
    filename "rdb_data:[sample_tracking]GRALBEAIFI"
    allocation is 1100 pages
    page size is 1
    page format is uniform
    extent is 111
    snapshot filename "rdb_snap:[sample_tracking]GRALBEAIFI.snp"
    snapshot allocation is 10 pages
    snapshot extent is 2 pages

create storage area GRALBEOT
    filename "rdb_data:[sample_tracking]GRALBEOT"
    allocation is 1100 pages
    page size is 1
    page format is uniform
    extent is 111
    snapshot filename "rdb_snap:[sample_tracking]GRALBEOT.snp"
    snapshot allocation is 10 pages
    snapshot extent is 2 pages

create storage area SPIN
    filename "rdb_data:[sample_tracking]SPIN"
    allocation is 1466 pages
    page size is 6
    page format is uniform
    extent is 147
    snapshot filename "rdb_snap:[sample_tracking]SPIN.snp"
    snapshot allocation is 10 pages
    snapshot extent is 2 pages

create storage area TR
    filename "rdb_data:[sample_tracking]TR"
    allocation is 4400 pages
    page size is 3
    page format is uniform
    extent is 441
    snapshot filename "rdb_snap:[sample_tracking]TR.snp"
    snapshot allocation is 10 pages
    snapshot extent is 2 pages
    ;

!
! Create the domains, including defaults, constraints, and comments.
!
create domain USER_NAME char(30)
    default NULL;

comment on domain USER_NAME
    is 'Domain for user name.';

create domain VMS_DATE_TIME char(18)
    default NULL;

comment on domain VMS_DATE_TIME
    is 'Domain for VMS date (dd-mmm-yy) and time (hh:mm:ss).';
```

```
create domain CHARACTER_TIME_HH_MM char(5)
   default NULL;

comment on domain CHARACTER_TIME_HH_MM
   is 'Domain for time in the form hh:mm.';

create domain CHARGE_NUMBERS char(9)
   default 'PF7400000';

comment on domain CHARGE_NUMBERS
   is 'Domain for nine-alphanumeric-long charge numbers.';

create domain FIFTY_CHARACTERS char(50)
   default NULL;

comment on domain FIFTY_CHARACTERS
   is 'Domain for fifty-alphanumeric-long fields.';

create domain FLAGS char(1)
   default 'N';

comment on domain FLAGS
   is 'Domain for one-character-long yes/no flags.';

create domain FORTY_CHARACTERS char(40)
   default NULL;

comment on domain FORTY_CHARACTERS
   is 'Domain for forty-alphanumeric-long fields.';

create domain IDENTIFICATION_CODES char(12)
   default NULL;

comment on domain IDENTIFICATION_CODES
   is 'Domain for twelve-character-long identification codes.';

create domain INTEGER_DATE_COUNT integer
   default NULL;

comment on domain INTEGER_DATE_COUNT
   is 'Domain for integer*4 date count starting at January 1, 1990.';

create domain INTEGER_NUMBERS integer
   default NULL;

comment on domain INTEGER_NUMBERS
   is 'Domain for integer*4 numbers.';

create domain NAMES char(25)
   default NULL;

comment on domain NAMES
   is 'Domain for twenty-five-alphanumeric-long names.';

create domain PHONE_NUMBERS char(12)
   default NULL;

comment on domain PHONE_NUMBERS
   is 'Domain for twelve-alphanumeric-long phone numbers.';
```

```
create domain REAL_NUMBERS real
   default NULL;

comment on domain REAL_NUMBERS
   is 'Domain for real*4 numbers.';

create domain RML_SPECTRAL_ID_CODES char(14)
   default NULL;

comment on domain RML_SPECTRAL_ID_CODES
   is 'Domain for fourteen-character-long RML spectral identification
codes (<save area>$<spectral id>).';

create domain SIXTEEN_CHARACTERS char(16)
   default NULL;

comment on domain SIXTEEN_CHARACTERS
   is 'Domain for sixteen-alphanumeric-long fields.';

create domain SIXTY_CHARACTERS char(60)
   default NULL;

comment on domain SIXTY_CHARACTERS
   is 'Domain for sixty-alphanumeric-long fields.';

create domain TEN_CHARACTERS char(10)
   default NULL;

comment on domain TEN_CHARACTERS
   is 'Domain for ten-alphanumeric-long fields.';

create domain TWENTY_CHARACTERS char(20)
   default NULL;

comment on domain TWENTY_CHARACTERS
   is 'Domain for twenty-alphanumeric-long fields.';

create domain UNITS char(5)
   default NULL;

comment on domain UNITS
   is 'Domain for five-alphanumeric-long units.';

!
! Create tables.
!
create table SAMPLE_INFORMATION (
      SAMPLE_TRACKING_ID              IDENTIFICATION_CODES      not null,
      DATE_SAMPLE_ENTERED              INTEGER_DATE_COUNT,
      DATE_ALL_WORK_COMPLETED          INTEGER_DATE_COUNT,
      CUSTOMER_SAMPLE_ID               IDENTIFICATION_CODES,
      RESULTS_NEEDED_BY_DATE           INTEGER_DATE_COUNT,
      CUSTOMER_PROJECT_NAME            FORTY_CHARACTERS,
      WORK_CHARGE_NUMBER               CHARGE_NUMBERS,
      CUSTOMER_SAMPLE_NAME             SIXTY_CHARACTERS,
      SAMPLE_TYPE                      TEN_CHARACTERS,
      SAMPLE_SIZE                      REAL_NUMBERS,
      SAMPLE_SIZE_UNITS                UNITS,
      SAMPLE_COLLECTION_DATE           INTEGER_DATE_COUNT,
      SAMPLE_COLLECTION_TIME           CHARACTER_TIME_HH_MM,
```

```
      ANY_SPECIAL_INSTRUCTIONS          FLAGS,
      SAMPLE_HP_SURVEYED                FLAGS,
      SAMPLE_HP_SURVEY_ACTIVITY         REAL_NUMBERS,
      SAMPLE_SUBMITTER_NAME             NAMES,
      SAMPLE_SUBMITTER_PHONE_NUMBER     PHONE_NUMBERS,
      SAMPLE_SUBMITTER_ADDRESS          SIXTY_CHARACTERS,
      TECHNICAL_CONTACT_NAME            NAMES,
      TECHNICAL_CONTACT_PHONE_NUMBER    PHONE_NUMBERS,
      TECHNICAL_CONTACT_ADDRESS         SIXTY_CHARACTERS,
      SEND_RESULTS_TO_NAME              NAMES,
      SEND_RESULTS_TO_PHONE_NUMBER      PHONE_NUMBERS,
      SEND_RESULTS_TO_ADDRESS           SIXTY_CHARACTERS,
      SAMPLE_PICKUP_BY_NAME             NAMES,
      SAMPLE_PICKUP_BY_PHONE_NUMBER     PHONE_NUMBERS,
      SAMPLE_PICKUP_BY_ADDRESS          SIXTY_CHARACTERS,
      NEED_GROSS_ALPHA_BETA             FLAGS,
      NEED_ALPHA                        FLAGS,
      NEED_BETA                         FLAGS,
      NEED_GAMMA                        FLAGS,
      NEXT_SPECIAL_INSTRUCTION_LINE     INTEGER_NUMBERS,
      NEXT_POSSESSOR_SEQUENCE_NUMBER    INTEGER_NUMBERS,
      NEED_GAMMA_SCREEN                 FLAGS,
      NEED_GAMMA_FULL_ISOTOPIC          FLAGS,
      NEED_GAMMA_OTHER                  FLAGS,
      NEED_GROSS_ALPHA_BETA_FILTERS     FLAGS,
      NEED_GROSS_ALPHA_BETA_OTHER       FLAGS,
      NEED_BETA_STRONTIUM_90            FLAGS,
      NEED_BETA_STRONTIUM_89_AND_90     FLAGS,
      NEED_BETA_TOTAL_STRONTIUM         FLAGS,
      NEED_BETA_TRITIUM                 FLAGS,
      NEED_BETA_OTHER                   FLAGS,
      NEED_ALPHA_URANIUM                FLAGS,
      NEED_ALPHA_THORIUM                FLAGS,
      NEED_ALPHA_PLUTONIUM              FLAGS,
      NEED_ALPHA_AM241_SANS_PU238       FLAGS,
      NEED_ALPHA_AM241_WITH_PU238       FLAGS,
      NEED_ALPHA_TOTAL_SPECTROMETRIC    FLAGS,
      NEED_ALPHA_OTHER                  FLAGS,
      USER_CREATING_OR_MODIFYING        USER_NAME,
      DATE_CREATED_OR_MODIFIED          VMS_DATE_TIME,
      CHAIN_OF_CUSTODY_NUMBER           TEN_CHARACTERS,
   primary key (SAMPLE_TRACKING_ID) );

comment on table SAMPLE_INFORMATION
   is 'Table for sample information.';

create unique index SAIN_PRIMARY_KEY_INDEX
   on SAMPLE_INFORMATION (SAMPLE_TRACKING_ID)
   type is sorted
   node size 104
   store in SAIN;

comment on index SAIN_PRIMARY_KEY_INDEX
   is 'Primary key index for table SAMPLE_INFORMATION.';

create storage map SAIN_map for SAMPLE_INFORMATION
   store in SAIN
   placement via index SAIN_PRIMARY_KEY_INDEX
   disable compression;
```

```
create table ALPHA_URANIUM (
      SAMPLE_TRACKING_ID              IDENTIFICATION_CODES      not null,
      RESULTS_NEEDED_BY_DATE           INTEGER_DATE_COUNT,
      DATE_ALL_WORK_COMPLETED          INTEGER_DATE_COUNT,
      RESULTS_REPORT_CITATION          TWENTY_CHARACTERS,
      ANY_SPECIAL_INSTRUCTIONS         FLAGS,
      USER_CREATING_OR_MODIFYING       USER_NAME,
      DATE_CREATED_OR_MODIFIED         VMS_DATE_TIME,
   primary key (SAMPLE_TRACKING_ID),
   foreign key (SAMPLE_TRACKING_ID)
      references SAMPLE_INFORMATION (SAMPLE_TRACKING_ID) );

comment on table ALPHA_URANIUM
   is 'Table for requested alpha analyses for Uranium radionuclides.';

create unique index ALUR_PRIMARY_KEY_INDEX
   on ALPHA_URANIUM (SAMPLE_TRACKING_ID)
   type is sorted
   node size 104
   store in ALUR;

comment on index ALUR_PRIMARY_KEY_INDEX
   is 'Primary key index for table ALPHA_URANIUM.';

create storage map ALUR_map for ALPHA_URANIUM
   store in ALUR
   placement via index ALUR_PRIMARY_KEY_INDEX
   disable compression;

create table ALPHA_THORIUM (
      SAMPLE_TRACKING_ID              IDENTIFICATION_CODES      not null,
      RESULTS_NEEDED_BY_DATE           INTEGER_DATE_COUNT,
      DATE_ALL_WORK_COMPLETED          INTEGER_DATE_COUNT,
      RESULTS_REPORT_CITATION          TWENTY_CHARACTERS,
      ANY_SPECIAL_INSTRUCTIONS         FLAGS,
      USER_CREATING_OR_MODIFYING       USER_NAME,
      DATE_CREATED_OR_MODIFIED         VMS_DATE_TIME,
   primary key (SAMPLE_TRACKING_ID),
   foreign key (SAMPLE_TRACKING_ID)
      references SAMPLE_INFORMATION (SAMPLE_TRACKING_ID) );

comment on table ALPHA_THORIUM
   is 'Table for requested alpha analyses for Thorium radionuclides.';

create unique index ALTH_PRIMARY_KEY_INDEX
   on ALPHA_THORIUM (SAMPLE_TRACKING_ID)
   type is sorted
   node size 104
   store in ALTH;

comment on index ALTH_PRIMARY_KEY_INDEX
   is 'Primary key index for table ALPHA_THORIUM.';

create storage map ALTH_map for ALPHA_THORIUM
   store in ALTH
   placement via index ALTH_PRIMARY_KEY_INDEX
   disable compression;

create table ALPHA_PLUTONIUM (
      SAMPLE_TRACKING_ID              IDENTIFICATION_CODES      not null,
```

```
      RESULTS_NEEDED_BY_DATE          INTEGER_DATE_COUNT,
      DATE_ALL_WORK_COMPLETED         INTEGER_DATE_COUNT,
      RESULTS_REPORT_CITATION         TWENTY_CHARACTERS,
      ANY_SPECIAL_INSTRUCTIONS        FLAGS,
      USER_CREATING_OR_MODIFYING      USER_NAME,
      DATE_CREATED_OR_MODIFIED        VMS_DATE_TIME,
   primary key (SAMPLE_TRACKING_ID),
   foreign key (SAMPLE_TRACKING_ID)
      references SAMPLE_INFORMATION (SAMPLE_TRACKING_ID) );

comment on table ALPHA_PLUTONIUM
   is 'Table for requested alpha analyses for Plutonium radionuclides.';

create unique index ALPL_PRIMARY_KEY_INDEX
   on ALPHA_PLUTONIUM (SAMPLE_TRACKING_ID)
   type is sorted
   node size 104
   store in ALPL;

comment on index ALPL_PRIMARY_KEY_INDEX
   is 'Primary key index for table ALPHA_PLUTONIUM.';

create storage map ALPL_map for ALPHA_PLUTONIUM
   store in ALPL
   placement via index ALPL_PRIMARY_KEY_INDEX
   disable compression;

create table ALPHA_AM241_SANS_PU238 (
      SAMPLE_TRACKING_ID              IDENTIFICATION_CODES      not null,
      RESULTS_NEEDED_BY_DATE          INTEGER_DATE_COUNT,
      DATE_ALL_WORK_COMPLETED         INTEGER_DATE_COUNT,
      RESULTS_REPORT_CITATION         TWENTY_CHARACTERS,
      ANY_SPECIAL_INSTRUCTIONS        FLAGS,
      USER_CREATING_OR_MODIFYING      USER_NAME,
      DATE_CREATED_OR_MODIFIED        VMS_DATE_TIME,
   primary key (SAMPLE_TRACKING_ID),
   foreign key (SAMPLE_TRACKING_ID)
      references SAMPLE_INFORMATION (SAMPLE_TRACKING_ID) );

comment on table ALPHA_AM241_SANS_PU238
   is 'Table for requested alpha analyses for Americium-241 separate from
Plutonium-238.';

create unique index ALAMSAPU_PRIMARY_KEY_INDEX
   on ALPHA_AM241_SANS_PU238 (SAMPLE_TRACKING_ID)
   type is sorted
   node size 104
   store in ALAMSAPU;

comment on index ALAMSAPU_PRIMARY_KEY_INDEX
   is 'Primary key index for table ALPHA_AM241_SANS_PU238.';

create storage map ALAMSAPU_map for ALPHA_AM241_SANS_PU238
   store in ALAMSAPU
   placement via index ALAMSAPU_PRIMARY_KEY_INDEX
   disable compression;

create table ALPHA_AM241_WITH_PU238 (
      SAMPLE_TRACKING_ID              IDENTIFICATION_CODES      not null,
      RESULTS_NEEDED_BY_DATE          INTEGER_DATE_COUNT,
```

```
      DATE_ALL_WORK_COMPLETED          INTEGER_DATE_COUNT,
      RESULTS_REPORT_CITATION          TWENTY_CHARACTERS,
      ANY_SPECIAL_INSTRUCTIONS         FLAGS,
      USER_CREATING_OR_MODIFYING       USER_NAME,
      DATE_CREATED_OR_MODIFIED         VMS_DATE_TIME,
   primary key (SAMPLE_TRACKING_ID),
   foreign key (SAMPLE_TRACKING_ID)
      references SAMPLE_INFORMATION (SAMPLE_TRACKING_ID) );

comment on table ALPHA_AM241_WITH_PU238
   is 'Table for requested alpha analyses for Americium-241 combined with
Plutonium-238.';

create unique index ALAMWIPU_PRIMARY_KEY_INDEX
   on ALPHA_AM241_WITH_PU238 (SAMPLE_TRACKING_ID)
   type is sorted
   node size 104
   store in ALAMWIPU;

comment on index ALAMWIPU_PRIMARY_KEY_INDEX
   is 'Primary key index for table ALPHA_AM241_WITH_PU238.';

create storage map ALAMWIPU_map for ALPHA_AM241_WITH_PU238
   store in ALAMWIPU
   placement via index ALAMWIPU_PRIMARY_KEY_INDEX
   disable compression;

create table ALPHA_TOTAL_SPECTROMETRIC (
      SAMPLE_TRACKING_ID               IDENTIFICATION_CODES      not null,
      RESULTS_NEEDED_BY_DATE           INTEGER_DATE_COUNT,
      DATE_ALL_WORK_COMPLETED          INTEGER_DATE_COUNT,
      RESULTS_REPORT_CITATION          TWENTY_CHARACTERS,
      ANY_SPECIAL_INSTRUCTIONS         FLAGS,
      USER_CREATING_OR_MODIFYING       USER_NAME,
      DATE_CREATED_OR_MODIFIED         VMS_DATE_TIME,
   primary key (SAMPLE_TRACKING_ID),
   foreign key (SAMPLE_TRACKING_ID)
      references SAMPLE_INFORMATION (SAMPLE_TRACKING_ID) );

comment on table ALPHA_TOTAL_SPECTROMETRIC
   is 'Table for requested total spectrometric alpha analyses.';

create unique index ALTOSP_PRIMARY_KEY_INDEX
   on ALPHA_TOTAL_SPECTROMETRIC (SAMPLE_TRACKING_ID)
   type is sorted
   node size 104
   store in ALTOSP;

comment on index ALTOSP_PRIMARY_KEY_INDEX
   is 'Primary key index for table ALPHA_TOTAL_SPECTROMETRIC.';

create storage map ALTOSP_map for ALPHA_TOTAL_SPECTROMETRIC
   store in ALTOSP
   placement via index ALTOSP_PRIMARY_KEY_INDEX
   disable compression;

create table ALPHA_OTHER (
      SAMPLE_TRACKING_ID               IDENTIFICATION_CODES      not null,
      RESULTS_NEEDED_BY_DATE           INTEGER_DATE_COUNT,
      DATE_ALL_WORK_COMPLETED          INTEGER_DATE_COUNT,
```

```
      RESULTS_REPORT_CITATION          TWENTY_CHARACTERS,
      ANY_SPECIAL_INSTRUCTIONS         FLAGS,
      USER_CREATING_OR_MODIFYING       USER_NAME,
      DATE_CREATED_OR_MODIFIED         VMS_DATE_TIME,
   primary key (SAMPLE_TRACKING_ID),
   foreign key (SAMPLE_TRACKING_ID)
      references SAMPLE_INFORMATION (SAMPLE_TRACKING_ID) );

comment on table ALPHA_OTHER
   is 'Table for requested alpha analyses for other, unlisted
radionuclides.';

create unique index ALOT_PRIMARY_KEY_INDEX
   on ALPHA_OTHER (SAMPLE_TRACKING_ID)
   type is sorted
   node size 104
   store in ALOT;

comment on index ALOT_PRIMARY_KEY_INDEX
   is 'Primary key index for table ALPHA_OTHER.';

create storage map ALOT_map for ALPHA_OTHER
   store in ALOT
   placement via index ALOT_PRIMARY_KEY_INDEX
   disable compression;

create table BETA_STRONTIUM_90 (
      SAMPLE_TRACKING_ID              IDENTIFICATION_CODES      not null,
      DETECTOR_COUNT_LENGTH            REAL_NUMBERS,
      DETECTOR_COUNT_LENGTH_UNITS      UNITS,
      RESULTS_NEEDED_BY_DATE           INTEGER_DATE_COUNT,
      RESULTS_NEEDED_BY_TIME           CHARACTER_TIME_HH_MM,
      DATE_ALL_WORK_COMPLETED          INTEGER_DATE_COUNT,
      RESULTS_REPORT_CITATION          TWENTY_CHARACTERS,
      ANY_SPECIAL_INSTRUCTIONS         FLAGS,
      USER_CREATING_OR_MODIFYING       USER_NAME,
      DATE_CREATED_OR_MODIFIED         VMS_DATE_TIME,
   primary key (SAMPLE_TRACKING_ID),
   foreign key (SAMPLE_TRACKING_ID)
      references SAMPLE_INFORMATION (SAMPLE_TRACKING_ID) );

comment on table BETA_STRONTIUM_90
   is 'Table for requested beta analyses for Strontium-90.';

create unique index BEST90_PRIMARY_KEY_INDEX
   on BETA_STRONTIUM_90 (SAMPLE_TRACKING_ID)
   type is sorted
   node size 104
   store in BEST90;

comment on index BEST90_PRIMARY_KEY_INDEX
   is 'Primary key index for table BETA_STRONTIUM_90.';

create storage map BEST90_map for BETA_STRONTIUM_90
   store in BEST90
   placement via index BEST90_PRIMARY_KEY_INDEX
   disable compression;

create table BETA_STRONTIUM_89_AND_90 (
      SAMPLE_TRACKING_ID              IDENTIFICATION_CODES      not null,
```

```
      DETECTOR_COUNT_LENGTH            REAL_NUMBERS,
      DETECTOR_COUNT_LENGTH_UNITS      UNITS,
      RESULTS_NEEDED_BY_DATE           INTEGER_DATE_COUNT,
      RESULTS_NEEDED_BY_TIME           CHARACTER_TIME_HH_MM,
      DATE_ALL_WORK_COMPLETED          INTEGER_DATE_COUNT,
      RESULTS_REPORT_CITATION          TWENTY_CHARACTERS,
      ANY_SPECIAL_INSTRUCTIONS         FLAGS,
      USER_CREATING_OR_MODIFYING       USER_NAME,
      DATE_CREATED_OR_MODIFIED         VMS_DATE_TIME,
   primary key (SAMPLE_TRACKING_ID),
   foreign key (SAMPLE_TRACKING_ID)
      references SAMPLE_INFORMATION (SAMPLE_TRACKING_ID) );

comment on table BETA_STRONTIUM_89_AND_90
   is 'Table for requested beta analyses for Strontium-89 and -90.';

create unique index BEST89AN90_PRIMARY_KEY_INDEX
   on BETA_STRONTIUM_89_AND_90 (SAMPLE_TRACKING_ID)
   type is sorted
   node size 104
   store in BEST89AN90;

comment on index BEST89AN90_PRIMARY_KEY_INDEX
   is 'Primary key index for table BETA_STRONTIUM_89_AND_90.';

create storage map BEST89AN90_map for BETA_STRONTIUM_89_AND_90
   store in BEST89AN90
   placement via index BEST89AN90_PRIMARY_KEY_INDEX
   disable compression;

create table BETA_TOTAL_STRONTIUM (
      SAMPLE_TRACKING_ID               IDENTIFICATION_CODES      not null,
      DETECTOR_COUNT_LENGTH            REAL_NUMBERS,
      DETECTOR_COUNT_LENGTH_UNITS      UNITS,
      RESULTS_NEEDED_BY_DATE           INTEGER_DATE_COUNT,
      RESULTS_NEEDED_BY_TIME           CHARACTER_TIME_HH_MM,
      DATE_ALL_WORK_COMPLETED          INTEGER_DATE_COUNT,
      RESULTS_REPORT_CITATION          TWENTY_CHARACTERS,
      ANY_SPECIAL_INSTRUCTIONS         FLAGS,
      USER_CREATING_OR_MODIFYING       USER_NAME,
      DATE_CREATED_OR_MODIFIED         VMS_DATE_TIME,
   primary key (SAMPLE_TRACKING_ID),
   foreign key (SAMPLE_TRACKING_ID)
      references SAMPLE_INFORMATION (SAMPLE_TRACKING_ID) );

comment on table BETA_TOTAL_STRONTIUM
   is 'Table for requested beta analyses for total Strontium.';

create unique index BETOST_PRIMARY_KEY_INDEX
   on BETA_TOTAL_STRONTIUM (SAMPLE_TRACKING_ID)
   type is sorted
   node size 104
   store in BETOST;

comment on index BETOST_PRIMARY_KEY_INDEX
   is 'Primary key index for table BETA_TOTAL_STRONTIUM.';

create storage map BETOST_map for BETA_TOTAL_STRONTIUM
   store in BETOST
   placement via index BETOST_PRIMARY_KEY_INDEX
```

```
   disable compression;

create table BETA_TRITIUM (
      SAMPLE_TRACKING_ID              IDENTIFICATION_CODES      not null,
      DETECTOR_COUNT_LENGTH            REAL_NUMBERS,
      DETECTOR_COUNT_LENGTH_UNITS      UNITS,
      RESULTS_NEEDED_BY_DATE           INTEGER_DATE_COUNT,
      RESULTS_NEEDED_BY_TIME           CHARACTER_TIME_HH_MM,
      DATE_ALL_WORK_COMPLETED          INTEGER_DATE_COUNT,
      RESULTS_REPORT_CITATION          TWENTY_CHARACTERS,
      ANY_SPECIAL_INSTRUCTIONS         FLAGS,
      USER_CREATING_OR_MODIFYING       USER_NAME,
      DATE_CREATED_OR_MODIFIED         VMS_DATE_TIME,
   primary key (SAMPLE_TRACKING_ID),
   foreign key (SAMPLE_TRACKING_ID)
      references SAMPLE_INFORMATION (SAMPLE_TRACKING_ID) );

comment on table BETA_TRITIUM
   is 'Table for requested beta analyses for Tritium.';

create unique index BETR_PRIMARY_KEY_INDEX
   on BETA_TRITIUM (SAMPLE_TRACKING_ID)
   type is sorted
   node size 104
   store in BETR;

comment on index BETR_PRIMARY_KEY_INDEX
   is 'Primary key index for table BETA_TRITIUM.';

create storage map BETR_map for BETA_TRITIUM
   store in BETR
   placement via index BETR_PRIMARY_KEY_INDEX
   disable compression;

create table BETA_OTHER (
      SAMPLE_TRACKING_ID              IDENTIFICATION_CODES      not null,
      DETECTOR_COUNT_LENGTH            REAL_NUMBERS,
      DETECTOR_COUNT_LENGTH_UNITS      UNITS,
      NEED_BETA_NICKEL_63              FLAGS,
      NEED_BETA_IRON_55                FLAGS,
      NEED_BETA_SULFUR_35              FLAGS,
      NEED_BETA_PLUTONIUM_241          FLAGS,
      RESULTS_NEEDED_BY_DATE           INTEGER_DATE_COUNT,
      RESULTS_NEEDED_BY_TIME           CHARACTER_TIME_HH_MM,
      DATE_ALL_WORK_COMPLETED          INTEGER_DATE_COUNT,
      RESULTS_REPORT_CITATION          TWENTY_CHARACTERS,
      ANY_SPECIAL_INSTRUCTIONS         FLAGS,
      USER_CREATING_OR_MODIFYING       USER_NAME,
      DATE_CREATED_OR_MODIFIED         VMS_DATE_TIME,
   primary key (SAMPLE_TRACKING_ID),
   foreign key (SAMPLE_TRACKING_ID)
      references SAMPLE_INFORMATION (SAMPLE_TRACKING_ID) );

comment on table BETA_OTHER
   is 'Table for requested beta analyses for other, unlisted
radionuclides.';

create unique index BEOT_PRIMARY_KEY_INDEX
   on BETA_OTHER (SAMPLE_TRACKING_ID)
   type is sorted
```

```
   node size 104
   store in BEOT;

comment on index BEOT_PRIMARY_KEY_INDEX
   is 'Primary key index for table BETA_OTHER.';

create storage map BEOT_map for BETA_OTHER
   store in BEOT
   placement via index BEOT_PRIMARY_KEY_INDEX
   disable compression;

create table GAMMA_SCREEN (
      SAMPLE_TRACKING_ID              IDENTIFICATION_CODES      not null,
      DETECTOR_COUNT_LENGTH            REAL_NUMBERS,
      DETECTOR_COUNT_LENGTH_UNITS      UNITS,
      RESULTS_NEEDED_BY_DATE           INTEGER_DATE_COUNT,
      RESULTS_NEEDED_BY_TIME           CHARACTER_TIME_HH_MM,
      DATE_ALL_WORK_COMPLETED          INTEGER_DATE_COUNT,
      DATE_SAMPLE_COUNTED              INTEGER_DATE_COUNT,
      RML_SPECTRAL_ID                  RML_SPECTRAL_ID_CODES,
      IS_THIS_A_SPECTRUM_RECOUNT       FLAGS,
      RESULTS_REPORT_CITATION          TWENTY_CHARACTERS,
      ANY_SPECIAL_INSTRUCTIONS         FLAGS,
      USER_CREATING_OR_MODIFYING       USER_NAME,
      DATE_CREATED_OR_MODIFIED         VMS_DATE_TIME,
   primary key (SAMPLE_TRACKING_ID),
   foreign key (SAMPLE_TRACKING_ID)
      references SAMPLE_INFORMATION (SAMPLE_TRACKING_ID) );

comment on table GAMMA_SCREEN
   is 'Table for requested screening and shipping gamma analyses.';

create unique index GASC_PRIMARY_KEY_INDEX
   on GAMMA_SCREEN (SAMPLE_TRACKING_ID)
   type is sorted
   node size 104
   store in GASC;

comment on index GASC_PRIMARY_KEY_INDEX
   is 'Primary key index for table GAMMA_SCREEN.';

create storage map GASC_map for GAMMA_SCREEN
   store in GASC
   placement via index GASC_PRIMARY_KEY_INDEX
   disable compression;

create table GAMMA_FULL_ISOTOPIC (
      SAMPLE_TRACKING_ID              IDENTIFICATION_CODES      not null,
      DETECTOR_COUNT_LENGTH            REAL_NUMBERS,
      DETECTOR_COUNT_LENGTH_UNITS      UNITS,
      RESULTS_NEEDED_BY_DATE           INTEGER_DATE_COUNT,
      RESULTS_NEEDED_BY_TIME           CHARACTER_TIME_HH_MM,
      DATE_ALL_WORK_COMPLETED          INTEGER_DATE_COUNT,
      DATE_SAMPLE_COUNTED              INTEGER_DATE_COUNT,
      RML_SPECTRAL_ID                  RML_SPECTRAL_ID_CODES,
      IS_THIS_A_SPECTRUM_RECOUNT       FLAGS,
      RESULTS_REPORT_CITATION          TWENTY_CHARACTERS,
      ANY_SPECIAL_INSTRUCTIONS         FLAGS,
      USER_CREATING_OR_MODIFYING       USER_NAME,
      DATE_CREATED_OR_MODIFIED         VMS_DATE_TIME,
```

```
   primary key (SAMPLE_TRACKING_ID),
   foreign key (SAMPLE_TRACKING_ID)
      references SAMPLE_INFORMATION (SAMPLE_TRACKING_ID) );

comment on table GAMMA_FULL_ISOTOPIC
   is 'Table for requested full isotopic gamma analyses.';

create unique index GAFUIS_PRIMARY_KEY_INDEX
   on GAMMA_FULL_ISOTOPIC (SAMPLE_TRACKING_ID)
   type is sorted
   node size 104
   store in GAFUIS;

comment on index GAFUIS_PRIMARY_KEY_INDEX
   is 'Primary key index for table GAMMA_FULL_ISOTOPIC.';

create storage map GAFUIS_map for GAMMA_FULL_ISOTOPIC
   store in GAFUIS
   placement via index GAFUIS_PRIMARY_KEY_INDEX
   disable compression;

create table GAMMA_OTHER (
      SAMPLE_TRACKING_ID              IDENTIFICATION_CODES      not null,
      DETECTOR_COUNT_LENGTH            REAL_NUMBERS,
      DETECTOR_COUNT_LENGTH_UNITS      UNITS,
      RESULTS_NEEDED_BY_DATE           INTEGER_DATE_COUNT,
      RESULTS_NEEDED_BY_TIME           CHARACTER_TIME_HH_MM,
      DATE_ALL_WORK_COMPLETED          INTEGER_DATE_COUNT,
      DATE_SAMPLE_COUNTED              INTEGER_DATE_COUNT,
      RML_SPECTRAL_ID                  RML_SPECTRAL_ID_CODES,
      IS_THIS_A_SPECTRUM_RECOUNT       FLAGS,
      RESULTS_REPORT_CITATION          TWENTY_CHARACTERS,
      ANY_SPECIAL_INSTRUCTIONS         FLAGS,
      USER_CREATING_OR_MODIFYING       USER_NAME,
      DATE_CREATED_OR_MODIFIED         VMS_DATE_TIME,
   primary key (SAMPLE_TRACKING_ID),
   foreign key (SAMPLE_TRACKING_ID)
      references SAMPLE_INFORMATION (SAMPLE_TRACKING_ID) );

comment on table GAMMA_OTHER
   is 'Table for requested gamma analyses for other, unlisted
radionuclides.';

create unique index GAOT_PRIMARY_KEY_INDEX
   on GAMMA_OTHER (SAMPLE_TRACKING_ID)
   type is sorted
   node size 104
   store in GAOT;

comment on index GAOT_PRIMARY_KEY_INDEX
   is 'Primary key index for table GAMMA_OTHER.';

create storage map GAOT_map for GAMMA_OTHER
   store in GAOT
   placement via index GAOT_PRIMARY_KEY_INDEX
   disable compression;

create table GROSS_ALPHA_BETA_AIR_FILTERS (
      SAMPLE_TRACKING_ID              IDENTIFICATION_CODES      not null,
      DETECTOR_COUNT_LENGTH            REAL_NUMBERS,
```

```
      DETECTOR_COUNT_LENGTH_UNITS      UNITS,
      RESULTS_NEEDED_BY_DATE           INTEGER_DATE_COUNT,
      RESULTS_NEEDED_BY_TIME           CHARACTER_TIME_HH_MM,
      DATE_ALL_WORK_COMPLETED          INTEGER_DATE_COUNT,
      RESULTS_REPORT_CITATION          TWENTY_CHARACTERS,
      ANY_SPECIAL_INSTRUCTIONS         FLAGS,
      USER_CREATING_OR_MODIFYING       USER_NAME,
      DATE_CREATED_OR_MODIFIED         VMS_DATE_TIME,
   primary key (SAMPLE_TRACKING_ID),
   foreign key (SAMPLE_TRACKING_ID)
      references SAMPLE_INFORMATION (SAMPLE_TRACKING_ID) );

comment on table GROSS_ALPHA_BETA_AIR_FILTERS
   is 'Table for requested gross alpha-beta analyses of air filters.';

create unique index GRALBEAIFI_PRIMARY_KEY_INDEX
   on GROSS_ALPHA_BETA_AIR_FILTERS (SAMPLE_TRACKING_ID)
   type is sorted
   node size 104
   store in GRALBEAIFI;

comment on index GRALBEAIFI_PRIMARY_KEY_INDEX
   is 'Primary key index for table GROSS_ALPHA_BETA_AIR_FILTERS.';

create storage map GRALBEAIFI_map for GROSS_ALPHA_BETA_AIR_FILTERS
   store in GRALBEAIFI
   placement via index GRALBEAIFI_PRIMARY_KEY_INDEX
   disable compression;

create table GROSS_ALPHA_BETA_OTHER (
      SAMPLE_TRACKING_ID               IDENTIFICATION_CODES      not null,
      DETECTOR_COUNT_LENGTH             REAL_NUMBERS,
      DETECTOR_COUNT_LENGTH_UNITS       UNITS,
      RESULTS_NEEDED_BY_DATE            INTEGER_DATE_COUNT,
      RESULTS_NEEDED_BY_TIME            CHARACTER_TIME_HH_MM,
      DATE_ALL_WORK_COMPLETED           INTEGER_DATE_COUNT,
      RESULTS_REPORT_CITATION           TWENTY_CHARACTERS,
      ANY_SPECIAL_INSTRUCTIONS          FLAGS,
      USER_CREATING_OR_MODIFYING        USER_NAME,
      DATE_CREATED_OR_MODIFIED          VMS_DATE_TIME,
   primary key (SAMPLE_TRACKING_ID),
   foreign key (SAMPLE_TRACKING_ID)
      references SAMPLE_INFORMATION (SAMPLE_TRACKING_ID) );

comment on table GROSS_ALPHA_BETA_OTHER
   is 'Table for requested gross alpha-beta analyses of other, unlisted
samples.';

create unique index GRALBEOT_PRIMARY_KEY_INDEX
   on GROSS_ALPHA_BETA_OTHER (SAMPLE_TRACKING_ID)
   type is sorted
   node size 104
   store in GRALBEOT;

comment on index GRALBEOT_PRIMARY_KEY_INDEX
   is 'Primary key index for table GROSS_ALPHA_BETA_OTHER.';

create storage map GRALBEOT_map for GROSS_ALPHA_BETA_OTHER
   store in GRALBEOT
   placement via index GRALBEOT_PRIMARY_KEY_INDEX
```

```
   disable compression;

create table SPECIAL_INSTRUCTIONS (
      SAMPLE_TRACKING_ID              IDENTIFICATION_CODES      not null,
      ORIGIN_OF_SPECIAL_INSTRUCTION   SIXTEEN_CHARACTERS        not null,
      SPECIAL_INSTRUCTION              FIFTY_CHARACTERS,
      SPECIAL_INSTRUCTION_LINE        INTEGER_NUMBERS           not null,
      USER_CREATING_OR_MODIFYING       USER_NAME,
      DATE_CREATED_OR_MODIFIED         VMS_DATE_TIME,
   primary key (SAMPLE_TRACKING_ID, ORIGIN_OF_SPECIAL_INSTRUCTION,
                SPECIAL_INSTRUCTION_LINE),
   foreign key (SAMPLE_TRACKING_ID)
      references SAMPLE_INFORMATION (SAMPLE_TRACKING_ID) );

comment on table SPECIAL_INSTRUCTIONS
   is 'Table for special instructions.';

create unique index SPIN_PRIMARY_KEY_INDEX
    on    SPECIAL_INSTRUCTIONS    (SAMPLE_TRACKING_ID,
ORIGIN_OF_SPECIAL_INSTRUCTION, SPECIAL_INSTRUCTION_LINE)
   type is sorted
   node size 170
   store in SPIN;

comment on index SPIN_PRIMARY_KEY_INDEX
   is 'Primary key index for table SPECIAL_INSTRUCTIONS.';

create storage map SPIN_map for SPECIAL_INSTRUCTIONS
   store in SPIN
   placement via index SPIN_PRIMARY_KEY_INDEX
   disable compression;

create table TRACKING (
      SAMPLE_TRACKING_ID              IDENTIFICATION_CODES      not null,
      POSSESSOR_SEQUENCE_NUMBER       INTEGER_NUMBERS           not null,
      POSSESSOR_NAME                   NAMES,
      POSSESSOR_PHONE_NUMBER           PHONE_NUMBERS,
      POSSESSOR_ADDRESS                SIXTY_CHARACTERS,
      POSSESSOR_POSSESSION_REASON      FIFTY_CHARACTERS,
      DATE_ACCEPTED_BY_POSSESSOR       INTEGER_DATE_COUNT,
      DATE_RELINQUISHED_BY_POSSESSOR   INTEGER_DATE_COUNT,
      USER_CREATING_OR_MODIFYING       USER_NAME,
      DATE_CREATED_OR_MODIFIED         VMS_DATE_TIME,
   primary key (SAMPLE_TRACKING_ID, POSSESSOR_SEQUENCE_NUMBER),
   foreign key (SAMPLE_TRACKING_ID)
      references SAMPLE_INFORMATION (SAMPLE_TRACKING_ID) );

comment on table TRACKING
   is 'Table for tracking samples.';

create unique index TR_PRIMARY_KEY_INDEX
   on TRACKING (SAMPLE_TRACKING_ID, POSSESSOR_SEQUENCE_NUMBER)
   type is sorted
   node size 119
   store in TR;

comment on index TR_PRIMARY_KEY_INDEX
   is 'Primary key index for table TRACKING.';

create storage map TR_map for TRACKING
```

```
    store in TR
    placement via index TR_PRIMARY_KEY_INDEX
    disable compression;

commit;

exit;
```

E. BUILD_SCREEN - A ROUTINE TO CONSTRUCT FIELD LABELS AND LOCATIONS FOR DATA ENTRY SCREENS

```
$ ! Routine to construct appropriate labels and locations for screen
$ ! management calls.
$ !
$ ! Version 1 completed May 7, 1991 by D. A. Femec.
$ !
$ ! Open requisite files - input from p1.
$ !
$ open filein 'p1'_screen.dat /read
$ open declar fields.dat /write
$ open scrdec scrdec.dat /write
$ open deffld deffld.dat /write
$ open charac charac.dat /write
$ write charac "C"
$ write charac "C       Length-specified character fields for equivalences."
$ write charac "C"
$ write charac "      character "
$ open logica logica.dat /write
$ write logica "C"
$ write logica "C       Nonarrayed null indicators for data transfers."
$ write logica "C"
$ write logica "      integer*2 "
$ open equiva equiva.dat /write
$ write equiva "C"
$ write equiva "C       Equivalence statements for data transfers."
$ write equiva "C"
$ write equiva "      equivalence "
$ linmbr = 0
$ firstc = "abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789:"
$ maxnfc = 63
$ numfld = 0
$ !
$ ! Read and parse each line.
$ !
$ read filein line /end_of_file=exit   !Read column-counter line.
$ next_line:
$    read filein line /end_of_file=exit
$    linlen = f$length(line)
$    linmbr = linmbr + 1
$    if line .eqs. "" then goto next_line
$ !
$ !    Search for the next field.
$ !
$ fdscol = 1
$ search:
$    fdscol = fdscol + 1
$    if fdscol .ge. linlen then goto next_line
$    if f$locate(f$extract(fdscol,1,line),firstc) .eq. maxnfc then goto search
$ !
$ !    Beginning of field descriptor found - now look for start of input field.
$ !
$    fincol = fdscol
$ field_end:
$    fincol = fincol + 1
$    if f$extract(fincol,1,line) .nes. "_" then goto field_end
$ !
$ !    Beginning of input field found - now look for end of input field.
$ !
$    field = f$edit(f$extract(fdscol,(fincol-fdscol),line),"trim")
$    finend = fincol
```

```
$ input_end:
$    finend = finend + 1
$    if finend .ge. linlen then goto write_field
$    if f$extract(finend,1,line) .eqs. "_" then goto input_end
$ !
$ !    End of input field found - write the field to file, after doubling
$ !    apostrophes.
$ !
$ write_field:
$    apoloc = f$locate("'",field)
$    fldlen = f$length(field)
$    if apoloc .ne. fldlen then field = f$extract(0,apoloc,field) + "'" + -
      f$extract(apoloc,(fldlen - apoloc),field)
$    numfld = numfld + 1
$    maxnoc = finend - fincol
$    fdscol = fdscol - 1
$    fincol = fincol - 1
$    write declar "C"
$    write declar "      field''p2'(''numfld')='" + field + "'"   !Field descriptor.
$    write declar "      fdlen''p2'(''numfld')=''f$length(field)'"
$    write declar "      fdrow''p2'(''numfld')=''linmbr'"    !Row for field.
$    write declar "      fdcol''p2'(''numfld')=''fdscol'"    !Starting column for field descriptor.
$    write declar "      ficol''p2'(''numfld')=''fincol'"    !Starting column for input field.
$    write declar "      mxnoc''p2'(''numfld')=''maxnoc'"    !Length of input field.
$    write declar "      fityp''p2'(''numfld')='a'"    !Data type - start with alphanumeric.
$    write equiva "     . (fidat''p2'(''numfld')(1:''maxnoc'),fld''p2'''numfld'),"
$    write equiva "     . (ifnul''p2'(''numfld'),ifn''p2'''numfld'),"
$    write charac "     . fld''p2'''numfld'*''maxnoc',"
$    write logica "     . ifn''p2'''numfld',"
$    fdscol = finend
$    if fdscol .ge. linlen then goto next_line
$    fdscol = fdscol - 1
$    goto search
$ !
$ ! Exit.
$ !
$ exit:
$    write declar "C"
$    write declar "C    Set all input field renditions to the bit-wise .or. of the display"
$    write declar "C     field rendition with finput."
$    write declar "C"
$    write declar "      do i=1,numfd''p2'"
$    write declar "         do j=1,3
$    write declar "            firnd''p2'(j,i)=finput.or.fdrnd''p2'(j,i)"    !Input field rendition.
$    write declar "            enddo"
$    write declar "         enddo"
$    close filein
$    close declar
$    close charac
$    close logica
$    close equiva
$    if p3 .nes. "Y" then goto exit_continued
$    write scrdec "C"
$    write scrdec "C    Field rendition descriptions."
$    write scrdec "C"
$    write scrdec "      integer*4 ferror/smg$m_blink/,   !For errors in field entries."
$    write scrdec "     . ffixed/smg$m_reverse/,   !For fixed fields."
$    write scrdec "     . finput/smg$m_underline/,   !For input fields."
$    write scrdec "     . foptnl/smg$m_normal/,   !For optional fields."
$    write scrdec "     . freqd/smg$m_bold/   !For required fields."
$    write scrdec "      common /fldscr/ ferror,ffixed,finput,foptnl,freqd"
$ exit_continued:
$    write scrdec "C"
$    write scrdec "C    ... screen's field descriptors and field sizes."
$    write scrdec "C"
$    if p3 .eqs. "Y" then write scrdec -
```

```
"      character*80 field''p2''(''numfld'),fidat''p2''(''numfld'),findat,blanks"
$    if p3 .nes. "Y" then write scrdec -
      "      character*80 field''p2''(''numfld'),fidat''p2''(''numfld')"
$    write scrdec "      character*30 atrbt''p2''(2,''numfld')"
$    write scrdec "      character*14 varbl''p2''(''numfld')"
$    if p3 .eqs. "Y" then write scrdec -
      "      character*1 spaces(80)/80*' '/,fityp''p2''(''numfld')"
$    if p3 .nes. "Y" then write scrdec "      character*1 fityp''p2''(''numfld')"
$    write scrdec "      integer*4 fdlen''p2''(''numfld'),fdrow''p2''(''numfld'),"  + -
     "fdcol''p2''(''numfld'),ficol''p2''(''numfld'),mxnoc''p2''(''numfld'),"
$    write scrdec "     . fdrnd''p2'(3,''numfld'),firnd''p2'(3,''numfld'),   !Field renditions (add/update/search)."
$    if p3 .nes. "Y" then write scrdec "     . numfd''p2'/''numfld'/   !The number of fields."
$    if p3 .eqs. "Y" then write scrdec "     . numfd''p2'/''numfld'/,   !The number of fields."
$    if p3 .eqs. "Y" then write scrdec "     . ferror,ffixed,finput,foptnl,freqd   !Field renditions."
$    write scrdec "      integer*2 ifnul''p2''(''numfld')   !Null indicator (-1 if null, 0 otherwise)."
$    write scrdec "     . ifrqd''p2''(''numfld')/''numfld'*0/,   !Points to if-required dependent field."
$    write scrdec "     . idito''p2''(''numfld')/''numfld'*0/   !Points to allowed field for dittoing."
$    write scrdec "      logical ifmod''p2''(''numfld')   !Has a field been changed in an Update?"
$    write scrdec "      common /dbdat''p2'/ field''p2',fidat''p2',fdlen''p2',fdrow''p2',fdcol''p2',ficol''p2',mxnoc''p2',"
$    write scrdec "     . fdrnd''p2',firnd''p2',numfd''p2',fityp''p2',ifnul''p2',atrbt''p2',varbl''p2',ifrqd''p2',ifmod''p2',"
$    write scrdec "     . idito''p2'"
$    if p3 .eqs. "Y" then write scrdec "      common /fldscr/ ferror,ffixed,finput,foptnl,freqd"
$    if p3 .eqs. "Y" then write scrdec "      equivalence (blanks(1:1),spaces(1))"
$    write deffld "C"
$    write deffld "C    Set defaults for display field renditions now so they can be changed"
$    write deffld "C     with each field as necessary."
$    write deffld "C"
$    write deffld "      do i=1,numfd''p2'"
$    write deffld "         do j=1,3   !1 for adding, 2 for updating, and 3 for searching."
$    write deffld "            fdrnd''p2'(j,i)=foptnl"   !Display field rendition.
$    write deffld "            enddo"
$    write deffld "         enddo"
$    close scrdec
$    close deffld
$    copy/concatenate scrdec.dat + charac.dat + logica.dat + equiva.dat -
      + deffld.dat + fields.dat 'p1'_screen.for
$    delete scrdec.dat;,fields.dat;,charac.dat;,logica.dat;,equiva.dat;, -
      deffld.dat;
$    exit
```

F. SCREEN TEMPLATE FOR THE GROSS ALPHA-BETA ANALYSES SCREEN SUBMITTED TO BUILD_SCREEN

```
 12345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678
 1
 2
 3
 4
 5Tracking ID:  ____________
 6   Sample Name:  ____________________________________________________________
 7
 8Gross Alpha-Beta for Air Filters?  _
 9    Length of count:  _____    min or hr?  _
10    Results needed by:  date:  ______    time:  _____    Completed:  ______
11    Results report citation:  ____________________    Special Instructions?  _
12
13Gross Alpha-Beta for Other Samples?  _
14    Length of count:  _____    min or hr?  _
15    Results needed by:  date:  ______    time:  _____    Completed:  ______
16    Results report citation:  ____________________    Special Instructions?  _
17
18
19
20
```

G. BUILD_SCREEN OUTPUT FOR THE GROSS ALPHA-BETA ANALYSES SCREEN

```
C
C     ... screen's field descriptors and field sizes.
C
      character*80 fieldC(18),fidatC(18)
      character*30 atrbtC(2,18)
      character*14 varblC(18)
      character*1 fitypC(18)
      integer*4 fdlenC(18),fdrowC(18),fdcolC(18),ficolC(18),mxnocC(18),
     . fdrndC(3,18),firndC(3,18), !Field renditions (add/update/search).
     . numfdC/18/   !The number of fields.
      integer*2 ifnulC(18)   !Null indicator (-1 if null, 0 otherwise).
     . ifrqdC(18)/18*0/,   !Points to if-required dependent field.
     . iditoC(18)/18*0/   !Points to allowed field for dittoing.
      logical ifmodC(18)   !Has a field been changed in an Update?
      common /dbdatC/ fieldC,fidatC,fdlenC,fdrowC,fdcolC,ficolC,mxnocC,
     . fdrndC,firndC,numfdC,fitypC,ifnulC,atrbtC,varblC,ifrqdC,ifmodC,
     . iditoC
C
C       Length-specified character fields for equivalences.
C
      character
     . fldC1*12,
     . fldC2*60,
     . fldC3*1,
     . fldC4*4,
     . fldC5*1,
     . fldC6*6,
     . fldC7*5,
     . fldC8*6,
     . fldC9*20,
     . fldC10*1,
     . fldC11*1,
     . fldC12*4,
     . fldC13*1,
     . fldC14*6,
     . fldC15*5,
     . fldC16*6,
     . fldC17*20,
     . fldC18*1,
C
C       Nonarrayed null indicators for data transfers.
C
      integer*2
     . ifnC1,
     . ifnC2,
     . ifnC3,
     . ifnC4,
     . ifnC5,
     . ifnC6,
     . ifnC7,
     . ifnC8,
     . ifnC9,
     . ifnC10,
     . ifnC11,
     . ifnC12,
```

```
     . ifnC13,
     . ifnC14,
     . ifnC15,
     . ifnC16,
     . ifnC17,
     . ifnC18,
C
C       Equivalence statements for data transfers.
C
      equivalence
     . (fidatC(1)(1:12),fldC1),
     . (ifnulC(1),ifnC1),
     . (fidatC(2)(1:60),fldC2),
     . (ifnulC(2),ifnC2),
     . (fidatC(3)(1:1),fldC3),
     . (ifnulC(3),ifnC3),
     . (fidatC(4)(1:4),fldC4),
     . (ifnulC(4),ifnC4),
     . (fidatC(5)(1:1),fldC5),
     . (ifnulC(5),ifnC5),
     . (fidatC(6)(1:6),fldC6),
     . (ifnulC(6),ifnC6),
     . (fidatC(7)(1:5),fldC7),
     . (ifnulC(7),ifnC7),
     . (fidatC(8)(1:6),fldC8),
     . (ifnulC(8),ifnC8),
     . (fidatC(9)(1:20),fldC9),
     . (ifnulC(9),ifnC9),
     . (fidatC(10)(1:1),fldC10),
     . (ifnulC(10),ifnC10),
     . (fidatC(11)(1:1),fldC11),
     . (ifnulC(11),ifnC11),
     . (fidatC(12)(1:4),fldC12),
     . (ifnulC(12),ifnC12),
     . (fidatC(13)(1:1),fldC13),
     . (ifnulC(13),ifnC13),
     . (fidatC(14)(1:6),fldC14),
     . (ifnulC(14),ifnC14),
     . (fidatC(15)(1:5),fldC15),
     . (ifnulC(15),ifnC15),
     . (fidatC(16)(1:6),fldC16),
     . (ifnulC(16),ifnC16),
     . (fidatC(17)(1:20),fldC17),
     . (ifnulC(17),ifnC17),
     . (fidatC(18)(1:1),fldC18),
     . (ifnulC(18),ifnC18),
C
C    Set defaults for display field renditions now so they can be changed
C     with each field as necessary.
C
      do i=1,numfdC
         do j=1,3   !1 for adding, 2 for updating, and 3 for searching.
            fdrndC(i,j)=foptn1
            enddo
         enddo
C
      fieldC(1)='Tracking ID:'
      fdlenC(1)=12
      fdrowC(1)=5
      fdcolC(1)=1
```

```
      ficolC(1)=15
      mxnocC(1)=12
      fitypC(1)='a'
C
      fieldC(2)='Sample Name:'
      fdlenC(2)=12
      fdrowC(2)=6
      fdcolC(2)=4
      ficolC(2)=18
      mxnocC(2)=60
      fitypC(2)='a'
C
      fieldC(3)='Gross Alpha-Beta for Air Filters?'
      fdlenC(3)=33
      fdrowC(3)=8
      fdcolC(3)=1
      ficolC(3)=36
      mxnocC(3)=1
      fitypC(3)='a'
C
      fieldC(4)='Length of count:'
      fdlenC(4)=16
      fdrowC(4)=9
      fdcolC(4)=4
      ficolC(4)=22
      mxnocC(4)=4
      fitypC(4)='a'
C
      fieldC(5)='min or hr?'
      fdlenC(5)=10
      fdrowC(5)=9
      fdcolC(5)=30
      ficolC(5)=42
      mxnocC(5)=1
      fitypC(5)='a'
C
      fieldC(6)='Results needed by:  date:'
      fdlenC(6)=25
      fdrowC(6)=10
      fdcolC(6)=4
      ficolC(6)=31
      mxnocC(6)=6
      fitypC(6)='a'
C
      fieldC(7)='time:'
      fdlenC(7)=5
      fdrowC(7)=10
      fdcolC(7)=41
      ficolC(7)=48
      mxnocC(7)=5
      fitypC(7)='a'
C
      fieldC(8)='Completed:'
      fdlenC(8)=10
      fdrowC(8)=10
      fdcolC(8)=57
      ficolC(8)=69
      mxnocC(8)=6
      fitypC(8)='a'
C
```

```
      fieldC(9)='Results report citation:'
      fdlenC(9)=24
      fdrowC(9)=11
      fdcolC(9)=4
      ficolC(9)=30
      mxnocC(9)=20
      fitypC(9)='a'
C
      fieldC(10)='Special Instructions?'
      fdlenC(10)=21
      fdrowC(10)=11
      fdcolC(10)=54
      ficolC(10)=77
      mxnocC(10)=1
      fitypC(10)='a'
C
      fieldC(11)='Gross Alpha-Beta for Other Samples?'
      fdlenC(11)=35
      fdrowC(11)=13
      fdcolC(11)=1
      ficolC(11)=38
      mxnocC(11)=1
      fitypC(11)='a'
C
      fieldC(12)='Length of count:'
      fdlenC(12)=16
      fdrowC(12)=14
      fdcolC(12)=4
      ficolC(12)=22
      mxnocC(12)=4
      fitypC(12)='a'
C
      fieldC(13)='min or hr?'
      fdlenC(13)=10
      fdrowC(13)=14
      fdcolC(13)=30
      ficolC(13)=42
      mxnocC(13)=1
      fitypC(13)='a'
C
      fieldC(14)='Results needed by:  date:'
      fdlenC(14)=25
      fdrowC(14)=15
      fdcolC(14)=4
      ficolC(14)=31
      mxnocC(14)=6
      fitypC(14)='a'
C
      fieldC(15)='time:'
      fdlenC(15)=5
      fdrowC(15)=15
      fdcolC(15)=41
      ficolC(15)=48
      mxnocC(15)=5
      fitypC(15)='a'
C
      fieldC(16)='Completed:'
      fdlenC(16)=10
      fdrowC(16)=15
      fdcolC(16)=57
```

```
      ficolC(16)=69
      mxnocC(16)=6
      fitypC(16)='a'
C
      fieldC(17)='Results report citation:'
      fdlenC(17)=24
      fdrowC(17)=16
      fdcolC(17)=4
      ficolC(17)=30
      mxnocC(17)=20
      fitypC(17)='a'
C
      fieldC(18)='Special Instructions?'
      fdlenC(18)=21
      fdrowC(18)=16
      fdcolC(18)=54
      ficolC(18)=77
      mxnocC(18)=1
      fitypC(18)='a'
C
C     Set all input field renditions to the bit-wise .or. of the display
C      field rendition with finput.
C
      do i=1,numfdC
         do j=1,3
            firndC(j,i)=finput.or.fdrndC(j,i)
            enddo
         enddo
```

H. ARRAY AND SCALAR DECLARATIONS RELATED TO THE SCREEN MANAGEMENT ROUTINES

```
C
C Declaration statements.
C
C    Component of SAMPLE_TRACKING Version 1 completed June 6, 1991 by
C    D. A. Femec.
C
C    Note that variable names are limited to six-characters in length;
C    this is a limitation imposed by SQL for variables used in SQL calls.
C
C    Field rendition descriptions.
C
      integer*4 ferror/smg$m_blink/,   !For errors in field entries.
     . ffixed/smg$m_reverse/,   !For fixed fields.
     . finput/smg$m_underline/,   !For input fields.
     . foptnl/smg$m_normal/,   !For optional fields.
     . freqd/smg$m_bold/   !For required fields.
      common /fldscr/ ferror,ffixed,finput,foptnl,freqd
C
C    Main screen's field descriptors and field sizes.
C
      character*80 fieldM(36),fidatM(36),findat,blanks
      character*30 atrbtM(2,38)
      character*14 varblM(38)
      character*1 spaces(80)/80*' '/,fitypM(36)
      integer*4 fdlenM(36),fdrowM(36),fdcolM(36),ficolM(36),mxnocM(36),
     . fdrndM(3,36),firndM(3,36),  !Field renditions (add/update/search).
     . numfdM/36/   !The number of fields.
      integer*2 ifnulM(36),   !Null indicator (-1 if null, 0 otherwise).
     . ifrqdM(36)/36*0/,   !Points to if-required dependent field.
     . iditoM(36)/36*0/   !Points to allowed field for dittoing.
      logical ifmodM(36)   !Has a field been modified in an Update?
      common /dbdatM/ fieldM,fidatM,fdlenM,fdrowM,fdcolM,ficolM,mxnocM,
     . fdrndM,firndM,numfdM,fitypM,ifnulM,atrbtM,varblM,ifrqdM,ifmodM,
     . iditoM
      equivalence (blanks(1:1),spaces(1))
C
C      Length-specified character fields for equivalences (f for field).
C
      character fMSTI*12,fMDSE*6,fMDAWC*6,fMCSI*12,fMRNBD*6,fMCPN*40,
     . fMWCN*9,fMCSN*60,fMST*10,fMSS*5,fMSSU*5,fMSCD*6,fMSCT*5,fMASI*1,
     . fMSHS*1,fMSHSA*6,fMSSN*25,fMSSPN*12,fMSSA*60,fMTCN*25,fMTCPN*12,
     . fMTCA*60,fMSRTN*25,fMSRTP*12,fMSRTA*60,fMSPBN*25,fMSPBP*12,
     . fMSPBA*60,fMPN*25,fMPPN*12,fMPA*60,fMCOCN*10,fMNGAB*1,fMNA*1,
     . fMNB*1,fMNG*1
C
C       Real and integer field storage (r for real, i for integer).
C
      real*4 rMSS,rMSHSA
      integer*4 iMDSE,iMDAWC,iMRNBD,iMSCD
C     . ,nxinst,nxcust   !Declared in the main routine.
C
C       Nonarrayed field lengths (l for lengths).
C
      integer*4 lMSTI,lMDSE,lMDAWC,lMCSI,lMRNBD,lMCPN,lMWCN,lMCSN,lMST,
     . lMSS,lMSSU,lMSCD,lMSCT,lMASI,lMSHS,lMSHSA,lMSSN,lMSSPN,lMSSA,
```

```
     . lMTCN,lMTCPN,lMTCA,lMSRTN,lMSRTP,lMSRTA,lMSPBN,lMSPBP,lMSPBA,
     . lMPN,lMPPN,lMPA,lMCOCN,lMNGAB,lMNA,lMNB,lMNG
C
C       Nonarrayed null indicators for data transfers (n for null).
C
      integer*2 nMSTI,nMDSE,nMDAWC,nMCSI,nMRNBD,nMCPN,nMWCN,nMCSN,nMST,
     . nMSS,nMSSU,nMSCD,nMSCT,nMASI,nMSHS,nMSHSA,nMSSN,nMSSPN,nMSSA,
     . nMTCN,nMTCPN,nMTCA,nMSRTN,nMSRTP,nMSRTA,nMSPBN,nMSPBP,nMSPBA,
     . nMPN,nMPPN,nMPA,nMCOCN,nMNGAB,nMNA,nMNB,nMNG
C
C       Nonarrayed if-modified indicators (m for modified).
C
      integer*2 mMSTI,mMDSE,mMDAWC,mMCSI,mMRNBD,mMCPN,mMWCN,mMCSN,mMST,
     . mMSS,mMSSU,mMSCD,mMSCT,mMASI,mMSHS,mMSHSA,mMSSN,mMSSPN,mMSSA,
     . mMTCN,mMTCPN,mMTCA,mMSRTN,mMSRTP,mMSRTA,mMSPBN,mMSPBP,mMSPBA,
     . mMPN,mMPPN,mMPA,mMCOCN,mMNGAB,mMNA,mMNB,mMNG
C
C       Equivalence statements for data transfers.
C
      equivalence (fidatM(1)(1:12),fMSTI),(ifnulM(1),nMSTI),
     . (fidatM(2)(1:6),fMDSE),(ifnulM(2),nMDSE),
     . (fidatM(3)(1:6),fMDAWC),(ifnulM(3),nMDAWC),
     . (fidatM(4)(1:12),fMCSI),(ifnulM(4),nMCSI),
     . (fidatM(5)(1:6),fMRNBD),(ifnulM(5),nMRNBD),
     . (fidatM(6)(1:40),fMCPN),(ifnulM(6),nMCPN),
     . (fidatM(7)(1:9),fMWCN),(ifnulM(7),nMWCN),
     . (fidatM(8)(1:60),fMCSN),(ifnulM(8),nMCSN),
     . (fidatM(9)(1:10),fMST),(ifnulM(9),nMST),
     . (fidatM(10)(1:5),fMSS),(ifnulM(10),nMSS),
     . (fidatM(11)(1:5),fMSSU),(ifnulM(11),nMSSU),
     . (fidatM(12)(1:6),fMSCD),(ifnulM(12),nMSCD),
     . (fidatM(13)(1:5),fMSCT),(ifnulM(13),nMSCT),
     . (fidatM(14)(1:1),fMASI),(ifnulM(14),nMASI),
     . (fidatM(15)(1:1),fMSHS),(ifnulM(15),nMSHS),
     . (fidatM(16)(1:6),fMSHSA),(ifnulM(16),nMSHSA),
     . (fidatM(17)(1:25),fMSSN),(ifnulM(17),nMSSN),
     . (fidatM(18)(1:12),fMSSPN),(ifnulM(18),nMSSPN),
     . (fidatM(19)(1:60),fMSSA),(ifnulM(19),nMSSA),
     . (fidatM(20)(1:25),fMTCN),(ifnulM(20),nMTCN),
     . (fidatM(21)(1:12),fMTCPN),(ifnulM(21),nMTCPN),
     . (fidatM(22)(1:60),fMTCA),(ifnulM(22),nMTCA),
     . (fidatM(23)(1:25),fMSRTN),(ifnulM(23),nMSRTN),
     . (fidatM(24)(1:12),fMSRTP),(ifnulM(24),nMSRTP),
     . (fidatM(25)(1:60),fMSRTA),(ifnulM(25),nMSRTA),
     . (fidatM(26)(1:25),fMSPBN),(ifnulM(26),nMSPBN),
     . (fidatM(27)(1:12),fMSPBP),(ifnulM(27),nMSPBP),
     . (fidatM(28)(1:60),fMSPBA),(ifnulM(28),nMSPBA),
     . (fidatM(29)(1:25),fMPN),(ifnulM(29),nMPN),
     . (fidatM(30)(1:12),fMPPN),(ifnulM(30),nMPPN),
     . (fidatM(31)(1:60),fMPA),(ifnulM(31),nMPA),
     . (fidatM(32)(1:10),fMCOCN),(ifnulM(32),nMCOCN),
     . (fidatM(33)(1:1),fMNGAB),(ifnulM(33),nMNGAB),
     . (fidatM(34)(1:1),fMNA),(ifnulM(34),nMNA),
     . (fidatM(35)(1:1),fMNB),(ifnulM(35),nMNB),
     . (fidatM(36)(1:1),fMNG),(ifnulM(36),nMNG),
     . (mxnocM(1),lMSTI),(ifmodM(1),mMSTI),
     . (mxnocM(2),lMDSE),(ifmodM(2),mMDSE),
     . (mxnocM(3),lMDAWC),(ifmodM(3),mMDAWC),
     . (mxnocM(4),lMCSI),(ifmodM(4),mMCSI),
     . (mxnocM(5),lMRNBD),(ifmodM(5),mMRNBD),
```

```
     . (mxnocM(6),lMCPN),(ifmodM(6),mMCPN),
     . (mxnocM(7),lMWCN),(ifmodM(7),mMWCN),
     . (mxnocM(8),lMCSN),(ifmodM(8),mMCSN),
     . (mxnocM(9),lMST),(ifmodM(9),mMST),
     . (mxnocM(10),lMSS),(ifmodM(10),mMSS),
     . (mxnocM(11),lMSSU),(ifmodM(11),mMSSU),
     . (mxnocM(12),lMSCD),(ifmodM(12),mMSCD),
     . (mxnocM(13),lMSCT),(ifmodM(13),mMSCT),
     . (mxnocM(14),lMASI),(ifmodM(14),mMASI),
     . (mxnocM(15),lMSHS),(ifmodM(15),mMSHS),
     . (mxnocM(16),lMSHSA),(ifmodM(16),mMSHSA),
     . (mxnocM(17),lMSSN),(ifmodM(17),mMSSN),
     . (mxnocM(18),lMSSPN),(ifmodM(18),mMSSPN),
     . (mxnocM(19),lMSSA),(ifmodM(19),mMSSA),
     . (mxnocM(20),lMTCN),(ifmodM(20),mMTCN),
     . (mxnocM(21),lMTCPN),(ifmodM(21),mMTCPN),
     . (mxnocM(22),lMTCA),(ifmodM(22),mMTCA),
     . (mxnocM(23),lMSRTN),(ifmodM(23),mMSRTN),
     . (mxnocM(24),lMSRTP),(ifmodM(24),mMSRTP),
     . (mxnocM(25),lMSRTA),(ifmodM(25),mMSRTA),
     . (mxnocM(26),lMSPBN),(ifmodM(26),mMSPBN),
     . (mxnocM(27),lMSPBP),(ifmodM(27),mMSPBP),
     . (mxnocM(28),lMSPBA),(ifmodM(28),mMSPBA),
     . (mxnocM(29),lMPN),(ifmodM(29),mMPN),
     . (mxnocM(30),lMPPN),(ifmodM(30),mMPPN),
     . (mxnocM(31),lMPA),(ifmodM(31),mMPA),
     . (mxnocM(32),lMCOCN),(ifmodM(32),mMCOCN),
     . (mxnocM(33),lMNGAB),(ifmodM(33),mMNGAB),
     . (mxnocM(34),lMNA),(ifmodM(34),mMNA),
     . (mxnocM(35),lMNB),(ifmodM(35),mMNB),
     . (mxnocM(36),lMNG),(ifmodM(36),mMNG)
C
C    Database table and attribute names for screen fields.
C
      data atrbtM/
     . 'SAMPLE_INFORMATION','SAMPLE_TRACKING_ID',
     . 'SAMPLE_INFORMATION','DATE_SAMPLE_ENTERED',
     . 'SAMPLE_INFORMATION','DATE_ALL_WORK_COMPLETED',
     . 'SAMPLE_INFORMATION','CUSTOMER_SAMPLE_ID',
     . 'SAMPLE_INFORMATION','RESULTS_NEEDED_BY_DATE',
     . 'SAMPLE_INFORMATION','CUSTOMER_PROJECT_NAME',
     . 'SAMPLE_INFORMATION','WORK_CHARGE_NUMBER',
     . 'SAMPLE_INFORMATION','CUSTOMER_SAMPLE_NAME',
     . 'SAMPLE_INFORMATION','SAMPLE_TYPE',
     . 'SAMPLE_INFORMATION','SAMPLE_SIZE',
     . 'SAMPLE_INFORMATION','SAMPLE_SIZE_UNITS',
     . 'SAMPLE_INFORMATION','SAMPLE_COLLECTION_DATE',
     . 'SAMPLE_INFORMATION','SAMPLE_COLLECTION_TIME',
     . 'SAMPLE_INFORMATION','ANY_SPECIAL_INSTRUCTIONS',
     . 'SAMPLE_INFORMATION','SAMPLE_HP_SURVEYED',
     . 'SAMPLE_INFORMATION','SAMPLE_HP_SURVEY_ACTIVITY',
     . 'SAMPLE_INFORMATION','SAMPLE_SUBMITTER_NAME',
     . 'SAMPLE_INFORMATION','SAMPLE_SUBMITTER_PHONE_NUMBER',
     . 'SAMPLE_INFORMATION','SAMPLE_SUBMITTER_ADDRESS',
     . 'SAMPLE_INFORMATION','TECHNICAL_CONTACT_NAME',
     . 'SAMPLE_INFORMATION','TECHNICAL_CONTACT_PHONE_NUMBER',
     . 'SAMPLE_INFORMATION','TECHNICAL_CONTACT_ADDRESS',
     . 'SAMPLE_INFORMATION','SEND_RESULTS_TO_NAME',
     . 'SAMPLE_INFORMATION','SEND_RESULTS_TO_PHONE_NUMBER',
     . 'SAMPLE_INFORMATION','SEND_RESULTS_TO_ADDRESS',
```

```
     . 'SAMPLE_INFORMATION','SAMPLE_PICKUP_BY_NAME',
     . 'SAMPLE_INFORMATION','SAMPLE_PICKUP_BY_PHONE_NUMBER',
     . 'SAMPLE_INFORMATION','SAMPLE_PICKUP_BY_ADDRESS',
     . 'TRACKING','POSSESSOR_NAME',
     . 'TRACKING','POSSESSOR_PHONE_NUMBER',
     . 'TRACKING','POSSESSOR_ADDRESS',
     . 'SAMPLE_INFORMATION','CHAIN_OF_CUSTODY_NUMBER',
     . 'SAMPLE_INFORMATION','NEED_GROSS_ALPHA_BETA',
     . 'SAMPLE_INFORMATION','NEED_ALPHA',
     . 'SAMPLE_INFORMATION','NEED_BETA',
     . 'SAMPLE_INFORMATION','NEED_GAMMA',
     . 'SAMPLE_INFORMATION','NEXT_SPECIAL_INSTRUCTION_LINE',
     . 'SAMPLE_INFORMATION','NEXT_POSSESSOR_SEQUENCE_NUMBER'/
C
C    Program variable names (value and null indicator) for screen fields.
C
      data varblM/':fMSTI:nMSTI',':iMDSE:nMDSE',':iMDAWC:nMDAWC',
     . ':fMCSI:nMCSI',':iMRNBD:nMRNBD',':fMCPN:nMCPN',':fMWCN:nMWCN',
     . ':fMCSN:nMCSN',':fMST:nMST',':rMSS:nMSS',':fMSSU:nMSSU',
     . ':iMSCD:nMSCD',':fMSCT:nMSCT',':fMASI:nMASI',':fMSHS:nMSHS',
     . ':rMSHSA:nMSHSA',':fMSSN:nMSSN',':fMSSPN:nMSSPN',':fMSSA:nMSSA',
     . ':fMTCN:nMTCN',':fMTCPN:nMTCPN',':fMTCA:nMTCA',':fMSRTN:nMSRTN',
     . ':fMSRTP:nMSRTP',':fMSRTA:nMSRTA',':fMSPBN:nMSPBN',
     . ':fMSPBP:nMSPBP',':fMSPBA:nMSPBA',':fMPN:nMPN',
     . ':fMPPN:nMPPN',':fMPA:nMPA',':fMCOCN:nMCOCN',':fMNGAB:nMNGAB',
     . ':fMNA:nMNA',':fMNB:nMNB',':fMNG:nMNG',':nxinst:ifnnxi',
     . ':nxcust:ifnnxc'/
C
C    Alpha screen's field descriptors and field sizes.
C
      character*80 fieldA(37),fidatA(37)
      character*30 atrbtA(2,37)
      character*14 varblA(37)
      character*1 fitypA(37)
      integer*4 fdlenA(37),fdrowA(37),fdcolA(37),ficolA(37),mxnocA(37),
     . fdrndA(3,37),firndA(3,37), !Field renditions (add/update/search).
     . numfdA/37/   !The number of fields.
      integer*2 ifnulA(37),   !Null indicator (-1 if null, 0 otherwise).
     . ifrqdA(37)/37*0/,   !Points to if-required dependent field.
     . iditoA(37)/37*0/   !Points to allowed field for dittoing.
      logical ifmodA(37)   !Has a field been modified in an Update?
      common /dbdatA/ fieldA,fidatA,fdlenA,fdrowA,fdcolA,ficolA,mxnocA,
     . fdrndA,firndA,numfdA,fitypA,ifnulA,atrbtA,varblA,ifrqdA,ifmodA,
     . iditoA
C
C      Length-specified character fields for equivalences (f for field).
C
      character fASTI*12,fACSN*60,fANAU*1,fARNB1*6,fADAW1*6,fARRC1*20,
     . fAASI1*1,fANATh*1,fARNB2*6,fADAW2*6,fARRC2*20,fAASI2*1,fANAPu*1,
     . fARNB3*6,fADAW3*6,fARRC3*20,fAASI3*1,fANASP*1,fARNB4*6,fADAW4*6,
     . fARRC4*20,fAASI4*1,fANAWP*1,fARNB5*6,fADAW5*6,fARRC5*20,fAASI5*1,
     . fANATS*1,fARNB6*6,fADAW6*6,fARRC6*20,fAASI6*1,fANAO*1,fARNB7*6,
     . fADAW7*6,fARRC7*20,fAASI7*1
C
C       Integer field storage (i for integer).
C
      integer*4 iARNB1,iADAW1,iARNB2,iADAW2,iARNB3,iADAW3,iARNB4,iADAW4,
     . iARNB5,iADAW5,iARNB6,iADAW6,iARNB7,iADAW7
C
C       Nonarrayed maximum field lengths (l for length).
```

```
C
      integer*4 lASTI,lACSN,lANAU,lARNB1,lADAW1,lARRC1,lAASI1,lANATh,
     . lARNB2,lADAW2,lARRC2,lAASI2,lANAPu,lARNB3,lADAW3,lARRC3,lAASI3,
     . lANASP,lARNB4,lADAW4,lARRC4,lAASI4,lANAWP,lARNB5,lADAW5,lARRC5,
     . lAASI5,lANATS,lARNB6,lADAW6,lARRC6,lAASI6,lANAO,lARNB7,lADAW7,
     . lARRC7,lAASI7
C
C       Nonarrayed null indicators for data transfers (n for null).
C
      integer*2 nASTI,nACSN,nANAU,nARNB1,nADAW1,nARRC1,nAASI1,nANATh,
     . nARNB2,nADAW2,nARRC2,nAASI2,nANAPu,nARNB3,nADAW3,nARRC3,nAASI3,
     . nANASP,nARNB4,nADAW4,nARRC4,nAASI4,nANAWP,nARNB5,nADAW5,nARRC5,
     . nAASI5,nANATS,nARNB6,nADAW6,nARRC6,nAASI6,nANAO,nARNB7,nADAW7,
     . nARRC7,nAASI7
C
C       Nonarrayed if-modified indicators (m for modified).
C
      integer*2 mASTI,mACSN,mANAU,mARNB1,mADAW1,mARRC1,mAASI1,mANATh,
     . mARNB2,mADAW2,mARRC2,mAASI2,mANAPu,mARNB3,mADAW3,mARRC3,mAASI3,
     . mANASP,mARNB4,mADAW4,mARRC4,mAASI4,mANAWP,mARNB5,mADAW5,mARRC5,
     . mAASI5,mANATS,mARNB6,mADAW6,mARRC6,mAASI6,mANAO,mARNB7,mADAW7,
     . mARRC7,mAASI7
C
C       Equivalence statements for data transfers.
C
      equivalence (fidatA(1)(1:12),fASTI),(ifnulA(1),nASTI),
     . (fidatA(2)(1:60),fACSN),(ifnulA(2),nACSN),
     . (fidatA(3)(1:1),fANAU),(ifnulA(3),nANAU),
     . (fidatA(4)(1:6),fARNB1),(ifnulA(4),nARNB1),
     . (fidatA(5)(1:6),fADAW1),(ifnulA(5),nADAW1),
     . (fidatA(6)(1:20),fARRC1),(ifnulA(6),nARRC1),
     . (fidatA(7)(1:1),fAASI1),(ifnulA(7),nAASI1),
     . (fidatA(8)(1:1),fANATh),(ifnulA(8),nANATh),
     . (fidatA(9)(1:6),fARNB2),(ifnulA(9),nARNB2),
     . (fidatA(10)(1:6),fADAW2),(ifnulA(10),nADAW2),
     . (fidatA(11)(1:20),fARRC2),(ifnulA(11),nARRC2),
     . (fidatA(12)(1:1),fAASI2),(ifnulA(12),nAASI2),
     . (fidatA(13)(1:1),fANAPu),(ifnulA(13),nANAPu),
     . (fidatA(14)(1:6),fARNB3),(ifnulA(14),nARNB3),
     . (fidatA(15)(1:6),fADAW3),(ifnulA(15),nADAW3),
     . (fidatA(16)(1:20),fARRC3),(ifnulA(16),nARRC3),
     . (fidatA(17)(1:1),fAASI3),(ifnulA(17),nAASI3),
     . (fidatA(18)(1:1),fANASP),(ifnulA(18),nANASP),
     . (fidatA(19)(1:6),fARNB4),(ifnulA(19),nARNB4),
     . (fidatA(20)(1:6),fADAW4),(ifnulA(20),nADAW4),
     . (fidatA(21)(1:20),fARRC4),(ifnulA(21),nARRC4),
     . (fidatA(22)(1:1),fAASI4),(ifnulA(22),nAASI4),
     . (fidatA(23)(1:1),fANAWP),(ifnulA(23),nANAWP),
     . (fidatA(24)(1:6),fARNB5),(ifnulA(24),nARNB5),
     . (fidatA(25)(1:6),fADAW5),(ifnulA(25),nADAW5),
     . (fidatA(26)(1:20),fARRC5),(ifnulA(26),nARRC5),
     . (fidatA(27)(1:1),fAASI5),(ifnulA(27),nAASI5),
     . (fidatA(28)(1:1),fANATS),(ifnulA(28),nANATS),
     . (fidatA(29)(1:6),fARNB6),(ifnulA(29),nARNB6),
     . (fidatA(30)(1:6),fADAW6),(ifnulA(30),nADAW6),
     . (fidatA(31)(1:20),fARRC6),(ifnulA(31),nARRC6),
     . (fidatA(32)(1:1),fAASI6),(ifnulA(32),nAASI6),
     . (fidatA(33)(1:1),fANAO),(ifnulA(33),nANAO),
     . (fidatA(34)(1:6),fARNB7),(ifnulA(34),nARNB7),
     . (fidatA(35)(1:6),fADAW7),(ifnulA(35),nADAW7),
```

```
     . (fidatA(36)(1:20),fARRC7),(ifnulA(36),nARRC7),
     . (fidatA(37)(1:1),fAASI7),(ifnulA(37),nAASI7),
     . (mxnocA(1),lASTI),(ifmodA(1),mASTI),
     . (mxnocA(2),lACSN),(ifmodA(2),mACSN),
     . (mxnocA(3),lANAU),(ifmodA(3),mANAU),
     . (mxnocA(4),lARNB1),(ifmodA(4),mARNB1),
     . (mxnocA(5),lADAW1),(ifmodA(5),mADAW1),
     . (mxnocA(6),lARRC1),(ifmodA(6),mARRC1),
     . (mxnocA(7),lAASI1),(ifmodA(7),mAASI1),
     . (mxnocA(8),lANATh),(ifmodA(8),mANATh),
     . (mxnocA(9),lARNB2),(ifmodA(9),mARNB2),
     . (mxnocA(10),lADAW2),(ifmodA(10),mADAW2),
     . (mxnocA(11),lARRC2),(ifmodA(11),mARRC2),
     . (mxnocA(12),lAASI2),(ifmodA(12),mAASI2),
     . (mxnocA(13),lANAPu),(ifmodA(13),mANAPu),
     . (mxnocA(14),lARNB3),(ifmodA(14),mARNB3),
     . (mxnocA(15),lADAW3),(ifmodA(15),mADAW3),
     . (mxnocA(16),lARRC3),(ifmodA(16),mARRC3),
     . (mxnocA(17),lAASI3),(ifmodA(17),mAASI3),
     . (mxnocA(18),lANASP),(ifmodA(18),mANASP),
     . (mxnocA(19),lARNB4),(ifmodA(19),mARNB4),
     . (mxnocA(20),lADAW4),(ifmodA(20),mADAW4),
     . (mxnocA(21),lARRC4),(ifmodA(21),mARRC4),
     . (mxnocA(22),lAASI4),(ifmodA(22),mAASI4),
     . (mxnocA(23),lANAWP),(ifmodA(23),mANAWP),
     . (mxnocA(24),lARNB5),(ifmodA(24),mARNB5),
     . (mxnocA(25),lADAW5),(ifmodA(25),mADAW5),
     . (mxnocA(26),lARRC5),(ifmodA(26),mARRC5),
     . (mxnocA(27),lAASI5),(ifmodA(27),mAASI5),
     . (mxnocA(28),lANATS),(ifmodA(28),mANATS),
     . (mxnocA(29),lARNB6),(ifmodA(29),mARNB6),
     . (mxnocA(30),lADAW6),(ifmodA(30),mADAW6),
     . (mxnocA(31),lARRC6),(ifmodA(31),mARRC6),
     . (mxnocA(32),lAASI6),(ifmodA(32),mAASI6),
     . (mxnocA(33),lANAO),(ifmodA(33),mANAO),
     . (mxnocA(34),lARNB7),(ifmodA(34),mARNB7),
     . (mxnocA(35),lADAW7),(ifmodA(35),mADAW7),
     . (mxnocA(36),lARRC7),(ifmodA(36),mARRC7),
     . (mxnocA(37),lAASI7),(ifmodA(37),mAASI7)
C
C    Database table and attribute names for screen fields.
C
      data atrbtA/
     . 'ALPHA_URANIUM','SAMPLE_TRACKING_ID',
     . 'SAMPLE_INFORMATION','CUSTOMER_SAMPLE_NAME',
     . 'SAMPLE_INFORMATION','NEED_ALPHA_URANIUM',
     . 'ALPHA_URANIUM','RESULTS_NEEDED_BY_DATE',
     . 'ALPHA_URANIUM','DATE_ALL_WORK_COMPLETED',
     . 'ALPHA_URANIUM','RESULTS_REPORT_CITATION',
     . 'ALPHA_URANIUM','ANY_SPECIAL_INSTRUCTIONS',
     . 'SAMPLE_INFORMATION','NEED_ALPHA_THORIUM',
     . 'ALPHA_THORIUM','RESULTS_NEEDED_BY_DATE',
     . 'ALPHA_THORIUM','DATE_ALL_WORK_COMPLETED',
     . 'ALPHA_THORIUM','RESULTS_REPORT_CITATION',
     . 'ALPHA_THORIUM','ANY_SPECIAL_INSTRUCTIONS',
     . 'SAMPLE_INFORMATION','NEED_ALPHA_PLUTONIUM',
     . 'ALPHA_PLUTONIUM','RESULTS_NEEDED_BY_DATE',
     . 'ALPHA_PLUTONIUM','DATE_ALL_WORK_COMPLETED',
     . 'ALPHA_PLUTONIUM','RESULTS_REPORT_CITATION',
     . 'ALPHA_PLUTONIUM','ANY_SPECIAL_INSTRUCTIONS',
```

```
     . 'SAMPLE_INFORMATION','NEED_ALPHA_AM241_SANS_PU238',
     . 'ALPHA_AM241_SANS_PU238','RESULTS_NEEDED_BY_DATE',
     . 'ALPHA_AM241_SANS_PU238','DATE_ALL_WORK_COMPLETED',
     . 'ALPHA_AM241_SANS_PU238','RESULTS_REPORT_CITATION',
     . 'ALPHA_AM241_SANS_PU238','ANY_SPECIAL_INSTRUCTIONS',
     . 'SAMPLE_INFORMATION','NEED_ALPHA_AM241_WITH_PU238',
     . 'ALPHA_AM241_WITH_PU238','RESULTS_NEEDED_BY_DATE',
     . 'ALPHA_AM241_WITH_PU238','DATE_ALL_WORK_COMPLETED',
     . 'ALPHA_AM241_WITH_PU238','RESULTS_REPORT_CITATION',
     . 'ALPHA_AM241_WITH_PU238','ANY_SPECIAL_INSTRUCTIONS',
     . 'SAMPLE_INFORMATION','NEED_ALPHA_TOTAL_SPECTROMETRIC',
     . 'ALPHA_TOTAL_SPECTROMETRIC','RESULTS_NEEDED_BY_DATE',
     . 'ALPHA_TOTAL_SPECTROMETRIC','DATE_ALL_WORK_COMPLETED',
     . 'ALPHA_TOTAL_SPECTROMETRIC','RESULTS_REPORT_CITATION',
     . 'ALPHA_TOTAL_SPECTROMETRIC','ANY_SPECIAL_INSTRUCTIONS',
     . 'SAMPLE_INFORMATION','NEED_ALPHA_OTHER',
     . 'ALPHA_OTHER','RESULTS_NEEDED_BY_DATE',
     . 'ALPHA_OTHER','DATE_ALL_WORK_COMPLETED',
     . 'ALPHA_OTHER','RESULTS_REPORT_CITATION',
     . 'ALPHA_OTHER','ANY_SPECIAL_INSTRUCTIONS'/
C
C    Program variable names (value and null indicator) for screen fields.
C
      data varblA/':fASTI:nASTI',':fACSN:nACSN',':fANAU:nANAU',
     . ':iARNB1:nARNB1',':iADAW1:nADAW1',':fARRC1:nARRC1',
     . ':fAASI1:nAASI1',':fANATh:nANATh',':iARNB2:nARNB2',
     . ':iADAW2:nADAW2',':fARRC2:nARRC2',':fAASI2:nAASI2',
     . ':fANAPu:nANAPu',':iARNB3:nARNB3',':iADAW3:nADAW3',
     . ':fARRC3:nARRC3',':fAASI3:nAASI3',':fANASP:nANASP',
     . ':iARNB4:nARNB4',':iADAW4:nADAW4',':fARRC4:nARRC4',
     . ':fAASI4:nAASI4',':fANAWP:nANAWP',':iARNB5:nARNB5',
     . ':iADAW5:nADAW5',':fARRC5:nARRC5',':fAASI5:nAASI5',
     . ':fANATS:nANATS',':iARNB6:nARNB6',':iADAW6:nADAW6',
     . ':fARRC6:nARRC6',':fAASI6:nAASI6',':fANAO:nANAO',
     . ':iARNB7:nARNB7',':iADAW7:nADAW7',':fARRC7:nARRC7',
     . ':fAASI7:nAASI7'/
C
C    Beta screen's field descriptors and field sizes.
C
      character*80 fieldB(46),fidatB(46)
      character*30 atrbtB(2,46)
      character*14 varblB(46)
      character*1 fitypB(46)
      integer*4 fdlenB(46),fdrowB(46),fdcolB(46),ficolB(46),mxnocB(46),
     . fdrndB(3,46),firndB(3,46), !Field renditions (add/update/search).
     . numfdB/46/   !The number of fields.
      integer*2 ifnulB(46),   !Null indicator (-1 if null, 0 otherwise).
     . ifrqdB(46)/46*0/,   !Points to if-required dependent field.
     . iditoB(46)/46*0/   !Points to allowed field for dittoing.
      logical ifmodB(46)   !Has a field been modified in an Update?
      common /dbdatB/ fieldB,fidatB,fdlenB,fdrowB,fdcolB,ficolB,mxnocB,
     . fdrndB,firndB,numfdB,fitypB,ifnulB,atrbtB,varblB,ifrqdB,ifmodB,
     . iditoB
C
C      Length-specified character fields for equivalences (f for field).
C
      character fBSTI*12,fBCSN*60,fBNBSr*1,fBDCL1*4,fBDCU1*1,fBRND1*6,
     . fBRNT1*5,fBDAW1*6,fBRRC1*20,fBASI1*1,fBNBSC*1,fBDCL2*4,fBDCU2*1,
     . fBRND2*6,fBRNT2*5,fBDAW2*6,fBRRC2*20,fBASI2*1,fBNBST*1,fBDCL3*4,
     . fBDCU3*1,fBRND3*6,fBRNT3*5,fBDAW3*6,fBRRC3*20,fBASI3*1,fBNBT*1,
```

```
     . fBDCL4*4,fBDCU4*1,fBRND4*6,fBRNT4*5,fBDAW4*6,fBRRC4*20,fBASI4*1,
     . fBNBO*1,fBDCL5*4,fBDCU5*1,fBNBNi*1,fBNBFe*1,fBNBS*1,fBNBPu*1,
     . fBRND5*6,fBRNT5*5,fBDAW5*6,fBRRC5*20,fBASI5*1
C
C       Real and integer field storage (r for real, i for integer).
C
      real*4 rBDCL1,rBDCL2,rBDCL3,rBDCL4,rBDCL5
      integer*4 iBRND1,iBDAW1,iBRND2,iBDAW2,iBRND3,iBDAW3,iBRND4,iBDAW4,
     . iBRND5,iBDAW5
C
C       Nonarrayed maximum field lengths (l for length).
C
      integer*4 lBSTI,lBCSN,lBNBSr,lBDCL1,lBDCU1,lBRND1,lBRNT1,lBDAW1,
     . lBRRC1,lBASI1,lBNBSC,lBDCL2,lBDCU2,lBRND2,lBRNT2,lBDAW2,lBRRC2,
     . lBASI2,lBNBST,lBDCL3,lBDCU3,lBRND3,lBRNT3,lBDAW3,lBRRC3,lBASI3,
     . lBNBT,lBDCL4,lBDCU4,lBRND4,lBRNT4,lBDAW4,lBRRC4,lBASI4,lBNBO,
     . lBDCL5,lBDCU5,lBNBNi,lBNBFe,lBNBS,lBNBPu,lBRND5,lBRNT5,lBDAW5,
     . lBRRC5,lBASI5
C
C       Nonarrayed null indicators for data transfers (n for null).
C
      integer*2 nBSTI,nBCSN,nBNBSr,nBDCL1,nBDCU1,nBRND1,nBRNT1,nBDAW1,
     . nBRRC1,nBASI1,nBNBSC,nBDCL2,nBDCU2,nBRND2,nBRNT2,nBDAW2,nBRRC2,
     . nBASI2,nBNBST,nBDCL3,nBDCU3,nBRND3,nBRNT3,nBDAW3,nBRRC3,nBASI3,
     . nBNBT,nBDCL4,nBDCU4,nBRND4,nBRNT4,nBDAW4,nBRRC4,nBASI4,nBNBO,
     . nBDCL5,nBDCU5,nBNBNi,nBNBFe,nBNBS,nBNBPu,nBRND5,nBRNT5,nBDAW5,
     . nBRRC5,nBASI5
C
C       Nonarrayed if-modified indicators (m for modified).
C
      integer*2 mBSTI,mBCSN,mBNBSr,mBDCL1,mBDCU1,mBRND1,mBRNT1,mBDAW1,
     . mBRRC1,mBASI1,mBNBSC,mBDCL2,mBDCU2,mBRND2,mBRNT2,mBDAW2,mBRRC2,
     . mBASI2,mBNBST,mBDCL3,mBDCU3,mBRND3,mBRNT3,mBDAW3,mBRRC3,mBASI3,
     . mBNBT,mBDCL4,mBDCU4,mBRND4,mBRNT4,mBDAW4,mBRRC4,mBASI4,mBNBO,
     . mBDCL5,mBDCU5,mBNBNi,mBNBFe,mBNBS,mBNBPu,mBRND5,mBRNT5,mBDAW5,
     . mBRRC5,mBASI5
C
C       Equivalence statements for data transfers.
C
      equivalence (fidatB(1)(1:12),fBSTI),(ifnulB(1),nBSTI),
     . (fidatB(2)(1:60),fBCSN),(ifnulB(2),nBCSN),
     . (fidatB(3)(1:1),fBNBSr),(ifnulB(3),nBNBSr),
     . (fidatB(4)(1:4),fBDCL1),(ifnulB(4),nBDCL1),
     . (fidatB(5)(1:1),fBDCU1),(ifnulB(5),nBDCU1),
     . (fidatB(6)(1:6),fBRND1),(ifnulB(6),nBRND1),
     . (fidatB(7)(1:5),fBRNT1),(ifnulB(7),nBRNT1),
     . (fidatB(8)(1:6),fBDAW1),(ifnulB(8),nBDAW1),
     . (fidatB(9)(1:20),fBRRC1),(ifnulB(9),nBRRC1),
     . (fidatB(10)(1:1),fBASI1),(ifnulB(10),nBASI1),
     . (fidatB(11)(1:1),fBNBSC),(ifnulB(11),nBNBSC),
     . (fidatB(12)(1:4),fBDCL2),(ifnulB(12),nBDCL2),
     . (fidatB(13)(1:1),fBDCU2),(ifnulB(13),nBDCU2),
     . (fidatB(14)(1:6),fBRND2),(ifnulB(14),nBRND2),
     . (fidatB(15)(1:5),fBRNT2),(ifnulB(15),nBRNT2),
     . (fidatB(16)(1:6),fBDAW2),(ifnulB(16),nBDAW2),
     . (fidatB(17)(1:20),fBRRC2),(ifnulB(17),nBRRC2),
     . (fidatB(18)(1:1),fBASI2),(ifnulB(18),nBASI2),
     . (fidatB(19)(1:1),fBNBST),(ifnulB(19),nBNBST),
     . (fidatB(20)(1:4),fBDCL3),(ifnulB(20),nBDCL3),
     . (fidatB(21)(1:1),fBDCU3),(ifnulB(21),nBDCU3),
```

```
. (fidatB(22)(1:6),fBRND3),(ifnulB(22),nBRND3),
. (fidatB(23)(1:5),fBRNT3),(ifnulB(23),nBRNT3),
. (fidatB(24)(1:6),fBDAW3),(ifnulB(24),nBDAW3),
. (fidatB(25)(1:20),fBRRC3),(ifnulB(25),nBRRC3),
. (fidatB(26)(1:1),fBASI3),(ifnulB(26),nBASI3),
. (fidatB(27)(1:1),fBNBT),(ifnulB(27),nBNBT),
. (fidatB(28)(1:4),fBDCL4),(ifnulB(28),nBDCL4),
. (fidatB(29)(1:1),fBDCU4),(ifnulB(29),nBDCU4),
. (fidatB(30)(1:6),fBRND4),(ifnulB(30),nBRND4),
. (fidatB(31)(1:5),fBRNT4),(ifnulB(31),nBRNT4),
. (fidatB(32)(1:6),fBDAW4),(ifnulB(32),nBDAW4),
. (fidatB(33)(1:20),fBRRC4),(ifnulB(33),nBRRC4),
. (fidatB(34)(1:1),fBASI4),(ifnulB(34),nBASI4),
. (fidatB(35)(1:1),fBNBO),(ifnulB(35),nBNBO),
. (fidatB(36)(1:4),fBDCL5),(ifnulB(36),nBDCL5),
. (fidatB(37)(1:1),fBDCU5),(ifnulB(37),nBDCU5),
. (fidatB(38)(1:1),fBNBNi),(ifnulB(38),nBNBNi),
. (fidatB(39)(1:1),fBNBFe),(ifnulB(39),nBNBFe),
. (fidatB(40)(1:1),fBNBS),(ifnulB(40),nBNBS),
. (fidatB(41)(1:1),fBNBPu),(ifnulB(41),nBNBPu),
. (fidatB(42)(1:6),fBRND5),(ifnulB(42),nBRND5),
. (fidatB(43)(1:5),fBRNT5),(ifnulB(43),nBRNT5),
. (fidatB(44)(1:6),fBDAW5),(ifnulB(44),nBDAW5),
. (fidatB(45)(1:20),fBRRC5),(ifnulB(45),nBRRC5),
. (fidatB(46)(1:1),fBASI5),(ifnulB(46),nBASI5),
. (mxnocB(1),lBSTI),(ifmodB(1),mBSTI),
. (mxnocB(2),lBCSN),(ifmodB(2),mBCSN),
. (mxnocB(3),lBNBSr),(ifmodB(3),mBNBSr),
. (mxnocB(4),lBDCL1),(ifmodB(4),mBDCL1),
. (mxnocB(5),lBDCU1),(ifmodB(5),mBDCU1),
. (mxnocB(6),lBRND1),(ifmodB(6),mBRND1),
. (mxnocB(7),lBRNT1),(ifmodB(7),mBRNT1),
. (mxnocB(8),lBDAW1),(ifmodB(8),mBDAW1),
. (mxnocB(9),lBRRC1),(ifmodB(9),mBRRC1),
. (mxnocB(10),lBASI1),(ifmodB(10),mBASI1),
. (mxnocB(11),lBNBSC),(ifmodB(11),mBNBSC),
. (mxnocB(12),lBDCL2),(ifmodB(12),mBDCL2),
. (mxnocB(13),lBDCU2),(ifmodB(13),mBDCU2),
. (mxnocB(14),lBRND2),(ifmodB(14),mBRND2),
. (mxnocB(15),lBRNT2),(ifmodB(15),mBRNT2),
. (mxnocB(16),lBDAW2),(ifmodB(16),mBDAW2),
. (mxnocB(17),lBRRC2),(ifmodB(17),mBRRC2),
. (mxnocB(18),lBASI2),(ifmodB(18),mBASI2),
. (mxnocB(19),lBNBST),(ifmodB(19),mBNBST),
. (mxnocB(20),lBDCL3),(ifmodB(20),mBDCL3),
. (mxnocB(21),lBDCU3),(ifmodB(21),mBDCU3),
. (mxnocB(22),lBRND3),(ifmodB(22),mBRND3),
. (mxnocB(23),lBRNT3),(ifmodB(23),mBRNT3),
. (mxnocB(24),lBDAW3),(ifmodB(24),mBDAW3),
. (mxnocB(25),lBRRC3),(ifmodB(25),mBRRC3),
. (mxnocB(26),lBASI3),(ifmodB(26),mBASI3),
. (mxnocB(27),lBNBT),(ifmodB(27),mBNBT),
. (mxnocB(28),lBDCL4),(ifmodB(28),mBDCL4),
. (mxnocB(29),lBDCU4),(ifmodB(29),mBDCU4),
. (mxnocB(30),lBRND4),(ifmodB(30),mBRND4),
. (mxnocB(31),lBRNT4),(ifmodB(31),mBRNT4),
. (mxnocB(32),lBDAW4),(ifmodB(32),mBDAW4),
. (mxnocB(33),lBRRC4),(ifmodB(33),mBRRC4),
. (mxnocB(34),lBASI4),(ifmodB(34),mBASI4),
. (mxnocB(35),lBNBO),(ifmodB(35),mBNBO),
```

```
     . (mxnocB(36),lBDCL5),(ifmodB(36),mBDCL5),
     . (mxnocB(37),lBDCU5),(ifmodB(37),mBDCU5),
     . (mxnocB(38),lBNBNi),(ifmodB(38),mBNBNi),
     . (mxnocB(39),lBNBFe),(ifmodB(39),mBNBFe),
     . (mxnocB(40),lBNBS),(ifmodB(40),mBNBS),
     . (mxnocB(41),lBNBPu),(ifmodB(41),mBNBPu),
     . (mxnocB(42),lBRND5),(ifmodB(42),mBRND5),
     . (mxnocB(43),lBRNT5),(ifmodB(43),mBRNT5),
     . (mxnocB(44),lBDAW5),(ifmodB(44),mBDAW5),
     . (mxnocB(45),lBRRC5),(ifmodB(45),mBRRC5),
     . (mxnocB(46),lBASI5),(ifmodB(46),mBASI5)
C
C    Database table and attribute names for screen fields.
C
      data atrbtB/
     . 'BETA_STRONTIUM_90','SAMPLE_TRACKING_ID',
     . 'SAMPLE_INFORMATION','CUSTOMER_SAMPLE_NAME',
     . 'SAMPLE_INFORMATION','NEED_BETA_STRONTIUM_90',
     . 'BETA_STRONTIUM_90','DETECTOR_COUNT_LENGTH',
     . 'BETA_STRONTIUM_90','DETECTOR_COUNT_LENGTH_UNITS',
     . 'BETA_STRONTIUM_90','RESULTS_NEEDED_BY_DATE',
     . 'BETA_STRONTIUM_90','RESULTS_NEEDED_BY_TIME',
     . 'BETA_STRONTIUM_90','DATE_ALL_WORK_COMPLETED',
     . 'BETA_STRONTIUM_90','RESULTS_REPORT_CITATION',
     . 'BETA_STRONTIUM_90','ANY_SPECIAL_INSTRUCTIONS',
     . 'SAMPLE_INFORMATION','NEED_BETA_STRONTIUM_89_AND_90',
     . 'BETA_STRONTIUM_89_AND_90','DETECTOR_COUNT_LENGTH',
     . 'BETA_STRONTIUM_89_AND_90','DETECTOR_COUNT_LENGTH_UNITS',
     . 'BETA_STRONTIUM_89_AND_90','RESULTS_NEEDED_BY_DATE',
     . 'BETA_STRONTIUM_89_AND_90','RESULTS_NEEDED_BY_TIME',
     . 'BETA_STRONTIUM_89_AND_90','DATE_ALL_WORK_COMPLETED',
     . 'BETA_STRONTIUM_89_AND_90','RESULTS_REPORT_CITATION',
     . 'BETA_STRONTIUM_89_AND_90','ANY_SPECIAL_INSTRUCTIONS',
     . 'SAMPLE_INFORMATION','NEED_BETA_TOTAL_STRONTIUM',
     . 'BETA_TOTAL_STRONTIUM','DETECTOR_COUNT_LENGTH',
     . 'BETA_TOTAL_STRONTIUM','DETECTOR_COUNT_LENGTH_UNITS',
     . 'BETA_TOTAL_STRONTIUM','RESULTS_NEEDED_BY_DATE',
     . 'BETA_TOTAL_STRONTIUM','RESULTS_NEEDED_BY_TIME',
     . 'BETA_TOTAL_STRONTIUM','DATE_ALL_WORK_COMPLETED',
     . 'BETA_TOTAL_STRONTIUM','RESULTS_REPORT_CITATION',
     . 'BETA_TOTAL_STRONTIUM','ANY_SPECIAL_INSTRUCTIONS',
     . 'SAMPLE_INFORMATION','NEED_BETA_TRITIUM',
     . 'BETA_TRITIUM','DETECTOR_COUNT_LENGTH',
     . 'BETA_TRITIUM','DETECTOR_COUNT_LENGTH_UNITS',
     . 'BETA_TRITIUM','RESULTS_NEEDED_BY_DATE',
     . 'BETA_TRITIUM','RESULTS_NEEDED_BY_TIME',
     . 'BETA_TRITIUM','DATE_ALL_WORK_COMPLETED',
     . 'BETA_TRITIUM','RESULTS_REPORT_CITATION',
     . 'BETA_TRITIUM','ANY_SPECIAL_INSTRUCTIONS',
     . 'SAMPLE_INFORMATION','NEED_BETA_OTHER',
     . 'BETA_OTHER','DETECTOR_COUNT_LENGTH',
     . 'BETA_OTHER','DETECTOR_COUNT_LENGTH_UNITS',
     . 'BETA_OTHER','NEED_BETA_NICKEL_63',
     . 'BETA_OTHER','NEED_BETA_IRON_55',
     . 'BETA_OTHER','NEED_BETA_SULFUR_35',
     . 'BETA_OTHER','NEED_BETA_PLUTONIUM_241',
     . 'BETA_OTHER','RESULTS_NEEDED_BY_DATE',
     . 'BETA_OTHER','RESULTS_NEEDED_BY_TIME',
     . 'BETA_OTHER','DATE_ALL_WORK_COMPLETED',
     . 'BETA_OTHER','RESULTS_REPORT_CITATION',
```

```
     . 'BETA_OTHER','ANY_SPECIAL_INSTRUCTIONS'/
C
C    Program variable names (value and null indicator) for screen fields.
C
      data varblB/':fBSTI:nBSTI',':fBCSN:nBCSN',':fBNBSr:nBNBSr',
     . ':rBDCL1:nBDCL1',':fBDCU1:nBDCU1',':iBRND1:nBRND1',
     . ':fBRNT1:nBRNT1',':iBDAW1:nBDAW1',':fBRRC1:nBRRC1',
     . ':fBASI1:nBASI1',':fBNBSC:nBNBSC',':rBDCL2:nBDCL2',
     . ':fBDCU2:nBDCU2',':iBRND2:nBRND2',':fBRNT2:nBRNT2',
     . ':iBDAW2:nBDAW2',':fBRRC2:nBRRC2',':fBASI2:nBASI2',
     . ':fBNBST:nBNBST',':rBDCL3:nBDCL3',':fBDCU3:nBDCU3',
     . ':iBRND3:nBRND3',':fBRNT3:nBRNT3',':iBDAW3:nBDAW3',
     . ':fBRRC3:nBRRC3',':fBASI3:nBASI3',':fBNBT:nBNBT',
     . ':rBDCL4:nBDCL4',':fBDCU4:nBDCU4',':iBRND4:nBRND4',
     . ':fBRNT4:nBRNT4',':iBDAW4:nBDAW4',':fBRRC4:nBRRC4',
     . ':fBASI4:nBASI4',':fBNBO:nBNBO',':rBDCL5:nBDCL5',
     . ':fBDCU5:nBDCU5',':fBNBNi:nBNBNi',':fBNBFe:nBNBFe',
     . ':fBNBS:nBNBS',':fBNBPu:nBNBPu',':iBRND5:nBRND5',
     . ':fBRNT5:nBRNT5',':iBDAW5:nBDAW5',':fBRRC5:nBRRC5',
     . ':fBASI5:nBASI5'/
C
C    Gamma screen's field descriptors and field sizes.
C
      character*80 fieldG(35),fidatG(35)
      character*30 atrbtG(2,35)
      character*14 varblG(35)
      character*1 fitypG(35)
      integer*4 fdlenG(35),fdrowG(35),fdcolG(35),ficolG(35),mxnocG(35),
     . fdrndG(3,35),firndG(3,35), !Field renditions (add/update/search).
     . numfdG/35/   !The number of fields.
      integer*2 ifnulG(35),   !Null indicator (-1 if null, 0 otherwise).
     . ifrqdG(35)/35*0/,   !Points to if-required dependent field.
     . iditoG(35)/35*0/   !Points to allowed field for dittoing.
      logical ifmodG(35)   !Has a field been modified in an Update?
      common /dbdatG/ fieldG,fidatG,fdlenG,fdrowG,fdcolG,ficolG,mxnocG,
     . fdrndG,firndG,numfdG,fitypG,ifnulG,atrbtG,varblG,ifrqdG,ifmodG,
     . iditoG
C
C      Length-specified character fields for equivalences (f for field).
C
      character fGSTI*12,fGCSN*60,fGNGS*1,fGDCL1*4,fGDCU1*1,fGRND1*6,
     . fGRNT1*5,fGDAW1*6,fGDSC1*6,fGRSI1*14,fGITA1*1,fGRRC1*20,fGASI1*1,
     . fGNGFI*1,fGDCL2*4,fGDCU2*1,fGRND2*6,fGRNT2*5,fGDAW2*6,fGDSC2*6,
     . fGRSI2*14,fGITA2*1,fGRRC2*20,fGASI2*1,fGNGO*1,fGDCL3*4,fGDCU3*1,
     . fGRND3*6,fGRNT3*5,fGDAW3*6,fGDSC3*6,fGRSI3*14,fGITA3*1,fGRRC3*20,
     . fGASI3*1
C
C       Real and integer field storage (r for real, i for integer).
C
      real*4 rGDCL1,rGDCL2,rGDCL3
      integer*4 iGRND1,iGDAW1,iGDSC1,iGRND2,iGDAW2,iGDSC2,iGRND3,iGDAW3,
     . iGDSC3
C
C       Nonarrayed maximum field lengths (l for length).
C
      integer*4 lGSTI,lGCSN,lGNGS,lGDCL1,lGDCU1,lGRND1,lGRNT1,lGDAW1,
     . lGDSC1,lGRSI1,lGITA1,lGRRC1,lGASI1,lGNGFI,lGDCL2,lGDCU2,lGRND2,
     . lGRNT2,lGDAW2,lGDSC2,lGRSI2,lGITA2,lGRRC2,lGASI2,lGNGO,lGDCL3,
     . lGDCU3,lGRND3,lGRNT3,lGDAW3,lGDSC3,lGRSI3,lGITA3,lGRRC3,lGASI3
C
```

```
C       Nonarrayed null indicators for data transfers (n for null).
C
      integer*2 nGSTI,nGCSN,nGNGS,nGDCL1,nGDCU1,nGRND1,nGRNT1,nGDAW1,
     . nGDSC1,nGRSI1,nGITA1,nGRRC1,nGASI1,nGNGFI,nGDCL2,nGDCU2,nGRND2,
     . nGRNT2,nGDAW2,nGDSC2,nGRSI2,nGITA2,nGRRC2,nGASI2,nGNGO,nGDCL3,
     . nGDCU3,nGRND3,nGRNT3,nGDAW3,nGDSC3,nGRSI3,nGITA3,nGRRC3,nGASI3
C
C       Nonarrayed if-modified indicators (m for modified).
C
      integer*2 mGSTI,mGCSN,mGNGS,mGDCL1,mGDCU1,mGRND1,mGRNT1,mGDAW1,
     . mGDSC1,mGRSI1,mGITA1,mGRRC1,mGASI1,mGNGFI,mGDCL2,mGDCU2,mGRND2,
     . mGRNT2,mGDAW2,mGDSC2,mGRSI2,mGITA2,mGRRC2,mGASI2,mGNGO,mGDCL3,
     . mGDCU3,mGRND3,mGRNT3,mGDAW3,mGDSC3,mGRSI3,mGITA3,mGRRC3,mGASI3
C
C       Equivalence statements for data transfers.
C
      equivalence (fidatG(1)(1:12),fGSTI),(ifnulG(1),nGSTI),
     . (fidatG(2)(1:60),fGCSN),(ifnulG(2),nGCSN),
     . (fidatG(3)(1:1),fGNGS),(ifnulG(3),nGNGS),
     . (fidatG(4)(1:4),fGDCL1),(ifnulG(4),nGDCL1),
     . (fidatG(5)(1:1),fGDCU1),(ifnulG(5),nGDCU1),
     . (fidatG(6)(1:6),fGRND1),(ifnulG(6),nGRND1),
     . (fidatG(7)(1:5),fGRNT1),(ifnulG(7),nGRNT1),
     . (fidatG(8)(1:6),fGDAW1),(ifnulG(8),nGDAW1),
     . (fidatG(9)(1:6),fGDSC1),(ifnulG(9),nGDSC1),
     . (fidatG(10)(1:14),fGRSI1),(ifnulG(10),nGRSI1),
     . (fidatG(11)(1:1),fGITA1),(ifnulG(11),nGITA1),
     . (fidatG(12)(1:20),fGRRC1),(ifnulG(12),nGRRC1),
     . (fidatG(13)(1:1),fGASI1),(ifnulG(13),nGASI1),
     . (fidatG(14)(1:1),fGNGFI),(ifnulG(14),nGNGFI),
     . (fidatG(15)(1:4),fGDCL2),(ifnulG(15),nGDCL2),
     . (fidatG(16)(1:1),fGDCU2),(ifnulG(16),nGDCU2),
     . (fidatG(17)(1:6),fGRND2),(ifnulG(17),nGRND2),
     . (fidatG(18)(1:5),fGRNT2),(ifnulG(18),nGRNT2),
     . (fidatG(19)(1:6),fGDAW2),(ifnulG(19),nGDAW2),
     . (fidatG(20)(1:6),fGDSC2),(ifnulG(20),nGDSC2),
     . (fidatG(21)(1:14),fGRSI2),(ifnulG(21),nGRSI2),
     . (fidatG(22)(1:1),fGITA2),(ifnulG(22),nGITA2),
     . (fidatG(23)(1:20),fGRRC2),(ifnulG(23),nGRRC2),
     . (fidatG(24)(1:1),fGASI2),(ifnulG(24),nGASI2),
     . (fidatG(25)(1:1),fGNGO),(ifnulG(25),nGNGO),
     . (fidatG(26)(1:4),fGDCL3),(ifnulG(26),nGDCL3),
     . (fidatG(27)(1:1),fGDCU3),(ifnulG(27),nGDCU3),
     . (fidatG(28)(1:6),fGRND3),(ifnulG(28),nGRND3),
     . (fidatG(29)(1:5),fGRNT3),(ifnulG(29),nGRNT3),
     . (fidatG(30)(1:6),fGDAW3),(ifnulG(30),nGDAW3),
     . (fidatG(31)(1:6),fGDSC3),(ifnulG(31),nGDSC3),
     . (fidatG(32)(1:14),fGRSI3),(ifnulG(32),nGRSI3),
     . (fidatG(33)(1:1),fGITA3),(ifnulG(33),nGITA3),
     . (fidatG(34)(1:20),fGRRC3),(ifnulG(34),nGRRC3),
     . (fidatG(35)(1:1),fGASI3),(ifnulG(35),nGASI3),
     . (mxnocG(1),lGSTI),(ifmodG(1),mGSTI),
     . (mxnocG(2),lGCSN),(ifmodG(2),mGCSN),
     . (mxnocG(3),lGNGS),(ifmodG(3),mGNGS),
     . (mxnocG(4),lGDCL1),(ifmodG(4),mGDCL1),
     . (mxnocG(5),lGDCU1),(ifmodG(5),mGDCU1),
     . (mxnocG(6),lGRND1),(ifmodG(6),mGRND1),
     . (mxnocG(7),lGRNT1),(ifmodG(7),mGRNT1),
     . (mxnocG(8),lGDAW1),(ifmodG(8),mGDAW1),
     . (mxnocG(9),lGDSC1),(ifmodG(9),mGDSC1),
```

```
     . (mxnocG(10),lGRSI1),(ifmodG(10),mGRSI1),
     . (mxnocG(11),lGITA1),(ifmodG(11),mGITA1),
     . (mxnocG(12),lGRRC1),(ifmodG(12),mGRRC1),
     . (mxnocG(13),lGASI1),(ifmodG(13),mGASI1),
     . (mxnocG(14),lGNGFI),(ifmodG(14),mGNGFI),
     . (mxnocG(15),lGDCL2),(ifmodG(15),mGDCL2),
     . (mxnocG(16),lGDCU2),(ifmodG(16),mGDCU2),
     . (mxnocG(17),lGRND2),(ifmodG(17),mGRND2),
     . (mxnocG(18),lGRNT2),(ifmodG(18),mGRNT2),
     . (mxnocG(19),lGDAW2),(ifmodG(19),mGDAW2),
     . (mxnocG(20),lGDSC2),(ifmodG(20),mGDSC2),
     . (mxnocG(21),lGRSI2),(ifmodG(21),mGRSI2),
     . (mxnocG(22),lGITA2),(ifmodG(22),mGITA2),
     . (mxnocG(23),lGRRC2),(ifmodG(23),mGRRC2),
     . (mxnocG(24),lGASI2),(ifmodG(24),mGASI2),
     . (mxnocG(25),lGNGO),(ifmodG(25),mGNGO),
     . (mxnocG(26),lGDCL3),(ifmodG(26),mGDCL3),
     . (mxnocG(27),lGDCU3),(ifmodG(27),mGDCU3),
     . (mxnocG(28),lGRND3),(ifmodG(28),mGRND3),
     . (mxnocG(29),lGRNT3),(ifmodG(29),mGRNT3),
     . (mxnocG(30),lGDAW3),(ifmodG(30),mGDAW3),
     . (mxnocG(31),lGDSC3),(ifmodG(31),mGDSC3),
     . (mxnocG(32),lGRSI3),(ifmodG(32),mGRSI3),
     . (mxnocG(33),lGITA3),(ifmodG(33),mGITA3),
     . (mxnocG(34),lGRRC3),(ifmodG(34),mGRRC3),
     . (mxnocG(35),lGASI3),(ifmodG(35),mGASI3)
C
C    Database table and attribute names for screen fields.
C
      data atrbtG/
     . 'GAMMA_SCREEN','SAMPLE_TRACKING_ID',
     . 'SAMPLE_INFORMATION','CUSTOMER_SAMPLE_NAME',
     . 'SAMPLE_INFORMATION','NEED_GAMMA_SCREEN',
     . 'GAMMA_SCREEN','DETECTOR_COUNT_LENGTH',
     . 'GAMMA_SCREEN','DETECTOR_COUNT_LENGTH_UNITS',
     . 'GAMMA_SCREEN','RESULTS_NEEDED_BY_DATE',
     . 'GAMMA_SCREEN','RESULTS_NEEDED_BY_TIME',
     . 'GAMMA_SCREEN','DATE_ALL_WORK_COMPLETED',
     . 'GAMMA_SCREEN','DATE_SAMPLE_COUNTED',
     . 'GAMMA_SCREEN','RML_SPECTRAL_ID',
     . 'GAMMA_SCREEN','IS_THIS_A_SPECTRUM_RECOUNT',
     . 'GAMMA_SCREEN','RESULTS_REPORT_CITATION',
     . 'GAMMA_SCREEN','ANY_SPECIAL_INSTRUCTIONS',
     . 'SAMPLE_INFORMATION','NEED_GAMMA_FULL_ISOTOPIC',
     . 'GAMMA_FULL_ISOTOPIC','DETECTOR_COUNT_LENGTH',
     . 'GAMMA_FULL_ISOTOPIC','DETECTOR_COUNT_LENGTH_UNITS',
     . 'GAMMA_FULL_ISOTOPIC','RESULTS_NEEDED_BY_DATE',
     . 'GAMMA_FULL_ISOTOPIC','RESULTS_NEEDED_BY_TIME',
     . 'GAMMA_FULL_ISOTOPIC','DATE_ALL_WORK_COMPLETED',
     . 'GAMMA_FULL_ISOTOPIC','DATE_SAMPLE_COUNTED',
     . 'GAMMA_FULL_ISOTOPIC','RML_SPECTRAL_ID',
     . 'GAMMA_FULL_ISOTOPIC','IS_THIS_A_SPECTRUM_RECOUNT',
     . 'GAMMA_FULL_ISOTOPIC','RESULTS_REPORT_CITATION',
     . 'GAMMA_FULL_ISOTOPIC','ANY_SPECIAL_INSTRUCTIONS',
     . 'SAMPLE_INFORMATION','NEED_GAMMA_OTHER',
     . 'GAMMA_OTHER','DETECTOR_COUNT_LENGTH',
     . 'GAMMA_OTHER','DETECTOR_COUNT_LENGTH_UNITS',
     . 'GAMMA_OTHER','RESULTS_NEEDED_BY_DATE',
     . 'GAMMA_OTHER','RESULTS_NEEDED_BY_TIME',
     . 'GAMMA_OTHER','DATE_ALL_WORK_COMPLETED',
```

```
     . 'GAMMA_OTHER','DATE_SAMPLE_COUNTED',
     . 'GAMMA_OTHER','RML_SPECTRAL_ID',
     . 'GAMMA_OTHER','IS_THIS_A_SPECTRUM_RECOUNT',
     . 'GAMMA_OTHER','RESULTS_REPORT_CITATION',
     . 'GAMMA_OTHER','ANY_SPECIAL_INSTRUCTIONS'/
C
C    Program variable names (value and null indicator) for screen fields.
C
      data varblG/':fGSTI:nGSTI',':fGCSN:nGCSN',':fGNGS:nGNGS',
     . ':rGDCL1:nGDCL1',':fGDCU1:nGDCU1',':iGRND1:nGRND1',
     . ':fGRNT1:nGRNT1',':iGDAW1:nGDAW1',':iGDSC1:nGDSC1',
     . ':fGRSI1:nGRSI1',':fGITA1:nGITA1',':fGRRC1:nGRRC1',
     . ':fGASI1:nGASI1',':fGNGFI:nGNGFI',':rGDCL2:nGDCL2',
     . ':fGDCU2:nGDCU2',':iGRND2:nGRND2',':fGRNT2:nGRNT2',
     . ':iGDAW2:nGDAW2',':iGDSC2:nGDSC2',':fGRSI2:nGRSI2',
     . ':fGITA2:nGITA2',':fGRRC2:nGRRC2',':fGASI2:nGASI2',
     . ':fGNGO:nGNGO',':rGDCL3:nGDCL3',':fGDCU3:nGDCU3',
     . ':iGRND3:nGRND3',':fGRNT3:nGRNT3',':iGDAW3:nGDAW3',
     . ':iGDSC3:nGDSC3',':fGRSI3:nGRSI3',':fGITA3:nGITA3',
     . ':fGRRC3:nGRRC3',':fGASI3:nGASI3'/
C
C    Gross alpha-beta screen's field descriptors and field sizes.
C
      character*80 fieldC(18),fidatC(18)
      character*30 atrbtC(2,18)
      character*14 varblC(18)
      character*1 fitypC(18)
      integer*4 fdlenC(18),fdrowC(18),fdcolC(18),ficolC(18),mxnocC(18),
     . fdrndC(3,18),firndC(3,18), !Field renditions (add/update/search).
     . numfdC/18/   !The number of fields.
      integer*2 ifnulC(18),   !Null indicator (-1 if null, 0 otherwise).
     . ifrqdC(18)/18*0/,   !Points to if-required dependent field.
     . iditoC(18)/18*0/   !Points to allowed field for dittoing.
      logical ifmodC(18)   !Has a field been modified in an Update?
      common /dbdatC/ fieldC,fidatC,fdlenC,fdrowC,fdcolC,ficolC,mxnocC,
     . fdrndC,firndC,numfdC,fitypC,ifnulC,atrbtC,varblC,ifrqdC,ifmodC,
     . iditoC
C
C     Length-specified character fields for equivalences (1 for length).
C
      character fCSTI*12,fCCSN*60,fCNGAF*1,fCDCL1*4,fCDCU1*1,fCRND1*6,
     . fCRNT1*5,fCDAW1*6,fCRRC1*20,fCASI1*1,fCNGAO*1,fCDCL2*4,fCDCU2*1,
     . fCRND2*6,fCRNT2*5,fCDAW2*6,fCRRC2*20,fCASI2*1
C
C       Real and integer field storage (r for real, i for integer).
C
      real*4 rCDCL1,rCDCL2
      integer*4 iCRND1,iCDAW1,iCRND2,iCDAW2
C
C       Nonarrayed maximum field lengths (1 for length).
C
      integer*4 lCSTI,lCCSN,lCNGAF,lCDCL1,lCDCU1,lCRND1,lCRNT1,lCDAW1,
     . lCRRC1,lCASI1,lCNGAO,lCDCL2,lCDCU2,lCRND2,lCRNT2,lCDAW2,lCRRC2,
     . lCASI2
C
C       Nonarrayed null indicators for data transfers (n for null).
C
      integer*2 nCSTI,nCCSN,nCNGAF,nCDCL1,nCDCU1,nCRND1,nCRNT1,nCDAW1,
     . nCRRC1,nCASI1,nCNGAO,nCDCL2,nCDCU2,nCRND2,nCRNT2,nCDAW2,nCRRC2,
     . nCASI2
```

```
C
C       Nonarrayed if-modified indicators (m for modified).
C
      integer*2 mCSTI,mCCSN,mCNGAF,mCDCL1,mCDCU1,mCRND1,mCRNT1,mCDAW1,
     . mCRRC1,mCASI1,mCNGAO,mCDCL2,mCDCU2,mCRND2,mCRNT2,mCDAW2,mCRRC2,
     . mCASI2
C
C       Equivalence statements for data transfers.
C
      equivalence (fidatC(1)(1:12),fCSTI),(ifnulC(1),nCSTI),
     . (fidatC(2)(1:60),fCCSN),(ifnulC(2),nCCSN),
     . (fidatC(3)(1:1),fCNGAF),(ifnulC(3),nCNGAF),
     . (fidatC(4)(1:4),fCDCL1),(ifnulC(4),nCDCL1),
     . (fidatC(5)(1:1),fCDCU1),(ifnulC(5),nCDCU1),
     . (fidatC(6)(1:6),fCRND1),(ifnulC(6),nCRND1),
     . (fidatC(7)(1:5),fCRNT1),(ifnulC(7),nCRNT1),
     . (fidatC(8)(1:6),fCDAW1),(ifnulC(8),nCDAW1),
     . (fidatC(9)(1:20),fCRRC1),(ifnulC(9),nCRRC1),
     . (fidatC(10)(1:1),fCASI1),(ifnulC(10),nCASI1),
     . (fidatC(11)(1:1),fCNGAO),(ifnulC(11),nCNGAO),
     . (fidatC(12)(1:4),fCDCL2),(ifnulC(12),nCDCL2),
     . (fidatC(13)(1:1),fCDCU2),(ifnulC(13),nCDCU2),
     . (fidatC(14)(1:6),fCRND2),(ifnulC(14),nCRND2),
     . (fidatC(15)(1:5),fCRNT2),(ifnulC(15),nCRNT2),
     . (fidatC(16)(1:6),fCDAW2),(ifnulC(16),nCDAW2),
     . (fidatC(17)(1:20),fCRRC2),(ifnulC(17),nCRRC2),
     . (fidatC(18)(1:1),fCASI2),(ifnulC(18),nCASI2),
     . (mxnocC(1),lCSTI),(ifmodC(1),mCSTI),
     . (mxnocC(2),lCCSN),(ifmodC(2),mCCSN),
     . (mxnocC(3),lCNGAF),(ifmodC(3),mCNGAF),
     . (mxnocC(4),lCDCL1),(ifmodC(4),mCDCL1),
     . (mxnocC(5),lCDCU1),(ifmodC(5),mCDCU1),
     . (mxnocC(6),lCRND1),(ifmodC(6),mCRND1),
     . (mxnocC(7),lCRNT1),(ifmodC(7),mCRNT1),
     . (mxnocC(8),lCDAW1),(ifmodC(8),mCDAW1),
     . (mxnocC(9),lCRRC1),(ifmodC(9),mCRRC1),
     . (mxnocC(10),lCASI1),(ifmodC(10),mCASI1),
     . (mxnocC(11),lCNGAO),(ifmodC(11),mCNGAO),
     . (mxnocC(12),lCDCL2),(ifmodC(12),mCDCL2),
     . (mxnocC(13),lCDCU2),(ifmodC(13),mCDCU2),
     . (mxnocC(14),lCRND2),(ifmodC(14),mCRND2),
     . (mxnocC(15),lCRNT2),(ifmodC(15),mCRNT2),
     . (mxnocC(16),lCDAW2),(ifmodC(16),mCDAW2),
     . (mxnocC(17),lCRRC2),(ifmodC(17),mCRRC2),
     . (mxnocC(18),lCASI2),(ifmodC(18),mCASI2)
C
C    Database table and attribute names for screen fields.
C
      data atrbtC/
     . 'GROSS_ALPHA_BETA_AIR_FILTERS','SAMPLE_TRACKING_ID',
     . 'SAMPLE_INFORMATION','CUSTOMER_SAMPLE_NAME',
     . 'SAMPLE_INFORMATION','NEED_GROSS_ALPHA_BETA_FILTERS',
     . 'GROSS_ALPHA_BETA_AIR_FILTERS','DETECTOR_COUNT_LENGTH',
     . 'GROSS_ALPHA_BETA_AIR_FILTERS','DETECTOR_COUNT_LENGTH_UNITS',
     . 'GROSS_ALPHA_BETA_AIR_FILTERS','RESULTS_NEEDED_BY_DATE',
     . 'GROSS_ALPHA_BETA_AIR_FILTERS','RESULTS_NEEDED_BY_TIME',
     . 'GROSS_ALPHA_BETA_AIR_FILTERS','DATE_ALL_WORK_COMPLETED',
     . 'GROSS_ALPHA_BETA_AIR_FILTERS','RESULTS_REPORT_CITATION',
     . 'GROSS_ALPHA_BETA_AIR_FILTERS','ANY_SPECIAL_INSTRUCTIONS',
     . 'SAMPLE_INFORMATION','NEED_GROSS_ALPHA_BETA_OTHER',
```

```
     . 'GROSS_ALPHA_BETA_OTHER','DETECTOR_COUNT_LENGTH',
     . 'GROSS_ALPHA_BETA_OTHER','DETECTOR_COUNT_LENGTH_UNITS',
     . 'GROSS_ALPHA_BETA_OTHER','RESULTS_NEEDED_BY_DATE',
     . 'GROSS_ALPHA_BETA_OTHER','RESULTS_NEEDED_BY_TIME',
     . 'GROSS_ALPHA_BETA_OTHER','DATE_ALL_WORK_COMPLETED',
     . 'GROSS_ALPHA_BETA_OTHER','RESULTS_REPORT_CITATION',
     . 'GROSS_ALPHA_BETA_OTHER','ANY_SPECIAL_INSTRUCTIONS'/
C
C    Program variable names (value and null indicator) for screen fields.
C
      data varblC/':fCSTI:nCSTI',':fCCSN:nCCSN',':fCNGAF:nCNGAF',
     . ':rCDCL1:nCDCL1',':fCDCU1:nCDCU1',':iCRND1:nCRND1',
     . ':fCRNT1:nCRNT1',':iCDAW1:nCDAW1',':fCRRC1:nCRRC1',
     . ':fCASI1:nCASI1',':fCNGAO:nCNGAO',':rCDCL2:nCDCL2',
     . ':fCDCU2:nCDCU2',':iCRND2:nCRND2',':fCRNT2:nCRNT2',
     . ':iCDAW2:nCDAW2',':fCRRC2:nCRRC2',':fCASI2:nCASI2'/
C
C    Special instruction screen's field descriptors and field sizes.
C
      character*80 fieldI(32),fidatI(32)
      character*30 atrbtI(2,5)
      character*14 varblI(5)
      character*1 fitypI(32)
      integer*4 fdlenI(32),fdrowI(32),fdcolI(32),ficolI(32),mxnocI(32),
     . fdrndI(3,32),firndI(3,32), !Field renditions (add/update/search).
     . numfdI/32/    !The number of fields.
      integer*2 ifnulI(32),   !Null indicator (-1 if null, 0 otherwise).
     . ifrqdI(32)/32*0/,    !Points to if-required dependent field.
     . iditoI(32)/32*0/    !Points to allowed field for dittoing.
      logical ifmodI(32)    !Has a field been modified in an Update?
      common /dbdatI/ fieldI,fidatI,fdlenI,fdrowI,fdcolI,ficolI,mxnocI,
     . fdrndI,firndI,numfdI,fitypI,ifnulI,atrbtI,varblI,ifrqdI,ifmodI,
     . iditoI
C
C      Length-specified character fields for equivalences (f for field).
C
      character fISTI*12,fICSN*60,fIOS1*16,fISI1*50,fIOS2*16,fISI2*50,
     . fIOS3*16,fISI3*50,fIOS4*16,fISI4*50,fIOS5*16,fISI5*50,fIOS6*16,
     . fISI6*50,fIOS7*16,fISI7*50,fIOS8*16,fISI8*50,fIOS9*16,fISI9*50,
     . fIOS10*16,fISI10*50,fIOS11*16,fISI11*50,fIOS12*16,fISI12*50,
     . fIOS13*16,fISI13*50,fIOS14*16,fISI14*50,fIOS15*16,fISI15*50,
     . fIOS1t*16,fISI1t*50    !And spares for inserts and fetchs.
C
C       Integer field storage (i for integer).
C
C      integer*4 siline    !Declared in the main routine.
C
C       Nonarrayed maximum field lengths (l for length).
C
      integer*4 lISTI,lICSN,lIOS1,lISI1,lIOS2,lISI2,lIOS3,lISI3,lIOS4,
     . lISI4,lIOS5,lISI5,lIOS6,lISI6,lIOS7,lISI7,lIOS8,lISI8,lIOS9,
     . lISI9,lIOS10,lISI10,lIOS11,lISI11,lIOS12,lISI12,lIOS13,lISI13,
     . lIOS14,lISI14,lIOS15,lISI15,
     . lIOS1t,lISI1t    !And spares for inserts and fetchs.
C
C       Nonarrayed null indicators for data transfers (n for null).
C
      integer*2 nISTI,nICSN,nIOS1,nISI1,nIOS2,nISI2,nIOS3,nISI3,nIOS4,
     . nISI4,nIOS5,nISI5,nIOS6,nISI6,nIOS7,nISI7,nIOS8,nISI8,nIOS9,
     . nISI9,nIOS10,nISI10,nIOS11,nISI11,nIOS12,nISI12,nIOS13,nISI13,
```

```
     . nIOS14,nISI14,nIOS15,nISI15,
     . nIOS1t,nISI1t    !And spares for inserts and fetchs.
C
C       Nonarrayed if-modified indicators (m for modified).
C
      integer*2 mISTI,mICSN,mIOS1,mISI1,mIOS2,mISI2,mIOS3,mISI3,mIOS4,
     . mISI4,mIOS5,mISI5,mIOS6,mISI6,mIOS7,mISI7,mIOS8,mISI8,mIOS9,
     . mISI9,mIOS10,mISI10,mIOS11,mISI11,mIOS12,mISI12,mIOS13,mISI13,
     . mIOS14,mISI14,mIOS15,mISI15,
     . mIOS1t,mISI1t    !And spares for inserts and fetchs.
C
C       Equivalence statements for data transfers.
C
      equivalence (fidatI(1)(1:12),fISTI),(ifnulI(1),nISTI),
     . (fidatI(2)(1:60),fICSN),(ifnulI(2),nICSN),
     . (fidatI(3)(1:16),fIOS1),(ifnulI(3),nIOS1),
     . (fidatI(4)(1:50),fISI1),(ifnulI(4),nISI1),
     . (fidatI(5)(1:16),fIOS2),(ifnulI(5),nIOS2),
     . (fidatI(6)(1:50),fISI2),(ifnulI(6),nISI2),
     . (fidatI(7)(1:16),fIOS3),(ifnulI(7),nIOS3),
     . (fidatI(8)(1:50),fISI3),(ifnulI(8),nISI3),
     . (fidatI(9)(1:16),fIOS4),(ifnulI(9),nIOS4),
     . (fidatI(10)(1:50),fISI4),(ifnulI(10),nISI4),
     . (fidatI(11)(1:16),fIOS5),(ifnulI(11),nIOS5),
     . (fidatI(12)(1:50),fISI5),(ifnulI(12),nISI5),
     . (fidatI(13)(1:16),fIOS6),(ifnulI(13),nIOS6),
     . (fidatI(14)(1:50),fISI6),(ifnulI(14),nISI6),
     . (fidatI(15)(1:16),fIOS7),(ifnulI(15),nIOS7),
     . (fidatI(16)(1:50),fISI7),(ifnulI(16),nISI7),
     . (fidatI(17)(1:16),fIOS8),(ifnulI(17),nIOS8),
     . (fidatI(18)(1:50),fISI8),(ifnulI(18),nISI8),
     . (fidatI(19)(1:16),fIOS9),(ifnulI(19),nIOS9),
     . (fidatI(20)(1:50),fISI9),(ifnulI(20),nISI9),
     . (fidatI(21)(1:16),fIOS10),(ifnulI(21),nIOS10),
     . (fidatI(22)(1:50),fISI10),(ifnulI(22),nISI10),
     . (fidatI(23)(1:16),fIOS11),(ifnulI(23),nIOS11),
     . (fidatI(24)(1:50),fISI11),(ifnulI(24),nISI11),
     . (fidatI(25)(1:16),fIOS12),(ifnulI(25),nIOS12),
     . (fidatI(26)(1:50),fISI12),(ifnulI(26),nISI12),
     . (fidatI(27)(1:16),fIOS13),(ifnulI(27),nIOS13),
     . (fidatI(28)(1:50),fISI13),(ifnulI(28),nISI13),
     . (fidatI(29)(1:16),fIOS14),(ifnulI(29),nIOS14),
     . (fidatI(30)(1:50),fISI14),(ifnulI(30),nISI14),
     . (fidatI(31)(1:16),fIOS15),(ifnulI(31),nIOS15),
     . (fidatI(32)(1:50),fISI15),(ifnulI(32),nISI15),
     . (mxnocI(1),lISTI),(ifmodI(1),mISTI),
     . (mxnocI(2),lICSN),(ifmodI(2),mICSN),
     . (mxnocI(3),lIOS1),(ifmodI(3),mIOS1),
     . (mxnocI(4),lISI1),(ifmodI(4),mISI1),
     . (mxnocI(5),lIOS2),(ifmodI(5),mIOS2),
     . (mxnocI(6),lISI2),(ifmodI(6),mISI2),
     . (mxnocI(7),lIOS3),(ifmodI(7),mIOS3),
     . (mxnocI(8),lISI3),(ifmodI(8),mISI3),
     . (mxnocI(9),lIOS4),(ifmodI(9),mIOS4),
     . (mxnocI(10),lISI4),(ifmodI(10),mISI4),
     . (mxnocI(11),lIOS5),(ifmodI(11),mIOS5),
     . (mxnocI(12),lISI5),(ifmodI(12),mISI5),
     . (mxnocI(13),lIOS6),(ifmodI(13),mIOS6),
     . (mxnocI(14),lISI6),(ifmodI(14),mISI6),
     . (mxnocI(15),lIOS7),(ifmodI(15),mIOS7),
```

```
     . (mxnocI(16),lISI7),(ifmodI(16),mISI7),
     . (mxnocI(17),lIOS8),(ifmodI(17),mIOS8),
     . (mxnocI(18),lISI8),(ifmodI(18),mISI8),
     . (mxnocI(19),lIOS9),(ifmodI(19),mIOS9),
     . (mxnocI(20),lISI9),(ifmodI(20),mISI9),
     . (mxnocI(21),lIOS10),(ifmodI(21),mIOS10),
     . (mxnocI(22),lISI10),(ifmodI(22),mISI10),
     . (mxnocI(23),lIOS11),(ifmodI(23),mIOS11),
     . (mxnocI(24),lISI11),(ifmodI(24),mISI11),
     . (mxnocI(25),lIOS12),(ifmodI(25),mIOS12),
     . (mxnocI(26),lISI12),(ifmodI(26),mISI12),
     . (mxnocI(27),lIOS13),(ifmodI(27),mIOS13),
     . (mxnocI(28),lISI13),(ifmodI(28),mISI13),
     . (mxnocI(29),lIOS14),(ifmodI(29),mIOS14),
     . (mxnocI(30),lISI14),(ifmodI(30),mISI14),
     . (mxnocI(31),lIOS15),(ifmodI(31),mIOS15),
     . (mxnocI(32),lISI15),(ifmodI(32),mISI15)
C
C    Database table and attribute names for screen fields.
C
      data atrbtI/
     . 'SPECIAL_INSTRUCTIONS','SAMPLE_TRACKING_ID',
     . 'SAMPLE_INFORMATION','CUSTOMER_SAMPLE_NAME',
     . 'SPECIAL_INSTRUCTIONS','ORIGIN_OF_SPECIAL_INSTRUCTION',
     . 'SPECIAL_INSTRUCTIONS','SPECIAL_INSTRUCTION',
     . 'SPECIAL_INSTRUCTIONS','SPECIAL_INSTRUCTION_LINE'/
C
C    Program variable names (value and null indicator) for screen fields.
C
      data varblI/':fISTI:nISTI',':fICSN:nICSN',':fIOS1t:nIOS1',
     . ':fISI1t:nISI1',':siline:ifnsil'/
C
C    Possessor screen's field descriptors and field sizes.
C
      character*80 fieldT(23),fidatT(23)
      character*30 atrbtT(2,9)
      character*14 varblT(9)
      character*1 fitypT(23)
      integer*4 fdlenT(23),fdrowT(23),fdcolT(23),ficolT(23),mxnocT(23),
     . fdrndT(3,23),firndT(3,23), !Field renditions (add/update/search).
     . numfdT/23/   !The number of fields.
      integer*2 ifnulT(23),   !Null indicator (-1 if null, 0 otherwise).
     . ifrqdT(23)/23*0/,   !Points to if-required dependent field.
     . iditoT(23)/23*0/   !Points to allowed field for dittoing.
      logical ifmodT(23)   !Has a field been modified in an Update?
      common /dbdatT/ fieldT,fidatT,fdlenT,fdrowT,fdcolT,ficolT,mxnocT,
     . fdrndT,firndT,numfdT,fitypT,ifnulT,atrbtT,varblT,ifrqdT,ifmodT,
     . iditoT
C
C      Length-specified character fields for equivalences (f for field).
C
      character fTSTI*12,fTCSN*60,fTPSN1*3,fTPN1*25,fTPPN1*12,fTPA1*60,
     . fTPPR1*50,fTDAB1*6,fTDRB1*6,fTPSN2*3,fTPN2*25,fTPPN2*12,fTPA2*60,
     . fTPPR2*50,fTDAB2*6,fTDRB2*6,fTPSN3*3,fTPN3*25,fTPPN3*12,fTPA3*60,
     . fTPPR3*50,fTDAB3*6,fTDRB3*6,
     . fTPN1t*25,fTPPN1t*12,fTPA1t*60,fTPPR1t*50   !Temporary fields.
C
C       Integer field storage (i for integer).
C
      integer*4 iTPSN1,iTDAB1,iTDRB1
```

```
C
C       Nonarrayed maximum field lenghts (1 for length).
C
      integer*4 lTSTI,lTCSN,lTPSN1,lTPN1,lTPPN1,lTPA1,lTPPR1,lTDAB1,
     . lTDRB1,lTPSN2,lTPN2,lTPPN2,lTPA2,lTPPR2,lTDAB2,lTDRB2,lTPSN3,
     . lTPN3,lTPPN3,lTPA3,lTPPR3,lTDAB3,lTDRB3
C
C       Nonarrayed null indicators for data transfers (n for null).
C
      integer*2 nTSTI,nTCSN,nTPSN1,nTPN1,nTPPN1,nTPA1,nTPPR1,nTDAB1,
     . nTDRB1,nTPSN2,nTPN2,nTPPN2,nTPA2,nTPPR2,nTDAB2,nTDRB2,nTPSN3,
     . nTPN3,nTPPN3,nTPA3,nTPPR3,nTDAB3,nTDRB3
C
C       Nonarrayed if-modified indicators (m for modified).
C
      integer*2 mTSTI,mTCSN,mTPSN1,mTPN1,mTPPN1,mTPA1,mTPPR1,mTDAB1,
     . mTDRB1,mTPSN2,mTPN2,mTPPN2,mTPA2,mTPPR2,mTDAB2,mTDRB2,mTPSN3,
     . mTPN3,mTPPN3,mTPA3,mTPPR3,mTDAB3,mTDRB3
C
C       Equivalence statements for data transfers.
C
      equivalence (fidatT(1)(1:12),fTSTI),(ifnulT(1),nTSTI),
     . (fidatT(2)(1:60),fTCSN),(ifnulT(2),nTCSN),
     . (fidatT(3)(1:3),fTPSN1),(ifnulT(3),nTPSN1),
     . (fidatT(4)(1:25),fTPN1),(ifnulT(4),nTPN1),
     . (fidatT(5)(1:12),fTPPN1),(ifnulT(5),nTPPN1),
     . (fidatT(6)(1:60),fTPA1),(ifnulT(6),nTPA1),
     . (fidatT(7)(1:50),fTPPR1),(ifnulT(7),nTPPR1),
     . (fidatT(8)(1:6),fTDAB1),(ifnulT(8),nTDAB1),
     . (fidatT(9)(1:6),fTDRB1),(ifnulT(9),nTDRB1),
     . (fidatT(10)(1:3),fTPSN2),(ifnulT(10),nTPSN2),
     . (fidatT(11)(1:25),fTPN2),(ifnulT(11),nTPN2),
     . (fidatT(12)(1:12),fTPPN2),(ifnulT(12),nTPPN2),
     . (fidatT(13)(1:60),fTPA2),(ifnulT(13),nTPA2),
     . (fidatT(14)(1:50),fTPPR2),(ifnulT(14),nTPPR2),
     . (fidatT(15)(1:6),fTDAB2),(ifnulT(15),nTDAB2),
     . (fidatT(16)(1:6),fTDRB2),(ifnulT(16),nTDRB2),
     . (fidatT(17)(1:3),fTPSN3),(ifnulT(17),nTPSN3),
     . (fidatT(18)(1:25),fTPN3),(ifnulT(18),nTPN3),
     . (fidatT(19)(1:12),fTPPN3),(ifnulT(19),nTPPN3),
     . (fidatT(20)(1:60),fTPA3),(ifnulT(20),nTPA3),
     . (fidatT(21)(1:50),fTPPR3),(ifnulT(21),nTPPR3),
     . (fidatT(22)(1:6),fTDAB3),(ifnulT(22),nTDAB3),
     . (fidatT(23)(1:6),fTDRB3),(ifnulT(23),nTDRB3),
     . (mxnocT(1),lTSTI),(ifmodT(1),mTSTI),
     . (mxnocT(2),lTCSN),(ifmodT(2),mTCSN),
     . (mxnocT(3),lTPSN1),(ifmodT(3),mTPSN1),
     . (mxnocT(4),lTPN1),(ifmodT(4),mTPN1),
     . (mxnocT(5),lTPPN1),(ifmodT(5),mTPPN1),
     . (mxnocT(6),lTPA1),(ifmodT(6),mTPA1),
     . (mxnocT(7),lTPPR1),(ifmodT(7),mTPPR1),
     . (mxnocT(8),lTDAB1),(ifmodT(8),mTDAB1),
     . (mxnocT(9),lTDRB1),(ifmodT(9),mTDRB1),
     . (mxnocT(10),lTPSN2),(ifmodT(10),mTPSN2),
     . (mxnocT(11),lTPN2),(ifmodT(11),mTPN2),
     . (mxnocT(12),lTPPN2),(ifmodT(12),mTPPN2),
     . (mxnocT(13),lTPA2),(ifmodT(13),mTPA2),
     . (mxnocT(14),lTPPR2),(ifmodT(14),mTPPR2),
     . (mxnocT(15),lTDAB2),(ifmodT(15),mTDAB2),
     . (mxnocT(16),lTDRB2),(ifmodT(16),mTDRB2),
```

```
     . (mxnocT(17),lTPSN3),(ifmodT(17),mTPSN3),
     . (mxnocT(18),lTPN3),(ifmodT(18),mTPN3),
     . (mxnocT(19),lTPPN3),(ifmodT(19),mTPPN3),
     . (mxnocT(20),lTPA3),(ifmodT(20),mTPA3),
     . (mxnocT(21),lTPPR3),(ifmodT(21),mTPPR3),
     . (mxnocT(22),lTDAB3),(ifmodT(22),mTDAB3),
     . (mxnocT(23),lTDRB3),(ifmodT(23),mTDRB3)
C
C    Database table and attribute names for screen fields.
C
      data atrbtT/
     . 'TRACKING','SAMPLE_TRACKING_ID',
     . 'SAMPLE_INFORMATION','CUSTOMER_SAMPLE_NAME',
     . 'TRACKING','POSSESSOR_SEQUENCE_NUMBER',
     . 'TRACKING','POSSESSOR_NAME',
     . 'TRACKING','POSSESSOR_PHONE_NUMBER',
     . 'TRACKING','POSSESSOR_ADDRESS',
     . 'TRACKING','POSSESSOR_POSSESSION_REASON',
     . 'TRACKING','DATE_ACCEPTED_BY_POSSESSOR',
     . 'TRACKING','DATE_RELINQUISHED_BY_POSSESSOR'/
C
C   Program variable names (value and null indicator) for screen fields.
C
      data varblT/':fTSTI:nTSTI',':fTCSN:nTCSN',':iTPSN1:nTPSN1',
     . ':fTPN1:nTPN1',':fTPPN1:nTPPN1',':fTPA1:nTPA1',':fTPPR1:nTPPR1',
     . ':iTDAB1:nTDAB1',':iTDRB1:nTDRB1'/
C
C    Possessor name list field descriptors and sizes.
C
      character*80 fieldN(4,15)
      integer*4 fdlenN(4,15),fdrowN(15),fdcolN(15),ficolN(15),
     . numfdN   !The number of fields.
      common /lsdatN/ fieldN,fdlenN,fdrowN,fdcolN,ficolN,numfdN
C
C For the Main screen, . . .
C
C  Set defaults for display field renditions now so they can be modified
C   with each field as necessary.  Set all input fields to blanks.
C
      do i=1,numfdM
         fidatM(i)=blanks
         do j=1,3   !1 for adding, 2 for updating, and 3 for searching.
            fdrndM(j,i)=foptnl
            enddo
         enddo
C
      fieldM(1)='Tracking ID:'
      fdlenM(1)=12
      fdrowM(1)=2
      fdcolM(1)=1
      ficolM(1)=15
      mxnocM(1)=12
      fitypM(1)='1'
C
      fieldM(2)='Date Entered:'
      fdlenM(2)=13
      fdrowM(2)=2
      fdcolM(2)=31
      ficolM(2)=46
      mxnocM(2)=6
```

```
      fitypM(2)='d'
C
      do i=1,2
         do j=1,2    !Fixed when adding and updating.
            fdrndM(j,i)=ffixed
            enddo
         enddo
C
      fieldM(3)='Completed:'
      fdlenM(3)=10
      fdrowM(3)=2
      fdcolM(3)=56
      ficolM(3)=68
      mxnocM(3)=6
      fitypM(3)='d'
C
      fieldM(4)='Customer''s Sample ID:'
      fdlenM(4)=22
      fdrowM(4)=3
      fdcolM(4)=4
      ficolM(4)=27
      mxnocM(4)=12
      fitypM(4)='a'
C
      fieldM(5)='Desired Completion Date:'
      fdlenM(5)=24
      fdrowM(5)=3
      fdcolM(5)=45
      ficolM(5)=71
      mxnocM(5)=6
      fitypM(5)='d'
      do j=1,2    !Required when adding and updating.
         fdrndM(j,5)=freqd
         enddo
C
      fieldM(6)='Project:'
      fdlenM(6)=8
      fdrowM(6)=4
      fdcolM(6)=4
      ficolM(6)=14
      mxnocM(6)=40
      fitypM(6)='a'
C
      fieldM(7)='Charge #:'
      fdlenM(7)=9
      fdrowM(7)=4
      fdcolM(7)=58
      ficolM(7)=69
      mxnocM(7)=9
      fitypM(7)='l'
C
      fieldM(8)='Sample Name:'
      fdlenM(8)=12
      fdrowM(8)=5
      fdcolM(8)=4
      ficolM(8)=18
      mxnocM(8)=60
      fitypM(8)='a'
      iditoM(8)=4
C
```

```
      fieldM(9)='Sample Type:'
      fdlenM(9)=12
      fdrowM(9)=6
      fdcolM(9)=4
      ficolM(9)=18
      mxnocM(9)=10
      fitypM(9)='a'
C
      fieldM(10)='Sample Size:'
      fdlenM(10)=12
      fdrowM(10)=6
      fdcolM(10)=32
      ficolM(10)=46
      mxnocM(10)=5
      fitypM(10)='n'
C
      fieldM(11)='Size Units:'
      fdlenM(11)=11
      fdrowM(11)=6
      fdcolM(11)=55
      ficolM(11)=68
      mxnocM(11)=5
      fitypM(11)='a'
C
      fieldM(12)='Collection Date:'
      fdlenM(12)=16
      fdrowM(12)=7
      fdcolM(12)=4
      ficolM(12)=22
      mxnocM(12)=6
      fitypM(12)='d'
C
      do i=7,12
         do j=1,2   !Required when adding and updating.
            fdrndM(j,i)=freqd
            enddo
         enddo
C
      fieldM(13)='Time:'
      fdlenM(13)=5
      fdrowM(13)=7
      fdcolM(13)=32
      ficolM(13)=39
      mxnocM(13)=5
      fitypM(13)='t'
C
      fieldM(14)='Special Instructions?'
      fdlenM(14)=21
      fdrowM(14)=7
      fdcolM(14)=48
      ficolM(14)=71
      mxnocM(14)=1
      fitypM(14)='?'
C
      fieldM(15)='HP Surveyed?'
      fdlenM(15)=12
      fdrowM(15)=8
      fdcolM(15)=4
      ficolM(15)=18
      mxnocM(15)=1
```

```
      fitypM(15)='?'
      ifrqdM(15)=16
C
      fieldM(16)='If Yes, Sample Activity (mrem/h):'
      fdlenM(16)=33
      fdrowM(16)=8
      fdcolM(16)=23
      ficolM(16)=58
      mxnocM(16)=6
      fitypM(16)='n'
      ifrqdM(16)=15
C
      fieldM(17)='Submitter:'
      fdlenM(17)=10
      fdrowM(17)=9
      fdcolM(17)=1
      ficolM(17)=21
      mxnocM(17)=25
      fitypM(17)='a'
C
      lincnt=7
      do i=18,30,3
      lincnt=lincnt+2
         fieldM(i)='Phone Number:'
         fdlenM(i)=13
         fdrowM(i)=lincnt
         fdcolM(i)=50
         ficolM(i)=65
         mxnocM(i)=12
         fitypM(i)='a'
         ifrqdM(i)=i-1
         if(i.ne.18)iditoM(i)=i-3
C
         fieldM(i+1)='Address:'
         fdlenM(i+1)=8
         fdrowM(i+1)=lincnt+1
         fdcolM(i+1)=4
         ficolM(i+1)=14
         mxnocM(i+1)=60
         fitypM(i+1)='a'
         ifrqdM(i+1)=i-1
         if(i.ne.18)iditoM(i+1)=i-2
         enddo
C
      do i=17,19
         do j=1,2    !Required when adding and updating.
            fdrndM(j,i)=freqd
            enddo
         enddo
C
      fieldM(20)='Technical Contact:'
      fdlenM(20)=18
      fdrowM(20)=11
      fdcolM(20)=1
      ficolM(20)=21
      mxnocM(20)=25
      fitypM(20)='a'
      iditoM(20)=17
C
      fieldM(23)='Send Results To:'
```

```
      fdlenM(23)=16
      fdrowM(23)=13
      fdcolM(23)=1
      ficolM(23)=21
      mxnocM(23)=25
      fitypM(23)='a'
      iditoM(23)=20
C
      fieldM(26)='Sample Pickup By:'
      fdlenM(26)=17
      fdrowM(26)=15
      fdcolM(26)=1
      ficolM(26)=21
      mxnocM(26)=25
      fitypM(26)='a'
      iditoM(26)=23
C
      do i=23,28
         do j=1,2    !Required when adding and updating.
            fdrndM(j,i)=freqd
            enddo
         enddo
C
      fieldM(29)='Current Possessor:'
      fdlenM(29)=18
      fdrowM(29)=17
      fdcolM(29)=1
      ficolM(29)=21
      mxnocM(29)=25
      fitypM(29)='a'
C
      do i=29,31
         fdrndM(1,i)=ffixed    !Fixed when adding and updating.
         fdrndM(2,i)=ffixed
         enddo
C
      fieldM(32)='Chain of Custody Number:'
      fdlenM(32)=24
      fdrowM(32)=19
      fdcolM(32)=1
      ficolM(32)=27
      mxnocM(32)=10
      fitypM(32)='a'
C
      fieldM(33)='Analyses Needed:  Gross Alpha-Beta?'
      fdlenM(33)=35
      fdrowM(33)=20
      fdcolM(33)=1
      ficolM(33)=38
      mxnocM(33)=1
      fitypM(33)='?'
C
      fieldM(34)='Alpha?'
      fdlenM(34)=6
      fdrowM(34)=20
      fdcolM(34)=43
      ficolM(34)=51
      mxnocM(34)=1
      fitypM(34)='?'
C
```

```
      fieldM(35)='Beta?'
      fdlenM(35)=5
      fdrowM(35)=20
      fdcolM(35)=56
      ficolM(35)=63
      mxnocM(35)=1
      fitypM(35)='?'
C
      fieldM(36)='Gamma?'
      fdlenM(36)=6
      fdrowM(36)=20
      fdcolM(36)=68
      ficolM(36)=76
      mxnocM(36)=1
      fitypM(36)='?'
C
C     Set all input field renditions to the bit-wise .or. of the display
C      field rendition with finput.  Note that all flags are required
C      when adding and updating.
C
      do i=1,numfdM
         do j=1,3
            if((j.ne.3).and.(fitypM(i).eq.'?'))fdrndM(j,i)=freqd
            firndM(j,i)=finput.or.fdrndM(j,i)
            enddo
         enddo
C
C For the Alpha screen, . . .
C
C   Set defaults for display field renditions now so they can be modified
C    with each field as necessary.  Set all input fields to blanks.
C
      do i=1,numfdA
         fidatA(i)=blanks
         do j=1,3    !1 for adding, 2 for updating, and 3 for searching.
            fdrndA(j,i)=foptnl
            enddo
         enddo
C
      fieldA(1)='Tracking ID:'
      fdlenA(1)=12
      fdrowA(1)=3
      fdcolA(1)=1
      ficolA(1)=15
      mxnocA(1)=12
      fitypA(1)='1'
C
      fieldA(2)='Sample Name:'
      fdlenA(2)=12
      fdrowA(2)=4
      fdcolA(2)=4
      ficolA(2)=18
      mxnocA(2)=60
      fitypA(2)='a'
C
      do i=1,2
         do j=1,2    !Fixed when adding and updating.
            fdrndA(j,i)=ffixed
            enddo
         enddo
```

```
C
      fieldA(3)='Uranium isotopes?'
      fdlenA(3)=17
      fdrowA(3)=6
      fdcolA(3)=1
      ficolA(3)=20
      mxnocA(3)=1
      fitypA(3)='?'
C
      lincnt=4
      do i=4,34,5
         lincnt=lincnt+2
         fieldA(i)='Needed by:'
         fdlenA(i)=10
         fdrowA(i)=lincnt
         fdcolA(i)=36
         ficolA(i)=48
         mxnocA(i)=6
         fitypA(i)='d'
         ifrqdA(i)=i-1
         if(i.ne.4)iditoA(i)=i-5
C
         fieldA(i+1)='Completed:'
         fdlenA(i+1)=10
         fdrowA(i+1)=lincnt
         fdcolA(i+1)=58
         ficolA(i+1)=70
         mxnocA(i+1)=6
         fitypA(i+1)='d'
         ifrqdA(i+1)=i+2
         if(i.ne.4)iditoA(i+1)=i-4
C
         fieldA(i+2)='Results report citation:'
         fdlenA(i+2)=24
         fdrowA(i+2)=lincnt+1
         fdcolA(i+2)=4
         ficolA(i+2)=30
         mxnocA(i+2)=20
         fitypA(i+2)='a'
         ifrqdA(i+2)=i+1
         if(i.ne.4)iditoA(i+2)=i-3
C
         fieldA(i+3)='Special Instructions?'
         fdlenA(i+3)=21
         fdrowA(i+3)=lincnt+1
         fdcolA(i+3)=54
         ficolA(i+3)=77
         mxnocA(i+3)=1
         fitypA(i+3)='?'
         ifrqdA(i+3)=i-1
         if(i.ne.4)iditoA(i+3)=i-2
         enddo
C
      fieldA(8)='Thorium isotopes?'
      fdlenA(8)=17
      fdrowA(8)=8
      fdcolA(8)=1
      ficolA(8)=20
      mxnocA(8)=1
      fitypA(8)='?'
```

```
C
      fieldA(13)='Plutonium isotopes?'
      fdlenA(13)=19
      fdrowA(13)=10
      fdcolA(13)=1
      ficolA(13)=22
      mxnocA(13)=1
      fitypA(13)='?'
C
      fieldA(18)='Am-241 separate from Pu-238?'
      fdlenA(18)=28
      fdrowA(18)=12
      fdcolA(18)=1
      ficolA(18)=31
      mxnocA(18)=1
      fitypA(18)='?'
C
      fieldA(23)='Am-241 combined with Pu-238?'
      fdlenA(23)=28
      fdrowA(23)=14
      fdcolA(23)=1
      ficolA(23)=31
      mxnocA(23)=1
      fitypA(23)='?'
C
      fieldA(28)='Total Spectrometric Alpha?'
      fdlenA(28)=26
      fdrowA(28)=16
      fdcolA(28)=1
      ficolA(28)=29
      mxnocA(28)=1
      fitypA(28)='?'
C
      fieldA(33)='Other?'
      fdlenA(33)=6
      fdrowA(33)=18
      fdcolA(33)=1
      ficolA(33)=9
      mxnocA(33)=1
      fitypA(33)='?'
C
C    Set all input field renditions to the bit-wise .or. of the display
C     field rendition with finput.  Note that some flags are required
C     when adding and updating.
C
      do i=1,numfdA
         do j=1,3
            if((j.ne.3).and.(fitypA(i).eq.'?').and.
     .        (fieldA(i).ne.'Special Instructions?'))fdrndA(j,i)=freqd
            firndA(j,i)=finput.or.fdrndA(j,i)
            enddo
         enddo
C
C For the Beta screen, . . .
C
C   Set defaults for display field renditions now so they can be modified
C    with each field as necessary.  Set all input fields to blanks.
C
      do i=1,numfdB
         fidatB(i)=blanks
```

```
         do j=1,3   !1 for adding, 2 for updating, and 3 for searching.
            fdrndB(j,i)=foptnl
            enddo
         enddo
C
      fieldB(1)='Tracking ID:'
      fdlenB(1)=12
      fdrowB(1)=2
      fdcolB(1)=1
      ficolB(1)=15
      mxnocB(1)=12
      fitypB(1)='l'
C
      fieldB(2)='Sample Name:'
      fdlenB(2)=12
      fdrowB(2)=3
      fdcolB(2)=4
      ficolB(2)=18
      mxnocB(2)=60
      fitypB(2)='a'
C
      do i=1,2
         do j=1,2   !Fixed when adding and updating.
            fdrndB(j,i)=ffixed
            enddo
         enddo
C
      fieldB(3)='Strontium-90?'
      fdlenB(3)=13
      fdrowB(3)=5
      fdcolB(3)=1
      ficolB(3)=16
      mxnocB(3)=1
      fitypB(3)='?'
C
      lincnt=2
      do i=4,28,8
         lincnt=lincnt+3
         fieldB(i)='Length of count:'
         fdlenB(i)=16
         fdrowB(i)=lincnt
         fdcolB(i)=29
         ficolB(i)=47
         mxnocB(i)=4
         fitypB(i)='n'
         ifrqdB(i)=i+1
         if(i.ne.4)iditoB(i)=i-8
C
         fieldB(i+1)='min or hr?'
         fdlenB(i+1)=10
         fdrowB(i+1)=lincnt
         fdcolB(i+1)=55
         ficolB(i+1)=67
         mxnocB(i+1)=1
         fitypB(i+1)='u'
         ifrqdB(i+1)=i
         if(i.ne.4)iditoB(i+1)=i-7
C
         fieldB(i+2)='Results needed by:  date:'
         fdlenB(i+2)=25
```

```
            fdrowB(i+2)=lincnt+1
            fdcolB(i+2)=4
            ficolB(i+2)=31
            mxnocB(i+2)=6
            fitypB(i+2)='d'
            ifrqdB(i+2)=i-1
            if(i.ne.4)iditoB(i+2)=i-6
C
            fieldB(i+3)='time:'
            fdlenB(i+3)=5
            fdrowB(i+3)=lincnt+1
            fdcolB(i+3)=41
            ficolB(i+3)=48
            mxnocB(i+3)=5
            fitypB(i+3)='t'
            if(i.ne.4)iditoB(i+3)=i-5
C
            fieldB(i+4)='Completed:'
            fdlenB(i+4)=10
            fdrowB(i+4)=lincnt+1
            fdcolB(i+4)=57
            ficolB(i+4)=69
            mxnocB(i+4)=6
            fitypB(i+4)='d'
            ifrqdB(i+4)=i+5
            if(i.ne.4)iditoB(i+4)=i-4
C
            fieldB(i+5)='Results report citation:'
            fdlenB(i+5)=24
            fdrowB(i+5)=lincnt+2
            fdcolB(i+5)=4
            ficolB(i+5)=30
            mxnocB(i+5)=20
            fitypB(i+5)='a'
            ifrqdB(i+5)=i+4
            if(i.ne.4)iditoB(i+5)=i-3
C
            fieldB(i+6)='Special Instructions?'
            fdlenB(i+6)=21
            fdrowB(i+6)=lincnt+2
            fdcolB(i+6)=54
            ficolB(i+6)=77
            mxnocB(i+6)=1
            fitypB(i+6)='?'
            ifrqdB(i+6)=i-1
            if(i.ne.4)iditoB(i+6)=i-2
            enddo
C
          fieldB(11)='Strontium-89 and -90?'
          fdlenB(11)=21
          fdrowB(11)=8
          fdcolB(11)=1
          ficolB(11)=24
          mxnocB(11)=1
          fitypB(11)='?'
C
          fieldB(19)='Total Strontium?'
          fdlenB(19)=16
          fdrowB(19)=11
          fdcolB(19)=1
```

```
      ficolB(19)=19
      mxnocB(19)=1
      fitypB(19)='?'
C
      fieldB(27)='Tritium?'
      fdlenB(27)=8
      fdrowB(27)=14
      fdcolB(27)=1
      ficolB(27)=11
      mxnocB(27)=1
      fitypB(27)='?'
C
      fieldB(35)='Other?'
      fdlenB(35)=6
      fdrowB(35)=17
      fdcolB(35)=1
      ficolB(35)=9
      mxnocB(35)=1
      fitypB(35)='?'
C
      fieldB(36)='Length of count:'
      fdlenB(36)=16
      fdrowB(36)=17
      fdcolB(36)=29
      ficolB(36)=47
      mxnocB(36)=4
      fitypB(36)='n'
      iditoB(36)=28
C
      fieldB(37)='min or hr?'
      fdlenB(37)=10
      fdrowB(37)=17
      fdcolB(37)=55
      ficolB(37)=67
      mxnocB(37)=1
      fitypB(37)='u'
      ifrqdB(37)=36
      iditoB(37)=29
C
      fieldB(38)='Nickel-63?'
      fdlenB(38)=10
      fdrowB(38)=18
      fdcolB(38)=4
      ficolB(38)=16
      mxnocB(38)=1
      fitypB(38)='?'
      ifrqdB(38)=35
C
      fieldB(39)='Iron-55?'
      fdlenB(39)=8
      fdrowB(39)=18
      fdcolB(39)=21
      ficolB(39)=31
      mxnocB(39)=1
      fitypB(39)='?'
      ifrqdB(39)=35
C
      fieldB(40)='Sulfur-35?'
      fdlenB(40)=10
      fdrowB(40)=18
```

```
      fdcolB(40)=36
      ficolB(40)=48
      mxnocB(40)=1
      fitypB(40)='?'
      ifrqdB(40)=35
C
      fieldB(41)='Plutonium-241?'
      fdlenB(41)=14
      fdrowB(41)=18
      fdcolB(41)=53
      ficolB(41)=69
      mxnocB(41)=1
      fitypB(41)='?'
      ifrqdB(41)=35
C
      fieldB(42)='Results needed by:  date:'
      fdlenB(42)=25
      fdrowB(42)=19
      fdcolB(42)=4
      ficolB(42)=31
      mxnocB(42)=6
      fitypB(42)='d'
      ifrqdB(42)=35
      iditoB(42)=30
C
      fieldB(43)='time:'
      fdlenB(43)=5
      fdrowB(43)=19
      fdcolB(43)=41
      ficolB(43)=48
      mxnocB(43)=5
      fitypB(43)='t'
      iditoB(43)=31
C
      fieldB(44)='Completed:'
      fdlenB(44)=10
      fdrowB(44)=19
      fdcolB(44)=57
      ficolB(44)=69
      mxnocB(44)=6
      fitypB(44)='d'
      ifrqdB(44)=45
      iditoB(44)=32
C
      fieldB(45)='Results report citation:'
      fdlenB(45)=24
      fdrowB(45)=20
      fdcolB(45)=4
      ficolB(45)=30
      mxnocB(45)=20
      fitypB(45)='a'
      ifrqdB(44)=44
      iditoB(45)=33
C
      fieldB(46)='Special Instructions?'
      fdlenB(46)=21
      fdrowB(46)=20
      fdcolB(46)=54
      ficolB(46)=77
      mxnocB(46)=1
```

```
      fitypB(46)='?'
      ifrqdB(46)=35
      iditoB(46)=34
C
C     Set all input field renditions to the bit-wise .or. of the display
C      field rendition with finput.  Note that some flags are required
C      when adding and updating.
C
      do i=1,numfdB
         do j=1,3
            if((j.ne.3).and.(fitypB(i).eq.'?').and.(i.le.35).and.
     .          (fieldB(i).ne.'Special Instructions?'))fdrndB(j,i)=freqd
            firndB(j,i)=finput.or.fdrndB(j,i)
            enddo
         enddo
C
C For the Gamma screen, . . .
C
C   Set defaults for display field renditions now so they can be modified
C    with each field as necessary.  Set all input fields to blanks.
C
      do i=1,numf
         fidatG(i  =blanks
         do j=1,3    !1 for adding, 2 for updating, and 3 for searching.
            fdrndG(j,i)=foptnl
            enddo
         enddo
C
      fieldG(1)='Tracking ID:'
      fdlenG(1)=12
      fdrowG(1)=2
      fdcolG(1)=1
      ficolG(1)=15
      mxnocG(1)=12
      fitypG(1)='1'
C
      fieldG(2)='Sample Name:'
      fdlenG(2)=12
      fdrowG(2)=3
      fdcolG(2)=4
      ficolG(2)=18
      mxnocG(2)=60
      fitypG(2)='a'
C
      do i=1,2
         do j=1,2    !Fixed when adding and updating.
            fdrndG(j,i)=ffixed
            enddo
         enddo
C
      fieldG(3)='Screening/Shipping Count?'
      fdlenG(3)=25
      fdrowG(3)=6
      fdcolG(3)=1
      ficolG(3)=28
      mxnocG(3)=1
      fitypG(3)='?'
C
      lincnt=1
      do i=4,26,11
```

```
      lincnt=lincnt+5
      fieldG(i)='Length of count:'
      fdlenG(i)=16
      fdrowG(i)=lincnt
      fdcolG(i)=37
      ficolG(i)=55
      mxnocG(i)=4
      fitypG(i)='n'
      if(i.ne.4)iditoG(i)=i-11
C
      fieldG(i+1)='min or hr?'
      fdlenG(i+1)=10
      fdrowG(i+1)=lincnt
      fdcolG(i+1)=63
      ficolG(i+1)=75
      mxnocG(i+1)=1
      fitypG(i+1)='u'
      ifrqdG(i+1)=i
      if(i.ne.4)iditoG(i+1)=i-10
C
      fieldG(i+2)='Results needed by:  date:'
      fdlenG(i+2)=25
      fdrowG(i+2)=lincnt+1
      fdcolG(i+2)=4
      ficolG(i+2)=31
      mxnocG(i+2)=6
      fitypG(i+2)='d'
      ifrqdG(i+2)=i-1
      if(i.ne.4)iditoG(i+2)=i-9
C
      fieldG(i+3)='time:'
      fdlenG(i+3)=5
      fdrowG(i+3)=lincnt+1
      fdcolG(i+3)=41
      ficolG(i+3)=48
      mxnocG(i+3)=5
      fitypG(i+3)='t'
      if(i.ne.4)iditoG(i+3)=i-8
C
      fieldG(i+4)='Completed:'
      fdlenG(i+4)=10
      fdrowG(i+4)=lincnt+1
      fdcolG(i+4)=57
      ficolG(i+4)=69
      mxnocG(i+4)=6
      fitypG(i+4)='d'
      ifrqdG(i+4)=i+8
      if(i.ne.4)iditoG(i+4)=i-7
C
      fieldG(i+5)='Date counted:'
      fdlenG(i+5)=13
      fdrowG(i+5)=lincnt+2
      fdcolG(i+5)=4
      ficolG(i+5)=19
      mxnocG(i+5)=6
      fitypG(i+5)='d'
      ifrqdG(i+5)=i+4
C
      fieldG(i+6)='RML Spectral ID:'
      fdlenG(i+6)=16
```

```
            fdrowG(i+6)=lincnt+2
            fdcolG(i+6)=29
            ficolG(i+6)=47
            mxnocG(i+6)=14
            fitypG(i+6)='l'
            ifrqdG(i+6)=i+5
C
            fieldG(i+7)='Recount?'
            fdlenG(i+7)=8
            fdrowG(i+7)=lincnt+2
            fdcolG(i+7)=65
            ficolG(i+7)=75
            mxnocG(i+7)=1
            fitypG(i+7)='?'
            ifrqdG(i+7)=i+5
C
            fieldG(i+8)='Results report citation:'
            fdlenG(i+8)=24
            fdrowG(i+8)=lincnt+3
            fdcolG(i+8)=4
            ficolG(i+8)=30
            mxnocG(i+8)=20
            fitypG(i+8)='a'
            ifrqdG(i+8)=i+4
            if(i.ne.4)iditoG(i+8)=i-3
C
            fieldG(i+9)='Special Instructions?'
            fdlenG(i+9)=21
            fdrowG(i+9)=lincnt+3
            fdcolG(i+9)=54
            ficolG(i+9)=77
            mxnocG(i+9)=1
            fitypG(i+9)='?'
            ifrqdG(i+9)=i-1
            if(i.ne.4)iditoG(i+9)=i-2
            enddo
C
         fieldG(14)='Full Isotopic Gamma Analysis?'
         fdlenG(14)=29
         fdrowG(14)=11
         fdcolG(14)=1
         ficolG(14)=32
         mxnocG(14)=1
         fitypG(14)='?'
C
         fieldG(25)='Other Gamma Analysis?'
         fdlenG(25)=21
         fdrowG(25)=16
         fdcolG(25)=1
         ficolG(25)=24
         mxnocG(25)=1
         fitypG(25)='?'
C
C     Set all input field renditions to the bit-wise .or. of the display
C      field rendition with finput.  Note that some flags are required
C      when adding and updating.
C
         do i=1,numfdG
            do j=1,3
               if((j.ne.3).and.(fitypG(i).eq.'?').and.
```

```
     .       (fieldG(i).ne.'Recount?').and.
     .       (fieldG(i).ne.'Special Instructions?'))fdrndG(j,i)=freqd
            firndG(j,i)=finput.or.fdrndG(j,i)
            enddo
         enddo
C
C For the Gross Alpha-Beta screen, . . .
C
C   Set defaults for display field renditions now so they can be modified
C    with each field as necessary.  Set all input fields to blanks.
C
      do i=1,numfdC
         fidatC(i)=blanks
         do j=1,3   !1 for adding, 2 for updating, and 3 for searching.
            fdrndC(j,i)=foptnl
            enddo
         enddo
C
      fieldC(1)='Tracking ID:'
      fdlenC(1)=12
      fdrowC(1)=5
      fdcolC(1)=1
      ficolC(1)=15
      mxnocC(1)=12
      fitypC(1)='1'
C
      fieldC(2)='Sample Name:'
      fdlenC(2)=12
      fdrowC(2)=6
      fdcolC(2)=4
      ficolC(2)=18
      mxnocC(2)=60
      fitypC(2)='a'
C
      do i=1,2
         do j=1,2   !Fixed when adding and updating.
            fdrndC(j,i)=ffixed
            enddo
         enddo
C
      fieldC(3)='Gross Alpha-Beta for Air Filters?'
      fdlenC(3)=33
      fdrowC(3)=8
      fdcolC(3)=1
      ficolC(3)=36
      mxnocC(3)=1
      fitypC(3)='?'
C
      lincnt=4
      do i=4,12,8
         lincnt=lincnt+5
         fieldC(i)='Length of count:'
         fdlenC(i)=16
         fdrowC(i)=lincnt
         fdcolC(i)=4
         ficolC(i)=22
         mxnocC(i)=4
         fitypC(i)='n'
         if(i.ne.4)iditoC(i)=i-8
C
```

```
      fieldC(i+1)='min or hr?'
      fdlenC(i+1)=10
      fdrowC(i+1)=lincnt
      fdcolC(i+1)=30
      ficolC(i+1)=42
      mxnocC(i+1)=1
      fitypC(i+1)='u'
      ifrqdC(i+1)=i
      if(i.ne.4)iditoC(i+1)=i-7
C
      fieldC(i+2)='Results needed by:  date:'
      fdlenC(i+2)=25
      fdrowC(i+2)=lincnt+1
      fdcolC(i+2)=4
      ficolC(i+2)=31
      mxnocC(i+2)=6
      fitypC(i+2)='d'
      ifrqdC(i+2)=i-1
      if(i.ne.4)iditoC(i+2)=i-6
C
      fieldC(i+3)='time:'
      fdlenC(i+3)=5
      fdrowC(i+3)=lincnt+1
      fdcolC(i+3)=41
      ficolC(i+3)=48
      mxnocC(i+3)=5
      fitypC(i+3)='t'
      if(i.ne.4)iditoC(i+3)=i-5
C
      fieldC(i+4)='Completed:'
      fdlenC(i+4)=10
      fdrowC(i+4)=lincnt+1
      fdcolC(i+4)=57
      ficolC(i+4)=69
      mxnocC(i+4)=6
      fitypC(i+4)='d'
      ifrqdC(i+4)=i+5
      if(i.ne.4)iditoC(i+4)=i-4
C
      fieldC(i+5)='Results report citation:'
      fdlenC(i+5)=24
      fdrowC(i+5)=lincnt+2
      fdcolC(i+5)=4
      ficolC(i+5)=30
      mxnocC(i+5)=20
      fitypC(i+5)='a'
      ifrqdC(i+5)=i+4
      if(i.ne.4)iditoC(i+5)=i-3
C
      fieldC(i+6)='Special Instructions?'
      fdlenC(i+6)=21
      fdrowC(i+6)=lincnt+2
      fdcolC(i+6)=54
      ficolC(i+6)=77
      mxnocC(i+6)=1
      fitypC(i+6)='?'
      ifrqdC(i+6)=i-1
      if(i.ne.4)iditoC(i+6)=i-2
      enddo
C
```

```
      fieldC(11)='Gross Alpha-Beta for Other Samples?'
      fdlenC(11)=35
      fdrowC(11)=13
      fdcolC(11)=1
      ficolC(11)=38
      mxnocC(11)=1
      fitypC(11)='?'
C
C     Set all input field renditions to the bit-wise .or. of the display
C      field rendition with finput.  Note that some flags are required
C      when adding and updating.
C
      do i=1,numfdC
         do j=1,3
            if((j.ne.3).and.(fitypC(i).eq.'?').and.
     .          (fieldC(i).ne.'Special Instructions?'))fdrndC(j,i)=freqd
            firndC(j,i)=finput.or.fdrndC(j,i)
            enddo
         enddo
C
C For the Special Instruction screen, . . .
C
C   Set defaults for display field renditions now so they can be modified
C    with each field as necessary.  Set all input fields to blanks.
C
      do i=1,numfdI
         fidatI(i)=blanks
         do j=1,3    !1 for adding, 2 for updating, and 3 for searching.
            fdrndI(j,i)=foptnl
            enddo
         enddo
C
      fieldI(1)='Tracking ID:'
      fdlenI(1)=12
      fdrowI(1)=2
      fdcolI(1)=1
      ficolI(1)=15
      mxnocI(1)=12
      fitypI(1)='l'
C
      fieldI(2)='Sample Name:'
      fdlenI(2)=12
      fdrowI(2)=3
      fdcolI(2)=4
      ficolI(2)=18
      mxnocI(2)=60
      fitypI(2)='a'
C
      do i=1,2
         do j=1,2    !Fixed when adding and updating.
            fdrndI(j,i)=ffixed
            enddo
         enddo
C
      do i=3,31,2
         fieldI(i)='From'
         fdlenI(i)=4
         fdrowI(i)=3+(i+1)/2
         fdcolI(i)=4
         ficolI(i)=9
```

```
            mxnocI(i)=16
            fitypI(i)='a'
            fdrndI(1,i)=ffixed    !Fixed when adding.
            fdrndI(2,i)=ffixed    !Fixed when updating.
C
            fieldI(i+1)=':'
            fdlenI(i+1)=1
            fdrowI(i+1)=3+(i+1)/2
            fdcolI(i+1)=25
            ficolI(i+1)=28
            mxnocI(i+1)=50
            fitypI(i+1)='a'
            enddo
C
C     Set all input field renditions to the bit-wise .or. of the display
C      field rendition with finput.
C
         do i=1,numfdI
            do j=1,3
               firndI(j,i)=finput.or.fdrndI(j,i)
               enddo
            enddo
C
C For the Possessor screen, . . .
C
C   Set defaults for display field renditions now so they can be modified
C    with each field as necessary.  Set all input fields to blanks.
C
         do i=1,numfdT
            fidatT(i)=blanks
            do j=1,3    !1 for adding, 2 for updating, and 3 for searching.
               fdrndT(j,i)=foptnl
               enddo
            enddo
C
         fieldT(1)='Tracking ID:'
         fdlenT(1)=12
         fdrowT(1)=3
         fdcolT(1)=1
         ficolT(1)=15
         mxnocT(1)=12
         fitypT(1)='1'
C
         fieldT(2)='Sample Name:'
         fdlenT(2)=12
         fdrowT(2)=4
         fdcolT(2)=4
         ficolT(2)=18
         mxnocT(2)=60
         fitypT(2)='a'
C
         do i=1,2
            do j=1,2    !Fixed when adding and updating.
               fdrndT(j,i)=ffixed
               enddo
            enddo
C
         lincnt=1
         do i=3,17,7
            lincnt=lincnt+5
```

```
      fieldT(i)='Possessor #'
      fdlenT(i)=11
      fdrowT(i)=lincnt
      fdcolT(i)=1
      ficolT(i)=13
      mxnocT(i)=3
      fitypT(i)='n'
      fdrndT(1,i)=ffixed   !Fixed when adding and
      fdrndT(2,i)=ffixed   !updating.
C
      fieldT(i+1)=':'
      fdlenT(i+1)=1
      fdrowT(i+1)=lincnt
      fdcolT(i+1)=16
      ficolT(i+1)=22
      mxnocT(i+1)=25
      fitypT(i+1)='a'
      if(i.eq.3)then
            fdrndT(1,(i+1))=ffixed   !Fixed when adding and
            fdrndT(2,(i+1))=ffixed   !updating.
         else
            ifrqdT(i+1)=i-1
         endif
C
      fieldT(i+2)='Phone Number:'
      fdlenT(i+2)=13
      fdrowT(i+2)=lincnt
      fdcolT(i+2)=51
      ficolT(i+2)=66
      mxnocT(i+2)=12
      fitypT(i+2)='a'
      if(i.eq.3)then
            fdrndT(1,(i+2))=ffixed   !Fixed when adding and
            fdrndT(2,(i+2))=ffixed   !updating.
         else
            ifrqdT(i+2)=i+1
         endif
C
      fieldT(i+3)='Address:'
      fdlenT(i+3)=8
      fdrowT(i+3)=lincnt+1
      fdcolT(i+3)=4
      ficolT(i+3)=14
      mxnocT(i+3)=60
      fitypT(i+3)='a'
      if(i.eq.3)then
            fdrndT(1,(i+3))=ffixed   !Fixed when adding and
            fdrndT(2,(i+3))=ffixed   !updating.
         else
            ifrqdT(i+3)=i+1
         endif
C
      fieldT(i+4)='Reason for possession:'
      fdlenT(i+4)=22
      fdrowT(i+4)=lincnt+2
      fdcolT(i+4)=4
      ficolT(i+4)=28
      mxnocT(i+4)=50
      fitypT(i+4)='a'
      if(i.eq.3)then
```

```
               fdrndT(1,(i+4))=ffixed    !Fixed when adding and
               fdrndT(2,(i+4))=ffixed    !updating.
            else
               ifrqdT(i+4)=i+1
            endif
C
         fieldT(i+5)='Date sample accepted:'
         fdlenT(i+5)=21
         fdrowT(i+5)=lincnt+3
         fdcolT(i+5)=4
         ficolT(i+5)=27
         mxnocT(i+5)=6
         fitypT(i+5)='d'
         if(i.eq.3)then
               fdrndT(1,(i+5))=ffixed    !Fixed when adding and
               fdrndT(2,(i+5))=ffixed    !updating.
            else
               ifrqdT(i+5)=i+1
            endif
         if(i.ne.3)iditoT(i+5)=i-1    !To last date sample relinquished.
C
         fieldT(i+6)='Date sample relinquished:'
         fdlenT(i+6)=25
         fdrowT(i+6)=lincnt+3
         fdcolT(i+6)=37
         ficolT(i+6)=64
         mxnocT(i+6)=6
         fitypT(i+6)='d'
         enddo
C
C     Set all input field renditions to the bit-wise .or. of the display
C      field rendition with finput.
C
       do i=1,numfdT
          do j=1,3
             firndT(j,i)=finput.or.fdrndT(j,i)
             enddo
          enddo
C
C     Possessor name list data (set up to ease adding and removing
C      names - note that the current limit is fifteen names).
C
       numfdN=1
       fieldN(1,numfdN)='J. L. DOHERTY (JODIE)'
       fdlenN(1,numfdN)=21
       fieldN(2,numfdN)='6-6573'
       fdlenN(2,numfdN)=6
       fieldN(3,numfdN)='RML, MS 7111; TRA-604, ROOM 123'
       fdlenN(3,numfdN)=31
       fieldN(4,numfdN)='GAMMA SPECTRUM ACQUISITION'
       fdlenN(4,numfdN)=26
C
       numfdN=numfdN+1
       fieldN(1,numfdN)='T. J. HANEY (TOM)'
       fdlenN(1,numfdN)=17
       fieldN(2,numfdN)='6-4158'
       fdlenN(2,numfdN)=6
       fieldN(3,numfdN)='RML, MS 7111; TRA-604, ROOM 126'
       fdlenN(3,numfdN)=31
       fieldN(4,numfdN)='SAMPLE ROUTING COORDINATION'
```

```
      fdlenN(4,numfdN)=27
C
      numfdN=numfdN+1
      fieldN(1,numfdN)='R. K. MURRAY (RON)'
      fdlenN(1,numfdN)=18
      fieldN(2,numfdN)='6-4182'
      fdlenN(2,numfdN)=6
      fieldN(3,numfdN)='RML, MS 7111; TRA-604, ROOM 123'
      fdlenN(3,numfdN)=31
      fieldN(4,numfdN)='GAMMA SPECTRUM ACQUISITION'
      fdlenN(4,numfdN)=26
C
      numfdN=numfdN+1
      fieldN(1,numfdN)='C. L. ROWSELL (CAL)'
      fdlenN(1,numfdN)=19
      fieldN(2,numfdN)='6-4182'
      fdlenN(2,numfdN)=6
      fieldN(3,numfdN)='RML, MS 7111; TRA-604, ROOM 123'
      fdlenN(3,numfdN)=31
      fieldN(4,numfdN)='GROSS ALPHA-BETA COUNTING'
      fdlenN(4,numfdN)=25
C
      numfdN=numfdN+1
      fieldN(1,numfdN)='C. W. SILL (CLAUDE)'
      fdlenN(1,numfdN)=19
      fieldN(2,numfdN)='6-0605'
      fdlenN(2,numfdN)=6
      fieldN(3,numfdN)='RADIOCHEMISTRY, MS 7111; TRA-604, ROOM 111'
      fdlenN(3,numfdN)=42
      fieldN(4,numfdN)='ALPHA ANALYSES'
      fdlenN(4,numfdN)=14
C
      numfdN=numfdN+1
      fieldN(1,numfdN)='D. S. SILL (DAVE)'
      fdlenN(1,numfdN)=17
      fieldN(2,numfdN)='6-8031'
      fdlenN(2,numfdN)=6
      fieldN(3,numfdN)='RADIOCHEMISTRY, MS 7111; TRA-604, ROOM 110'
      fdlenN(3,numfdN)=42
      fieldN(4,numfdN)='ALPHA ANALYSES'
      fdlenN(4,numfdN)=14
C
      numfdN=numfdN+1
      fieldN(1,numfdN)='L. D. SMITH (LARRY)'
      fdlenN(1,numfdN)=19
      fieldN(2,numfdN)='6-4405'
      fdlenN(2,numfdN)=6
      fieldN(3,numfdN)='RML, MS 7111; TRA-604, ROOM 123'
      fdlenN(3,numfdN)=31
      fieldN(4,numfdN)='GAMMA SPECTRUM ACQUISITION'
      fdlenN(4,numfdN)=26
C
      numfdN=numfdN+1
      fieldN(1,numfdN)='G. K. TAYLOR (GENE)'
      fdlenN(1,numfdN)=19
      fieldN(2,numfdN)='6-4041'
      fdlenN(2,numfdN)=6
      fieldN(3,numfdN)='RML, MS 7111; TRA-604, ROOM 123'
      fdlenN(3,numfdN)=31
      fieldN(4,numfdN)='GAMMA SPECTRUM ACQUISITION'
```

```
      fdlenN(4,numfdN)=26
C
      numfdN=numfdN+1
      fieldN(1,numfdN)='L. A. WEINRICH (LOU)'
      fdlenN(1,numfdN)=20
      fieldN(2,numfdN)='6-4404'
      fdlenN(2,numfdN)=6
      fieldN(3,numfdN)='RADIOCHEMISTRY, MS 7111; TRA-661, ROOM 129/130'
      fdlenN(3,numfdN)=46
      fieldN(4,numfdN)='BETA ANALYSES'
      fdlenN(4,numfdN)=13
C
      numfdN=numfdN+1
      fieldN(1,numfdN)='R. P. WELLS (RICH)'
      fdlenN(1,numfdN)=18
      fieldN(2,numfdN)='6-7870'
      fdlenN(2,numfdN)=6
      fieldN(3,numfdN)='RADIOCHEMISTRY, MS 7111; TRA-661, ROOM 129/130'
      fdlenN(3,numfdN)=46
      fieldN(4,numfdN)='BETA ANALYSES'
      fdlenN(4,numfdN)=13
C
      do i=1,numfdN
         fdrowN(i)=i+1
         fdcolN(i)=2
         ficolN(i)=25
         enddo
```

I. DISPLY - FORTRAN ROUTINE FOR DISPLAYING DATA INPUT SCREENS USING VMS SCREEN MANAGEMENT SYSTEM ROUTINES

```
      subroutine disply(field,fdatin,fldlen,fldrow,fldcol,fincol,maxnoc,
     . fldrnd,finrnd,numfld,indaus,fintyp,ifnulf,ifreqd,ifmodi,iditto,
     . ifclst,ibegin,subttl)
C
C Subroutine for displaying data entry fields.
C
C Component of SAMPLE_TRACKING Version 1 completed June 6, 1991 by D. A. Femec.
C
C Screen design based on E. Wayne Killian's ADD_SAMPLE routine of 3/21/90.
C
C Declaration statements.
C
      include '($smgdef)'    !SMG Definitions.
      include '($smgmsg)'    !SMG Condition values.
      include '($smgtrmptr)'    !SMG Terminal and printer definitions.
      include '($trmdef)'
      include 'gap$src:vterm.inc'
      integer*4 termod,ipaste,mvideo,lvideo,hvideo,keybrd,  !Pasteboard values.
     . indaus,    !Add/Update/Search indicator.
     . ibegin,    !The field on which to begin the cursor.
     . iwhich    !Which custodian has been selected from CHOOSE.
      logical ifcont,    !Flag whether to continue in display loops.
     . ifclst    !Flag whether custodian name list applies.
      common /paste_board/ termod,ipaste,mvideo,lvideo,hvideo,keybrd
      integer*4 ferror,ffixed,finput,foptnl,freqd
      common /fldscr/ ferror,ffixed,finput,foptnl,freqd
      character subttl*(*)    !Subtitle for display.
      character*80 field(*),fdatin(*),findat,blanks
      character*1 spaces(80)/80*' '/,fintyp(*)
      integer*4 fldlen(*),fldrow(*),fldcol(*),fincol(*),maxnoc(*),
     . fldrnd(3,*),finrnd(3,*),    !Field renditions (add/edit/search).
     . numfld    !The number of fields.
      integer*2 daymnt(12)/31,28,31,30,31,30,31,31,30,31,30,31/,
     . ifnulf(*),    !Null indicator (-1 if null, 0 otherwise).
     . ifreqd(*),    !Points to if-required dependent field (except for Search).
     . iditto(*)    !Points to allowed field for dittoing.
      logical ifmodi(*)    !Have any of the fields been modified?
      equivalence (blanks(1:1),spaces(1))
C
C    Custodian name list field descriptors and sizes.
C
      character*80 fieldN(4,15)
      integer*4 fdlenN(4,15),fdrowN(15),fdcolN(15),ficolN(15),
     . numfdN    !The number of fields.
      common /lsdatN/ fieldN,fdlenN,fdrowN,fdcolN,ficolN,numfdN
C
C Beginning of the subroutine.
C
C Renew the keyboard.
C
      call smg$delete_virtual_keyboard(keybrd)
      call smg$create_virtual_keyboard(keybrd)
C
C Display the subtitle if there is one.
C
      if(len(subttl).ne.0)then
         if(subttl.ne.blanks(1:len(subttl)))then    !If there is a subtitle . .
.
            call smg$put_chars(mvideo,subttl,2,(78-len(subttl)),,freqd)
            endif
         endif
```

```
C
C Begin display of fields.
C
      do i=1,numfld
         call smg$put_chars(mvideo,field(i)(1:fldlen(i)),fldrow(i),
     .     fldcol(i),,fldrnd(indaus,i))
         if(fdatin(i).ne.blanks)then   !Display what data already exists.
               call smg$put_chars(mvideo,fdatin(i)(1:maxnoc(i)),
     .              fldrow(i),fincol(i),,finrnd(indaus,i))
            else   !Otherwise display blanks.
               call smg$put_chars(mvideo,blanks(1:maxnoc(i)),
     .              fldrow(i),fincol(i),,finrnd(indaus,i))
            endif
C
C    Set all if-modified flags to false.
C
         ifmodi(i)=.false.
         enddo
C
C    Screen directives.
C
      call smg$put_chars(mvideo,'^Z  ',22,4,,smg$m_reverse)
      call smg$put_chars(mvideo,'when finished with screen.',22,7)
      call smg$put_chars(mvideo,'PF1',22,36,,smg$m_reverse)
      call smg$put_chars(mvideo,'aborts program.',22,40)
      call smg$put_chars(mvideo,'PF2',22,58,,smg$m_reverse)
      call smg$put_chars(mvideo,'displays help.',22,62)
C
C Begin scanning the screen as requested.
C
      ifcont=.true.
      ifield=ibegin-1
  100 ifield=ifield+1
      if(fldrnd(indaus,ifield).eq.ffixed)goto 100
  110 if(ifclst.and.((ifield.eq.11).or.(ifield.eq.18)))then
         call choose(iwhich,fieldN,fdlenN,fdrowN,fdcolN,ficolN,numfdN)
         if(iwhich.ne.0)then   !Assign list values to appropriate fields.
            do i=1,4
               ii=ifield+i-1
               fdatin(ii)(1:maxnoc(ii))=fieldN(i,iwhich)(1:maxnoc(ii))
               call smg$put_chars(mvideo,fdatin(ii)(1:maxnoc(ii)),
     .              fldrow(ii),fincol(ii),,finrnd(indaus,ii))
               ifmodi(ii)=.true.
               enddo
            call idate(imonth,iday,iyear)
            write(fdatin(ifield+4)(1:6),fmt='(3i2.2)')imonth,iday,iyear
            call smg$put_chars(mvideo,
     .          fdatin(ifield+4)(1:maxnoc(ifield+4)),fldrow(ifield+4),
     .          fincol(ifield+4),,finrnd(indaus,(ifield+4)))
            ifmodi(ifield+4)=.true.
            goto 600
            endif
         endif
  120 call smg$set_cursor_abs(mvideo,fldrow(ifield),fincol(ifield))
      call smg$read_string(keybrd,findat,,(maxnoc(ifield)+1),termod,,,
     . nchrin,itrmch,mvideo,,finrnd(indaus,ifield))
C
C Was PF1 the terminator character?  Then cleanly terminate the program.
C
      if(itrmch.eq.smg$k_trm_pf1)then
         call smg$erase_display(mvideo)
         call smg$unpaste_virtual_display(mvideo,ipaste)
         close(1,status='delete')
         stop
         endif
```

```
C
C Was "Insert Here" the terminator character?  Then ditto entry if possible.
C
      if(itrmch.eq.smg$k_trm_insert_here)then
         if(iditto(ifield).ne.0)then
            fdatin(ifield)=fdatin(iditto(ifield))    !Ditto the data.
            call smg$put_chars(mvideo,    !Write the dittoed data.
     .        fdatin(ifield)(1:maxnoc(ifield)),fldrow(ifield),
     .        fincol(ifield),,finrnd(indaus,ifield))
            goto 400    !Assume that ditto data is correct since original is.
            endif
         endif
C
C Was PF2 or Help the terminator character?  Then display keypad.
C
      if((itrmch.eq.smg$k_trm_help).or.(itrmch.eq.smg$k_trm_pf2))then
         call keypad
         goto 120    !Go back to accept new input from current field.
         endif
C
C Check data input against requirements.
C
  300 if(nchrin.gt.maxnoc(ifield))then
         call smg$erase_chars(mvideo,nchrin,fldrow(ifield),
     .     fincol(ifield))
         goto 500
         endif
      if(nchrin.ne.0)then    !So as to save entries.
            ifmodi(ifield)=.true.
            if(nchrin.lt.maxnoc(ifield))    !Pad each entry with blanks.
     .        findat((nchrin+1):maxnoc(ifield))=blanks(1:(maxnoc(ifield)-
     .        nchrin))
            if(fdatin(ifield).ne.blanks)then    !Rewrite the entry.
               call smg$erase_chars(mvideo,(maxnoc(ifield)+1),
     .           fldrow(ifield),fincol(ifield))
               call smg$put_chars(mvideo,findat(1:maxnoc(ifield)),
     .           fldrow(ifield),fincol(ifield),,finrnd(indaus,ifield))
               endif
            fdatin(ifield)=findat(1:nchrin)//blanks(1:(80-nchrin))  !Add blanks.
            if(fintyp(ifield).eq.'a')goto 400  !Free entry field.
            if(fintyp(ifield).eq.'d')then   !Dates between 1/1/1990-12/31/2079.
               if(nchrin.ne.6)goto 500    !Must have 6 characters.
               read(findat(5:6),fmt='(i2)',err=500,end=500)iyear
               if((iyear.lt.0).or.
     .           ((iyear.ge.80).and.(iyear.le.89)))goto 500
               if(mod(iyear,4).eq.0)then
                     daymnt(2)=29    !29 days in February in a leap year.
                  else
                     daymnt(2)=28    !28 days in February in a non-leap year.
                  endif
               read(findat(1:2),fmt='(i2)',err=500,end=500)imonth
               if((imonth.lt.1).or.(imonth.gt.12))goto 500
               read(findat(3:4),fmt='(i2)',err=500,end=500)iday
               if((iday.lt.1).or.(iday.gt.daymnt(imonth)))goto 500
               goto 400
               endif
            if(fintyp(ifield).eq.'l')then   !Required field length.
               if(nchrin.eq.maxnoc(ifield))goto 400
               goto 500
               endif
            if(fintyp(ifield).eq.'n')then   !Number, real or integer.
               ndecml=0
               do 350 i=1,nchrin
                  inumbr=ichar(findat(i:i))
                  if((inumbr.ge.48).and.(inumbr.le.57))goto 350   !Digits okay.
```

```
                  if(inumbr.eq.46)then   !A single decimal point is okay.
                     ndecml=ndecml+1
                     if(ndecml.eq.1)goto 350
                     endif
                  goto 500   !Bad input.
  350             continue
               goto 400
               endif
            if(fintyp(ifield).eq.'t')then   !Time of the form hh:mm.
               ncolon=0
               do 360 i=1,nchrin
                  inumbr=ichar(findat(i:i))
                  if((inumbr.ge.48).and.(inumbr.le.57))goto 360   !Digits okay.
                  if((inumbr.eq.58).and.(i.ge.2).and.(i.le.4))then
C
C    A single colon is okay in positions 2-4.
C
                     ncolon=ncolon+1
                     icolon=i
                     if(ncolon.eq.1)goto 360
                     endif
                  goto 500   !Bad input (not a number or colon, or a 2nd colon).
  360             continue
C
C    After the colon is located, the digits 0-5 are okay in the
C    following position, and a 0 or 1 in the second position
C    before the colon if that position is available.
C
               if(ichar(findat((icolon+1):(icolon+1))).gt.53)goto 500
               if((icolon-2).ge.1)then
                  if(ichar(findat((icolon-2):(icolon-2))).gt.49)goto 500
                  endif
               goto 400
               endif
            if(fintyp(ifield).eq.'u')then   !Time units field (m or h).
               if((findat(1:1).eq.'M').or.(findat(1:1).eq.'m').or.
     .           (findat(1:1).eq.'H').or.(findat(1:1).eq.'h'))goto 400
               goto 500
               endif
            if(fintyp(ifield).eq.'?')then   !Yes or no (y or n).
               if((findat(1:1).eq.'Y').or.(findat(1:1).eq.'N'))goto 400
               goto 500
               endif
            goto 400
         else   !Check for missing but required entries.
            if(fdatin(ifield).eq.blanks)then
               if(fldrnd(indaus,ifield).eq.freqd)goto 500   !Explicitly?
               if((ifreqd(ifield).ne.0).and.(indaus.ne.3))then   !Implicitly?
                  if(fdatin(ifreqd(ifield)).ne.blanks)then
                     if((fintyp(ifreqd(ifield)).eq.'?').and.   !Skip "No"s.
     .                 (fdatin(ifreqd(ifield))(1:1).ne.'Y'))goto 400
                     goto 500
                     endif
                  endif
               endif
            endif
C
C See if an error has been corrected.
C
  400 if(.not.ifcont)then
         ifcont=.true.
         call smg$put_chars(mvideo,   !Rewrite the field name.
     .     field(ifield)(1:fldlen(ifield)),fldrow(ifield),
     .     fldcol(ifield),,fldrnd(indaus,ifield))
         call smg$put_chars(mvideo,   !Rewrite the data.
```

```
     .    fdatin(ifield)(1:maxnoc(ifield)),fldrow(ifield),
     .    fincol(ifield),,finrnd(indaus,ifield))
          endif
       goto 600    !Entry okay, so now check terminator for next step.
C
C Error in data entry - do not proceed to next step until entry
C corrected.
C
  500 call smg$put_chars(mvideo,field(ifield)(1:fldlen(ifield)),
     . fldrow(ifield),fldcol(ifield),,(fldrnd(indaus,ifield).or.ferror))
       call smg$put_chars(mvideo,blanks(1:maxnoc(ifield)),
     . fldrow(ifield),fincol(ifield),,(finrnd(inaes,ifield).or.ferror))
       call smg$ring_bell(mvideo,3)
       fdatin(ifield)=blanks    !Reset the input field (erase the error).
       ifcont=.false.
       goto 110
C
C Check the terminator for the next step.
C
  600 call trmntr(ifcont,itrmch,ifield,numfld,fldrnd,maxnoc(ifield),
     . fldrow,fincol(ifield),finrnd(indaus,ifield),fdatin(ifield),
     . indaus,fintyp(ifield),ifmodi(ifield),fdatin(1)(1:12))
       if((.not.ifcont).and.(itrmch.eq.smg$k_trm_pf2))then
          if(fdatin(ifield).eq.blanks)then    !Can the error be erased?
             if(fldrnd(indaus,ifield).eq.freqd)goto 620    !Explicitly required?
            if((ifreqd(ifield).ne.0).and.(indaus.ne.3))then !Implicitly required?
               if(fdatin(ifreqd(ifield)).ne.blanks)then
                  if((fintyp(ifreqd(ifield)).eq.'?').and.    !Skip "No"s.
     .             (fdatin(ifreqd(ifield))(1:1).ne.'Y'))goto 610
                  goto 620
                  endif
               endif
            endif
  610     ifcont=.true.    !If an error has been erased . . .
          call smg$put_chars(mvideo,    !Rewrite the field name.
     .     field(ifield)(1:fldlen(ifield)),fldrow(ifield),
     .     fldcol(ifield),,fldrnd(indaus,ifield))
          call smg$put_chars(mvideo,    !Rewrite the data.
     .     fdatin(ifield)(1:maxnoc(ifield)),fldrow(ifield),
     .     fincol(ifield),,finrnd(indaus,ifield))
          endif
  620 if(ifcont)goto 110
C
C Check for missing required data and flag null fields before proceeding.
C
       findat=blanks
       do i=1,numfld
          if(fdatin(i).eq.blanks)then
                if(fldrnd(indaus,i).eq.freqd)then    !Explicitly?
                   ifield=i
                   goto 500
                   endif
                if((ifreqd(i).ne.0).and.(indaus.ne.3))then    !Implicitly?
                   if(fdatin(ifreqd(i)).ne.blanks)then
                      if((fintyp(ifreqd(i)).eq.'?').and.    !Skip "No"s.
     .                 (fdatin(ifreqd(i))(1:1).ne.'Y'))goto 700
                      ifield=i
                      goto 500
                      endif
                   endif
  700              ifnulf(i)=-1  !-1 if null, . . .
             else
                ifnulf(i)=0    !0 otherwise.
             endif
          enddo
```

```
C
C Renew the keyboard.
C
      call smg$delete_virtual_keyboard(keybrd)
      call smg$create_virtual_keyboard(keybrd)
      return
C
C End of the subroutine DISPLY.
C
      end
```

J. TRMNTR - FORTRAN ROUTINE FOR INTERPRETING TERMINATOR CODES FROM VMS SCREEN MANAGEMENT SYSTEM ROUTINE CALLS

```
      subroutine trmntr(ifcont,itrmch,ifield,numfld,fldrnd,maxnoc,
     . fdsrow,fincol,finrnd,findat,indaus,fintyp,ifmodi,smplid)
C
C Subroutine to act on "terminators."
C
C Component of SAMPLE_TRACKING Version 1 completed June 6, 1991 by D. A. Femec.
C
C Remember to skip "fixed" fields, i.e. those fields which cannot be modified.
C
C Declaration statements.
C
      include '($smgdef)'    !SMG Definitions.
      include '($smgmsg)'    !SMG Condition values.
      include '($smgtrmptr)'    !SMG Terminal and printer definitions.
      include '($trmdef)'
      include 'gap$src:vterm.inc'
      integer*4 termod,ipaste,mvideo,lvideo,hvideo,disply,  !Pasteboard values.
     . indaus    !Add/Update/Search indicator.
      logical ifcont,    !Flag whether to continue in display loops.
     . ifskip    !Flag whether to skip to next required.
      common /paste_board/ termod,ipaste,mvideo,lvideo,hvideo,disply
      character*80 findat,blanks
      character*12 smplid
      character*1 spaces(80)/80*' '/,fintyp
      integer*4 fldrnd(3,*),fdsrow(*),fincol,finrnd
      logical ifmodi,    !Has the field been modified (here, erased)?
     . iffrwd/.true./    !Is the currently selected direction forward?
      equivalence (blanks(1:1),spaces(1))
      integer*4 ferror,ffixed,finput,foptnl,freqd   !Display field renditions.
      common /fldscr/ ferror,ffixed,finput,foptnl,freqd
C
C Beginning of the subroutine.
C
      ifcont=.true.    !Set ifcont to true as default.  Leave ifmodi as it is.
      ifskip=(fldrnd(indaus,ifield).eq.freqd).and.(fintyp.eq.'?').and.
     . (findat(1:1).eq.'N')
C
C Advance?  Then set direction as forward, but do not move.
C
      if(itrmch.eq.smg$k_trm_kp4)then
         iffrwd=.true.
         return
         endif
C
C Backup?  Then set direction as reverse, but do not move.
C
      if(itrmch.eq.smg$k_trm_kp5)then
         iffrwd=.false.
         return
         endif
C
C Return, Enter, right arrow, or horizontal tab?  Advancing and
C Keypad 1 or 3?  Then move to the next field.
C
      if((itrmch.eq.smg$k_trm_cr).or.(itrmch.eq.smg$k_trm_enter).or.
     . (itrmch.eq.smg$k_trm_right).or.(itrmch.eq.smg$k_trm_ht).or.
     . (((itrmch.eq.smg$k_trm_kp1).or.(itrmch.eq.smg$k_trm_kp3)).and.
     . iffrwd))then
  100    ifield=ifield+1
```

```
         if(ifield.gt.numfld)then
            ifield=1
            ifskip=.false.   !ifskip is no longer meaningful.
            endif
         if(fldrnd(indaus,ifield).eq.ffixed)goto 100   !Skip if fixed.
C
C    If the starting field is required, needs a yes or no answer, and was
C    answered no, then skip to the next required field.
C
         if((ifskip).and.(fldrnd(indaus,ifield).ne.freqd))goto 100
         return
         endif
C
C Do or ^Z?  Then data input has ended for this screen and the
C display loop should not be continued.
C
      if((itrmch.eq.smg$k_trm_do).or.(itrmch.eq.smg$k_trm_ctrlz))then
         ifcont=.false.
         return
         endif
C
C PF1?  Then abort execution of the program.
C
      if(itrmch.eq.smg$k_trm_pf1)then
         call smg$erase_display(hvideo)    !Help display.
         call smg$unpaste_virtual_display(hvideo,ipaste)
         call smg$erase_display(lvideo)    !"Choose" display.
         call smg$unpaste_virtual_display(lvideo,ipaste)
         call smg$erase_display(mvideo)    !Main display.
         call smg$unpaste_virtual_display(mvideo,ipaste)
         close(1,status='delete')
         stop
         endif
C
C PF2?  Then display keypad.
C
      if((itrmch.eq.smg$k_trm_help).or.(itrmch.eq.smg$k_trm_pf2))then
         call keypad
         return
         endif
C
C PF4, Remove, or Keypad minus, 6, or comma?  Then erase the field.
C
      if((itrmch.eq.smg$k_trm_pf4).or.(itrmch.eq.smg$k_trm_minus).or.
     . (itrmch.eq.smg$k_trm_comma).or.(itrmch.eq.smg$k_trm_kp6).or.
     . (itrmch.eq.smg$k_trm_remove))then
         call smg$erase_chars(mvideo,maxnoc,fdsrow(ifield),
     .    fincol)
         call smg$put_chars(mvideo,blanks(1:maxnoc),
     .    fdsrow(ifield),fincol,,finrnd)
         findat=blanks
         ifmodi=.true.   !Erasing a field changes it.
         return
         endif
C
C Left arrow?  Backing up and Keypad 1 or 3?  Then move to the previous field.
C
      if((itrmch.eq.smg$k_trm_left).or.(((itrmch.eq.smg$k_trm_kp1).or.
     . (itrmch.eq.smg$k_trm_kp3)).and.(.not.iffwrd)))then
  200    ifield=ifield-1
         if(ifield.lt.1)ifield=numfld
         if(fldrnd(indaus,ifield).eq.ffixed)goto 200   !Skip if fixed.
         return
         endif
C
```

```
C Up arrow?  Backing up and end-of-line?  Then move up one line.
C
      if((itrmch.eq.smg$k_trm_up).or.((itrmch.eq.smg$k_trm_kp2).and.
     . (.not.iffwrd)))then
         iforig=ifield
  300    if(ifield.eq.1)then
            ifield=numfld
            if(fldrnd(indaus,ifield).eq.ffixed)goto 310   !Skip if fixed.
            return
            endif
  310    ifield=ifield-1
         if(ifield.lt.1)ifield=numfld
         if(fldrnd(indaus,ifield).eq.ffixed)goto 300   !Skip if fixed.
         if((fdsrow(ifield).ne.fdsrow(iforig)).or.    !If new row found or
     .    (ifield.eq.iforig))return                    !if no new row findable.
         goto 300
         endif
C
C Down arrow or Keypad 0?  Then move down one line.
C
      if((itrmch.eq.smg$k_trm_down).or.(itrmch.eq.smg$k_trm_kp0))then
         iforig=ifield
  400    if(ifield.eq.numfld)then
            ifield=1
            if(fldrnd(indaus,ifield).eq.ffixed)goto 410   !Skip if fixed.
            return
            endif
  410    ifield=ifield+1
         if(ifield.gt.numfld)ifield=1
         if(fldrnd(indaus,ifield).eq.ffixed)goto 400   !Skip if fixed.
         if((fdsrow(ifield).ne.fdsrow(iforig)).or.    !If new row found or
     .    (ifield.eq.iforig))return                    !if no new row findable.
         goto 400
         endif
C
C Advancing and end-of-line?  Then move to end of current line, or, if
C presently at end of current line, move to end of next line.
C
      if((itrmch.eq.smg$k_trm_kp2).and.iffwrd)then
         iforig=ifield
  500    if(ifield.eq.numfld)then
            ifield=1
            if(fldrnd(indaus,ifield).eq.ffixed)goto 510   !Skip if fixed.
            return
            endif
  510    ifield=ifield+1
         if(ifield.gt.numfld)ifield=1
         if(fldrnd(indaus,ifield).eq.ffixed)goto 500   !Skip if fixed.
         if((fdsrow(ifield).eq.fdsrow(iforig)).and.   !End of current row.
     .    (fdsrow(ifield+1).ne.fdsrow(iforig)).and.
     .    (ifield.ne.iforig))return
         if(((fdsrow(ifield).ne.fdsrow(iforig)).and.   !End of next row found or
     .    (fdsrow(iforig).ne.fdsrow(iforig+1))).or.
     .    (ifield.eq.iforig))return                    !if no next row findable.
         goto 500
         endif
C
C Prev Screen?  Advancing and Keypad 7?  Go to the first not-fixed field.
C
      if((itrmch.eq.smg$k_trm_prev_screen).or.
     . ((itrmch.eq.smg$k_trm_kp7).and.iffrwd))then
         ifield=1
  600    if(fldrnd(indaus,ifield).ne.ffixed)return   !Skip if fixed.
         ifield=ifield+1
         goto 600
```

```
         endif
C
C Next Screen?  Backing up and Keypad 7?  Go to the last not-fixed field.
C
      if((itrmch.eq.smg$k_trm_next_screen).or.
     . ((itrmch.eq.smg$k_trm_kp7).and.(.not.iffrwd)))then
         ifield=numfld
  700    if(fldrnd(indaus,ifield).ne.ffixed)return   !Skip if fixed.
         ifield=ifield-1
         goto 700
         endif
C
C F17?  Print the barcode (field 1) if not blank.
C
      if((itrmch.eq.smg$k_trm_f17).and.(smplid.ne.blanks(1:12)))then
         print *,char(155),'5i'    !Turn on printer controller mode.
         print *,char(24),';',char(27),'E1;D,',smplid,';',char(23),';'
         print *,char(155),'4i'    !Turn off printer controller mode.
         endif
C
C Unrecognized terminator?  Stay at current field.
C
      return
C
C End of the subroutine TRMNTR.
C
      end
```

K. CHOOSE - FORTRAN ROUTINE FOR PRESENTING A LIST OF CHOICES USING VMS SCREEN MANAGEMENT SYSTEM ROUTINES

```
      subroutine choose(iwhich,fldata,fldlen,fldrow,fldcol,fincol,
     . numfld)
C
C Subroutine to choose from a list while in a data entry display.
C The list is created as a separate screen.  Selection of an entry
C terminates display of the list.
C
C Component of SAMPLE_TRACKING Version 1 completed June 6, 1991 by D. A. Femec.
C
C Declaration statements.
C
      include '($smgdef)'    !SMG Definitions.
      include '($smgmsg)'    !SMG Condition values.
      include '($smgtrmptr)'    !SMG Terminal and printer definitions.
      include '($trmdef)'
      include 'gap$src:vterm.inc'
      character*80 fldata(4,*),blanks,findat(20)
      character*1 spaces(80)/80*' '/
      integer*4 fldlen(4,*),fldrow(*),fldcol(*),fincol(*),
     . numfld,    !The number of fields.
     . iwhich,    !Which custodian has been selected.
     . termod,ipaste,mvideo,lvideo,hvideo,keybrd,    !Pasteboard values.
     . fldrnd(4,1)/4*smg$m_underline/    !Dummy renditions for trmntr.
      logical ifcont,ifmodi
      equivalence (blanks(1:1),spaces(1))
      common /paste_board/ termod,ipaste,mvideo,lvideo,hvideo,keybrd
C
C Beginning of the subroutine.
C
C Present the list subscreen.
C
      call smg$create_virtual_display(17,45,lvideo,smg$m_border)
      call smg$paste_virtual_display(lvideo,ipaste,3,30)
C
C Renew the keyboard.
C
      call smg$delete_virtual_keyboard(keybrd)
      call smg$create_virtual_keyboard(keybrd)
C
C Begin display of fields.
C
      do i=1,numfld
         findat(i)=blanks
         call smg$put_chars(lvideo,fldata(1,i)(1:fldlen(1,i)),
     .     fldrow(i),fldcol(i))
         call smg$put_chars(lvideo,blanks(1:1),fldrow(i),fincol(i),,
     .     smg$m_underline)
         enddo
C
C    Screen directives.
C
      call smg$put_chars(lvideo,'^Z',17,2,,smg$m_reverse)
      call smg$put_chars(lvideo,'when done.',17,5)
      call smg$put_chars(lvideo,'PF1',17,17,,smg$m_reverse)
      call smg$put_chars(lvideo,'to abort.',17,21)
      call smg$put_chars(lvideo,'PF2',17,32,,smg$m_reverse)
      call smg$put_chars(lvideo,'for help.',17,36)
C
C Begin scanning the screen as requested.
C
      ifcont=.true.
```

```
      ifield=1
  100 call smg$set_c sor_abs(lvideo,fldrow(ifield),fincol(ifield))
      call smg$read_ ring(keybrd,findat(ifield),,2,termod,,
     . nchrin,itrmch,lvideo,,smg$m_underline)
      if(nchrin.gt.1)then
         call smg$erase_chars(lvideo,nchrin,fldrow(ifield),
     .    fincol(ifield))
         call smg$put_chars(lvideo,fldata(1,i)(1:fldlen(1,i)),
     .    fldrow(i),fldcol(i))
         call smg$put_chars(lvideo,blanks(1:1),fldrow(i),fincol(i),,
     .    smg$m_underline)
         goto 100
         endif
C
C Check the terminator for the next step.
C
      call trmntr(ifcont,itrmch,ifield,numfld,fldrnd,1,
     . fldrow,fincol(ifield),smg$m_underline,findat(ifield),3,'a',
     . ifmodi,blanks(1:12))
      if(ifcont)goto 100
C
C Return the first entry marked to the calling routine.
C
      iwhich=0
      do i=1,numfld
         if(findat(i)(1:1).ne.blanks(1:1))then   !Pick the first item marked.
            iwhich=i
            goto 900
            endif
         enddo
C
C Remove the list subscreen before returning to calling routine.
C
  900 call smg$erase_display(lvideo)
      call smg$unpaste_virtual_display(lvideo,ipaste)
C
C Renew the keyboard.
C
      call smg$delete_virtual_keyboard(keybrd)
      call smg$create_virtual_keyboard(keybrd)
      return
C
C End of the subroutine CHOOSE.
C
      end
```

L. KEYPAD - FORTRAN ROUTINE FOR DISPLAYING KEYPAD MAPPING USING VMS SCREEN MANAGEMENT SYSTEM ROUTINE

```
      subroutine keypad
C
C Subroutine to display the keypad assignments.
C
C Component of SAMPLE_TRACKING Version 1 completed June 6, 1991 by D. A. Femec.
C
C Declaration statements.
C
      include '($smgdef)'    !SMG Definitions.
      include '($smgmsg)'    !SMG Condition values.
      include '($smgtrmptr)'    !SMG Terminal and printer definitions.
      include '($trmdef)'
      include 'gap$src:vterm.inc'
      character*80 findat
      character*7 asgmnt(2,34)/
     . ' Help  ',' ',    !"Help" box.
     . ' Next  ',' Screen',    !"Do" box.
     . '       ',' ',    !"Find" box.
     . ' Ditto ',' Entry ',    !"Insert here" box.
     . ' Delete',' Entry ',    !"Remove" box.
     . '       ',' ',    !"Select" box.
     . ' Top of',' Screen',    !"Prev Screen" box.
     . ' End of',' Screen',    !"Next Screen" box.
     . ' Move  ',' Up    ',    !Up arrow box.
     . ' Move  ',' Left  ',    !Left arrow box.
     . ' Move  ',' Down  ',    !Down arrow box.
     . ' Move  ',' Right ',    !Right arrow box.
     . ' Print ','Barcode',    !F17 box.
     . '       ',' ',    !F18 box.
     . '       ',' ',    !F19 box.
     . '       ',' ',    !F20 box.
     . ' Abort ','Program',    !PF1 box.
     . ' Help  ',' ',    !PF2 box.
     . '       ',' ',    !PF3 box.
     . ' Delete',' Entry ',    !PF4 box.
     . 'Top/End',' Screen',    !7 box.
     . '       ',' ',    !8 box.
     . '       ',' ',    !9 box.
     . ' Delete',' Entry ',    !- (minus) box.
     . 'Advance',' ',    !4 box.
     . ' Backup',' ',    !5 box.
     . ' Delete',' Entry ',    !6 box.
     . ' Delete',' Entry ',    !, (comma) box.
     . ' Next  ',' Entry ',    !1 box.
     . ' End of',' Line  ',    !2 box.
     . ' Next  ',' Entry ',    !3 box.
     . ' Next  ',' Entry ',    !Enter box.
     . ' Next  ',' Line  ',    !0 box.
     . '       ',' '/    !. (period) box.
      integer*4 termod,ipaste,mvideo,lvideo,hvideo,keybrd,   !Pasteboard values.
     . keybox(4,34)/
     .  1,2,4,10,       !"Help" box.
     .  1,10,4,26,      !"Do" box.
     .  5,2,8,10,       !"Find" box.
     .  5,10,8,18,      !"Insert here" box.
     .  5,18,8,26,      !"Remove" box.
     .  8,2,11,10,      !"Select" box.
     .  8,10,11,18,     !"Prev Screen" box.
     .  8,18,11,26,     !"Next Screen" box.
     .  11,10,14,18,    !Up arrow box.
     .  14,2,17,10,     !Left arrow box.
```

```
     .  14,10,17,18,    !Down arrow box.
     .  14,18,17,26,    !Right arrow box.
     .  1,34,4,42,      !F17 box.
     .  1,42,4,50,      !F18 box.
     .  1,50,4,58,      !F19 box.
     .  1,58,4,66,      !F20 box.
     .  5,34,8,42,      !PF1 box.
     .  5,42,8,50,      !PF2 box.
     .  5,50,8,58,      !PF3 box.
     .  5,58,8,66,      !PF4 box.
     .  8,34,11,42,     !7 box.
     .  8,42,11,50,     !8 box.
     .  8,50,11,58,     !9 box.
     .  8,58,11,66,     !- (minus) box.
     .  11,34,14,42,    !4 box.
     .  11,42,14,50,    !5 box.
     .  11,50,14,58,    !6 box.
     .  11,58,14,66,    !, (comma) box.
     .  14,34,17,42,    !1 box.
     .  14,42,17,50,    !2 box.
     .  14,50,17,58,    !3 box.
     .  14,58,20,66,    !Enter box.
     .  17,34,20,50,    !0 box.
     .  17,50,20,58/,    !. (period) box.
     .  numbox/34/
      common /paste_board/ termod,ipaste,mvideo,lvideo,hvideo,keybrd
C
C Beginning of the subroutine.
C
C Present the list subscreen.
C
      call smg$create_virtual_display(20,67,hvideo,smg$m_border)
      call smg$paste_virtual_display(hvideo,ipaste,3,10)
C
C Renew the keyboard.
C
      call smg$delete_virtual_keyboard(keybrd)
      call smg$create_virtual_keyboard(keybrd)
C
C Display the keypad key assignments.
C
      call smg$put_chars_highwide(hvideo,'Keypad Help',18,3)
      do i=1,numbox
         call smg$draw_rectangle(hvideo,keybox(1,i),keybox(2,i),
     .    keybox(3,i),keybox(4,i))   !Draw the outline of the box.
         if(asgmnt(1,i).ne.'       ')call smg$put_chars(hvideo,
     .    asgmnt(1,i),(keybox(1,i)+1),(keybox(2,i)+1))
         if(asgmnt(2,i).ne.'       ')call smg$put_chars(hvideo,
     .    asgmnt(2,i),(keybox(1,i)+2),(keybox(2,i)+1))
         enddo
C
C    Prompt to stop display.
C
      call smg$put_chars(hvideo,
     . 'Please strike any key to return.',19,1)
      call smg$set_cursor_abs(hvideo,20,1)
      call smg$read_string(keybrd,findat,,1,termod,,,nchrin,itrmch,
     . hvideo)
C
C Remove the list subscreen before returning to calling routine.
C
  900 call smg$erase_display(hvideo)
      call smg$unpaste_virtual_display(hvideo,ipaste)
C
C Renew the keyboard.
```

```
C
      call smg$delete_virtual_keyboard(keybrd)
      call smg$create_virtual_keyboard(keybrd)
      return
C
C End of the subroutine KEYPAD.
C
      end
```

M. RVALUE - FORTRAN ROUTINE FOR CHARACTER STRING-TO-NUMERICAL VALUE CONVERSIONS

```
      real*4 function rvalue(cstrng,maxnoc)
C
C Function to obtain the real numerical value corresponding to the
C contents of a character string consisting only of a real or integer
C number and padding blanks.  Works properly for integers, real, or real
C -exponential numbers.
C
C Component of SAMPLE_TRACKING Version 1 completed June 6, 1991 by D. A. Femec.
C
C Declaration statements.
C
      character*(*) cstrng
      integer*4 maxnoc,nochar,idecml,inumbr
C
C Beginning of the function.
C
C Find the length of the string, excluding trailing blanks.
C
      do i=maxnoc,1,-1
         if(cstrng(i:i).ne.' ')then
            nochar=i
            goto 100
            endif
         enddo
C
C Locate the decimal point in the character string, if there is one.
C
  100 idecml=index(cstrng,'.')
C
C    If there is no decimal point, treat the character string as
C    representing an integer.
C
       if(idecml.eq.0)then
             read(cstrng(1:nochar),fmt='(i<nochar>)')inumbr
             rvalue=real(inumbr)
C
C    If there is a decimal point, treat the character string as
C    representing an real (or real-exponential) number.
C
          else
             read(cstrng(1:nochar),fmt='(f<nochar>.<nochar-idecml>)')
     .        rvalue
          endif
       return
C
C End of the function RVALUE.
C
      end
```

N. CVALUE - FORTRAN ROUTINE FOR NUMERICAL-TO-CHARACTER STRING VALUE CONVERSIONS

```
      subroutine cvalue(cstrng,maxnoc,rnumbr)
C
C Subroutine to return the most complete character string representation
C of real numerical value.  Works properly for real or real-exponential
C numbers.   Will return an integer representation if appropriate.
C
C 'Most complete' here refers to how close the real numerical value
C obtained from the character string produced is to the original real
C numerical value.
C
C Declaration statements.
C
      character*(*) cstrng
      character*80 cstrg1,cstrg2,blanks
      character*1 spaces(80)/80*' '/
      real*4 rnumbr,rnmbr1,rnmbr2,rvalue
      integer*4 maxnoc,n
      logical ifFbad,ifEbad
      equivalence (blanks(1:1),spaces(1))
C
C Beginning of the routine.
C
C In both F format and E format cases, use 'err' to determine whether
C the number can even be successfully written in the desired character
C string.  The greatest number of figures beyond the decimal point
C obtainable is desired in both cases.
C
C Using the F format . . .
C
      n=maxnoc
  100 write(cstrg1(1:maxnoc),fmt='(f<maxnoc>.<n>)',err=110)rnumbr
      ifFbad=.false.
      goto 120
  110 n=n-1
      if(n.eq.0)then   !Cannot use F format.
         ifFbad=.true.
         cstrg1='0.0'//blanks(1:77)
         goto 120
         endif
      goto 100
  120 rnmbr1=rvalue(cstrg1,maxnoc)   !The real value described by the string.
C
C Using the E format (which should work even when the F format does not) . . .
C
      n=maxnoc
  200 write(cstrg2(1:maxnoc),fmt='(e<maxnoc>.<n>)',err=210)rnumbr
      ifEbad=.false.
      goto 220
  210 n=n-1
      if(n.eq.0)then   !Cannot use E format.
         ifEbad=.true.
         cstrg2='0.0'//blanks(1:77)
         goto 220
         endif
      goto 200
  220 rnmbr2=rvalue(cstrg2,maxnoc)
C
C Check for unusable format cases.
C
      if(ifFbad)then   !Use E format value, then return.
         cstrng=cstrg2(1:maxnoc)
```

```
         return
         endif
      if(ifEbad)then   !Use F format value, or I format if appropriate.
         if(aint(rnumbr).eq.rnumbr)then   !I format is appropriate.
               write(cstrng(1:maxnoc),fmt='(i<maxnoc>)')int(rnumbr)
            else   !Use F format.
               cstrng=cstrg1(1:maxnoc)
            endif
         return
         endif
C
C Compare the back translated results to see which character string
C representation is the most complete.
C
      if(abs(rnmbr1-rnumbr).le.abs(rnmbr2-rnumbr))then   !F is most complete.
         if(aint(rnumbr).eq.rnumbr)then   !I format is appropriate.
               write(cstrng(1:maxnoc),fmt='(i<maxnoc>)')int(rnumbr)
            else   !Use F format.
               cstrng=cstrg1(1:maxnoc)
            endif
         else
            cstrng=cstrg2(1:maxnoc)   !E format value is the most complete.
         endif
      return
C
C End of the subroutine CVALUE.
C
      end
```

O. DAYCNT AND CHRDAT - FORTRAN ROUTINES FOR CONVERSIONS BETWEEN MMDDYY DATES AND INTEGER DATE COUNTS

```
C Functions for converting dates from mmddyy to day count from January 1,
C  1990 (day 1) and back again.
C
C Component of SAMPLE_TRACKING Version 1 completed June 6, 1991 by D. A. Femec.
C
C For counting days, note that all years evenly divisible by four for
C the next 100+ years are leap years (the year 2000, being evenly
C divisible by 400, is a leap year).
C
C The function daycnt is good through December 31, 2079; beyond this the
C value of daycnt is too large to store in an integer*4 variable.
C
C The nth element of edamnt contains the cumulative number of days
C expired in all months prior to the nth month.
C
      function daycnt(datchr)
C
C Declaration statements.
C
      character*6 datchr
      integer*4 daycnt
      integer*2 edamnt(12)/0,31,59,90,120,151,181,212,243,273,304,334/,
     . mm,dd,yy
C
C Beginning of the function.
C
      read(datchr(1:2),fmt='(i2)')mm
      read(datchr(3:4),fmt='(i2)')dd
      read(datchr(5:6),fmt='(i2)')yy
      if(yy.lt.90)yy=yy+100   !Correct for change of century.
      daycnt=(yy-90)*365+int((yy-89)/4)+edamnt(mm)+dd
      if((mod(yy,4).eq.0).and.(mm.gt.2))daycnt=daycnt+1  !Past leap day.
      return
C
C End of the function DAYCNT.
C
      end
      function chrdat(cntday)
C
C Declaration statements.
C
      character*6 chrdat
      integer*4 cntday
      integer*2 daymnt(12)/31,28,31,30,31,30,31,31,30,31,30,31/,
     . mm,dd,yy
C
C Beginning of the function.
C
      yy=90+int(cntday/365)   !Rough guess of year (no leap years counted).
      yy=90+int((cntday-int((yy-89)/4)-1)/365)   !Refined guess.
      dd=cntday-(yy-90)*365-int((yy-89)/4)
      if(mod(yy,4).eq.0)then
         daymnt(2)=29   !29 days in February in a leap year.
      else
         daymnt(2)=28   !28 days in February in a non-leap year.
      endif
      do i=1,11
         if((dd-daymnt(i)).le.0)then
            mm=i
```

```
            goto 10
            endif
         dd=dd-daymnt(i)
         enddo
      mm=12
10    if(yy.ge.100)yy=yy-100
      write(chrdat,fmt='(3i2.2)')mm,dd,yy
      return
C
C End of the function CHRDAT.
C
      end
```

P. SRCBLD - FORTRAN ROUTINE FOR CONSTRUCTING SQL SEARCHES

```
      subroutine srcbld(comnd1,lcmnd1,comnd2,lcmnd2,numfld,fdatin,
     . maxnoc,fintyp,atrbut)
C
C Subroutine for building the search command.
C
C Component of SAMPLE_TRACKING Version 1 completed June 6, 1991 by D. A. Femec.
C
C Declaration statements.
C
      character*3000 comnd1,comnd2
      integer*4 lcmnd1,lcmnd2
      character*80 fdatin(*),blanks
      character*30 atrbut(2,*)
      character*6 datchr
      character*1 spaces(80)/80*' '/,fintyp(*)
      integer*4 daycnt,maxnoc(*),numfld   !The number of fields.
      equivalence (blanks(1:1),spaces(1))
C
C Beginning of the subroutine.
C
C Find the "from" in the "from" clause.
C
      locfrm=index(comnd1(1:lcmnd1),'from')
      do i=1,numfld
         if(fdatin(i).ne.blanks)then
            if(index(comnd1(locfrm:lcmnd1),   !Entity not in "from" clause.
     .       atrbut(1,i)(1:(index(atrbut(1,i),' ')-1))).eq.0)then
               call apstrg(comnd1,lcmnd1,   !Add entity to "from" clause.
     .          (', '//atrbut(1,i)(1:(index(atrbut(1,i),' ')-1))),
     .          .false.)
               if(lcmnd2.eq.6)then   !Add SAMPLE_TRACKING_ID match.
                     call apstrg(comnd2,lcmnd2,(' '//
     .                atrbut(1,i)(1:(index(atrbut(1,i),' ')-1))//
     .                '.SAMPLE_TRACKING_ID=SAMPLE_INFORMATION'//
     .                '.SAMPLE_TRACKING_ID'),.false.)
                  else
                     call apstrg(comnd2,lcmnd2,(' and '//
     .                atrbut(1,i)(1:(index(atrbut(1,i),' ')-1))//
     .                '.SAMPLE_TRACKING_ID=SAMPLE_INFORMATION'//
     .                '.SAMPLE_TRACKING_ID'),.false.)
                  endif
               endif
            if(lcmnd2.eq.6)then   !First entry to "where" clause.
                  call apstrg(comnd2,lcmnd2,(' '//
     .             atrbut(1,i)(1:(index(atrbut(1,i),' ')-1))//'.'//
     .             atrbut(2,i)(1:(index(atrbut(2,i),' ')-1))),.false.)
               else   !Include the "and" for subsequent entries.
                  call apstrg(comnd2,lcmnd2,(' and '//
     .             atrbut(1,i)(1:(index(atrbut(1,i),' ')-1))//'.'//
     .             atrbut(2,i)(1:(index(atrbut(2,i),' ')-1))),.false.)
               endif
            if(fintyp(i).eq.'d')then   !Date field.
                  write(datchr,fmt='(i6)')daycnt(fdatin(i)(1:maxnoc(i)))
                  call apstrg(comnd2,lcmnd2,('='//datchr),.false.)
               else
                  if(fintyp(i).eq.'n')then   !Numeric fields.
                        call apstrg(comnd2,lcmnd2,('='//
     .                   fdatin(i)(1:maxnoc(i))),.false.)
                     else   !Enclose character fields in "s.
                        call apstrg(comnd2,lcmnd2,('="'//
     .                   fdatin(i)(1:maxnoc(i))//'"'),.false.)
```

```
                        endif
                    endif
                endif
            enddo
        return
C
C End of the subroutine SRCBLD.
C
        end
```

Q. APSTRG - FORTRAN ROUTINE FOR APPENDING CHARACTER STRINGS IN A SPACE-CONSERVING FASHION

```
      subroutine apstrg(main,mainln,segmnt,ifsnew)
C
C Subroutine to space-efficiently concatentate string SEGMNT to string
C MAIN.  The total length of MAIN is kept track of in MAINLN.
C
C Component of SAMPLE_TRACKING Version 1 completed June 6, 1991 by D. A. Femec.
C
C Declaration statements.
C
      character*3000 main
      character*(*) segmnt
      integer*4 mainln,seglen
      logical ifsnew   !Whether a string is being started anew.
C
C Beginning of the subroutine.
C
      if(ifsnew)mainln=0
      seglen=len(segmnt)
      main((mainln+1):(mainln+seglen))=segmnt
      mainln=mainln+seglen
      return
C
C End of the subroutine APSTRG.
C
      end
```