PNNL-14852, Volume 3

Pacific Northwest National Laboratory
Operated by Battelle for the U.S. Department of Energy

User Instructions for the Systems Assessment Capability, Rev. 1, Computer Codes

Volume 3: Utility Codes

P. W. Eslinger
R. L. Aaberg
C. A. Lopresti
T. B. Miley
W. E. Nichols
D. L. Strenge

September 2004

Prepared for the U.S. Department of Energy
under Contract DE-AC06-76RL01830

PACIFIC NORTHWEST NATIONAL LABORATORY
operated by
BATTELLE
for the
UNITED STATES DEPARTMENT OF ENERGY
under Contract DE-AC06-76RL01830

PNNL-14852, Volume 3

User Instructions for the Systems Assessment Capability, Rev. 1, Computer Codes

Volume 3: Utility Codes

P. W. Eslinger
R. L. Aaberg
C. A. Lopresti
T. B. Miley
W. E. Nichols
D. L. Strenge

September 2004

Prepared for
the U.S. Department of Energy
under Contract DE-AC06-76RL01830

Pacific Northwest National Laboratory
Richland, Washington 99352

Contents

Figures

Tables

1.0 Introduction and Background

In 1999, the U.S. Department of Energy (DOE) initiated the development of an assessment tool that will enable users to model the movement of contaminants from all waste sites at Hanford through the vadose zone, groundwater, and Columbia River and estimate the impact of contaminants on human health, ecology, and the local cultures and economy. This is an integrated system of computer models and databases used to assess the impact of waste remaining on the Hanford Site. This tool was named the System Assessment Capability (SAC). The SAC will help decision makers and the public evaluate the cumulative effects of contamination from Hanford.

The design of the SAC resulted from extensive interactions with Hanford projects, regulators, Tribal Nations, and stakeholders. The approach taken in the assessment follows that advanced by regulatory agencies, such as the U. S. Environmental Protection Agency (EPA) in its guidance on uncertainty analyses (Firestone et al. 1997) and ecological risk assessments (EPA 1998) and DOE in its radioactive waste management manual (DOE 1999a) and implementation guide (DOE 1999b) that support DOE Order 435.1 on radioactive waste management. The SAC also was designed with the intent to use it to perform the next composite analysis, an assessment first performed to satisfy Defense Nuclear Facility Safety Board recommendation 94-2. The approach taken is also consistent with the methods, characteristics, and controls associated with acceptable analyses as described by the Columbia River Comprehensive Impact Assessment (CRCIA) team (DOE 1998).

1.1 Overview of the SAC Systems Code

The SAC Systems Code is a tool used to simulate the migration of contaminants (analytes) present on the Hanford Site and assess the potential impacts of the analytes, including dose to humans, socio-cultural impacts, economic impacts, and ecological impacts. The system of codes includes existing computer programs, new computer programs, electronic data libraries, and data formatting processors (or data translators). The relationships among code modules that make up the SAC Systems Code are illustrated in Figure 1.1.

Major modules appearing on the left side of the diagram perform inventory and transport calculations providing estimates of the concentrations of analytes in various media. Modules shown on the right perform calculations related to impacts from the contaminated media. Impacts include potential effects on humans, the ecology of the area, the economy of the region, and the proximity of contaminants to social and cultural resources.

The general approach to handling uncertainty in SAC, Rev. 1, is a Monte Carlo approach. Conceptually, a value is generated for every stochastic parameter in the code (the entire sequence of modules from inventory through transport and impacts) and then the simulation is executed to obtain an output value or result. This process is often called one *realization*. The entire process is then repeated, obtaining another result that is different from the first, but as equally likely to occur as the first result. After repeating this process a number of times, one has a set of equally likely consequences that represent the statistical distribution of all outcomes. Several specialized sampling techniques have been developed to reduce the number of realizations required in a Monte Carlo analysis to obtain a satisfactory description of the output distribution. One of the techniques, called Latin Hypercube Sampling (Iman and Conover 1982), has proven successful for mass transport applications in groundwater systems. The general Monte Carlo approach still applies, and the specific values of the input parameters are chosen from the same statistical distributions; however, the sampling scheme spreads the values in a way that reduces sampling variability while also supporting a correlation structure between input variables.

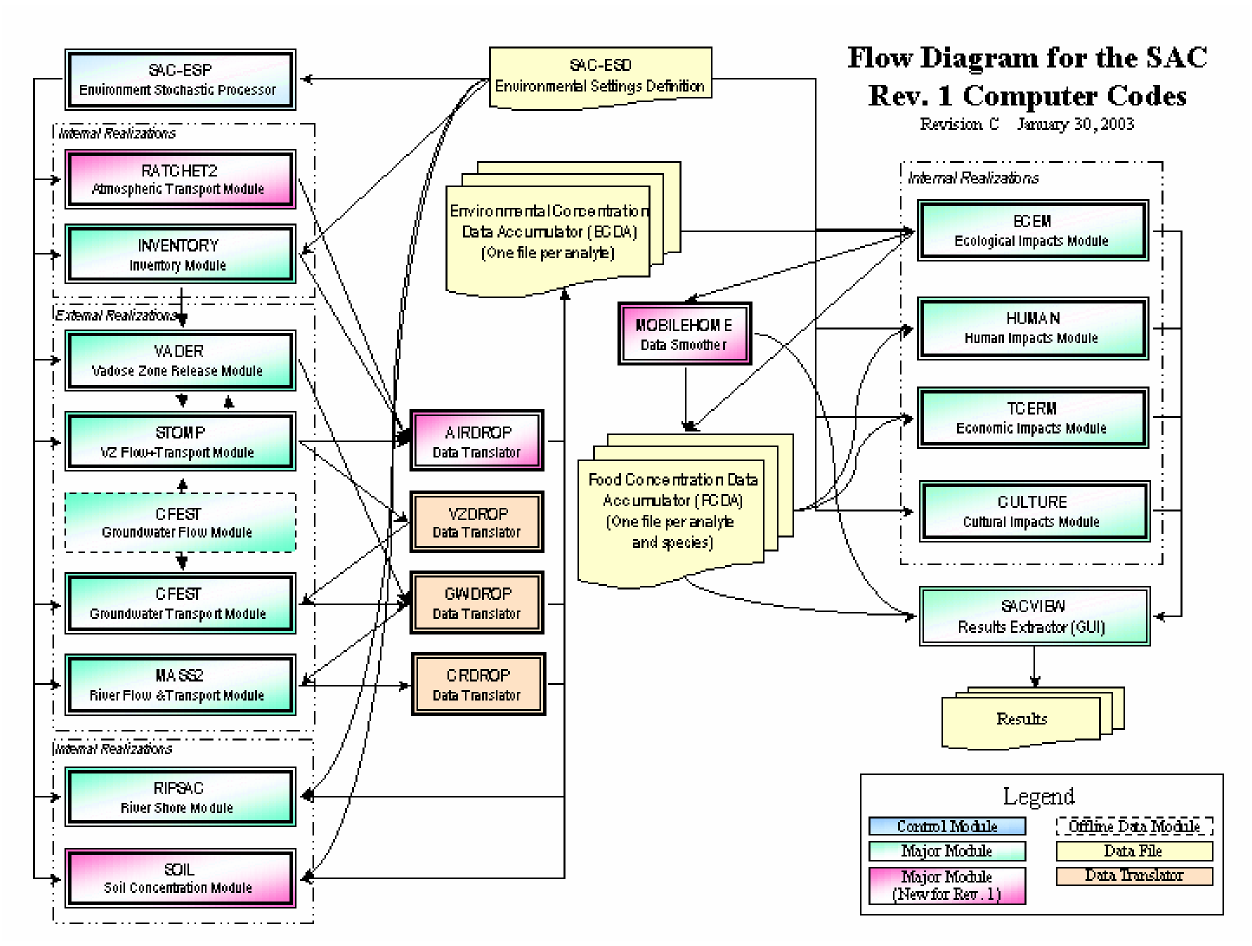


Figure 1.1 Module Information Flow for the SAC Rev. 1 Systems Code

1.2 Purpose of This Document

This document contains detailed user instructions for a suite of utility codes developed for Rev. 1 of the SAC. The suite of computer codes for Rev. 1 of SAC performs many functions. User instructions for the main transport and impacts codes are provided in Eslinger et al. (2004).

1.3 Overview of the Utility Codes

An overview of the utility codes described in this document is provided in Table 1.1. The utility codes are grouped by functional area shown in Figure 1.1. Some of the utility codes, such as INPROC, form an integral part of the inventory, release, transport, and impact generation functions. Other utility codes, such as VZGRAB, are used to summarize and extract data generated by the primary codes. Some codes such as MAKEU or CSCALE efficiently create data that could have been generated by the main sequence of codes.

Table 1.1 Overview of Utility Codes

Name	**Purpose**
Inventory-Related Functions	
HTWOS-TDP	This program reformats and decay corrects HTWOS (Hanford Tank Waste Operations Simulator) data so it can be imported into the inventory database.
HTWOS-TDLP	This program extracts and reformats tank leak data from the HTWOS (Hanford Tank Waste Operations Simulator) data so it can be imported into the inventory database.
INGRAB	This program summarizes the analyte inventory and waste volumes in the inventory data generated by the INVENTORY code. Data extracted (from the inventory.all file) may be annual or accumulated annual values and radioactive analyte totals are decay corrected.
INGRES	This program summarizes the analyte inventory and waste volumes in the inventory data generated by the INVENTORY code. Data extracted (from a single inv*.res file) may be annual or accumulated annual values and radioactive analyte totals are decay corrected.
INPROC	This program reads output from the inventory database, applies statistical and data fill-in rules, and generates a stochastic disposal action file for use in the INVENTORY code. It also generates WASTEMAP keywords for the INVENTORY code.
Inventory Database	This Microsoft Access-based database compiles inventory data from all sources. This database provides inputs for the INPROC code to start the inventory processing for a simulation case.
INVPLOT	This program generates data files of inventory on an annual time step for user-specified combinations of analytes, waste sites, and waste locations that be used in a plotting package.
INVSUM	This program generates summary information for inventory data. It can be used to identify sites for which no inventory was generated.
MAKEU	This program reads a single inventory results (inv*.res) file and writes a modified inv*.res file that contains the element uranium computed as the sum of all the uranium isotope values.
SIMS	This program reformats and decay corrects SIM (Soil Inventory Model) data so they can be imported into the inventory database.
Release and Vadose Zone-Related Functions	
GWGRAB	This program checks on the status of groundwater transport (CFEST) runs and also can extract summary results from CFEST runs once they have completed.
RLGRAB	This program consolidates waste release profiles generated by VADER for a given scenario as specified by one or more Sites, a specific analyte, and a set of waste release model parameters, over N realizations
VZGRAB	This program can extracts and consolidates data for several variables used in the vadose zone flow and transport calculations.
VZSWAP	This program provides a means to swap third party vadose zones release results into or out of the SAC system.

Name	**Purpose**
Saturated Zone-Related Functions	
River-Related Functions	
BACK_MASS2	This program generates files containing stochastic representation of background water and sediment concentrations that will be used in the river transport code MASS2.
CRGRAB	This program allows the user to collect and summarize the analyte mass or activity to the Columbia River as prepared for use by MASS2.
Median Value Generation-Related Functions	
Imp_Med	These programs convert a stochastic keyword file into a median-values keyword file. This process applies to keyword files for the ECEM, HUMAN, and TCERM codes.
Inv_Med	These programs convert a stochastic disposal action file into a median-values disposal action file.
Sto_Med	These programs convert a stochastic keyword file containing data for STOMP (actually used in the ESPcode) into a median-values keyword file.
STOCHASTIC	This program can generate sample data sets and summary statistics for any stochastic variable.
ECDA File-Related Functions	
CSCALE	This program reads one or more environmental concentration data accumulator (ECDA) files and writes another file that is a linear combination of the input files. The linear combination weights can be a function of time. This program can be used to build a file of decay product concentrations given a parent file.
ECDA	This program creates the binary environmental concentration data accumulator (ECDA) files and initializes the contents for each medium.
ECDA_ASCII	This program converts a user specific number of records in an ECDA file from binary storage mode to ASCII storage mode to allow visual inspection.
FCDA_ASCII	This program converts a user specific number of records in an FCDA (food concentration data) file from binary storage mode to ASCII storage mode to allow visual inspection.
FILLECDA	This program allows user-specified values of concentration data to be inserted into an existing environmental concentration data accumulator file.
HIGHMEDIA	This program allows extraction of the largest media value in an ECDA file for a suite of user specified locations and years.
Impacts-Related Functions	
CONSUME	This program reads a predation matrix and converts in into CONSUME keywords for use in the ecological impacts model (ECEM).
HIGHDOSE	This program generates a summary table of the ecological species with the highest impacts for a suite of user-specified solutions.
HIGHIMPACT	This program extracts the highest impact (from a suite of possible impacts) for each year for a user specific set of impact locations.

1.4 General Rules for Reading Keyword Descriptions

Many of the programs in this document use control files that contain keywords. A description of the general syntax for keywords is provided in Section 1.0. For simplicity of interpretation, the following additional rules apply to the keyword descriptions:

- Data that are required are enclosed in square brackets. For AB to be required, it would be denoted by [AB].
- If only one of the three items AB, BC, CD were required, it would be written as [AB|BC|CD]. The vertical bars indicate that the user must select one of the items in the list.
- Optional items are enclosed in normal brackets. For DE to be an optional entry, it would be denoted by {DE}.
- The { } or [ ] symbols indicate whether the data are required; the symbols do not need to be entered when the keyword is constructed.
- In some instances, numerical values or quote strings are associated with a modifier. This association is indicated by using the equal (=) symbol. The = symbol is not required but may be used when the keyword is constructed. When a numerical value or quote string is associated with a modifier, it must be physically entered on the input line directly after the modifier. For example:

```
FILE C_ECDA   ANALYTE="U"   NAME="/home/CPO/ecda/U_CPO.dat"    CREATE
```

2.0 Back_MASS2 – Columbia River Background Data Generation

2.1 Overview

Background concentrations for the river are calculated differently for SAC Rev. 1 than they were for SAC Rev. 0. A general equation for calculating background concentrations in surface water applies to all analytes irrespective of whether they have any significant contribution from fallout. For simplicity sake, contributions from fallout are considered only in 1990 or later years. The following governing equation for the background concentration in surface water is used for all years before 1990:

$$C_R = C_B$$

The following equation is used for 1990 or later years (equation and data provided by Bruce Napier):

$$C_R = C_B + M_B e^{(-\lambda_b [\text{Year}-1990])}$$

where

C_R = Concentration of the analyte in surface water (Ci/m^3 or kg/m^3)
C_B = Nominal background concentration (Ci/m^3 or kg/m^3) from natural sources
M_B = Multiplier (unitless) on the washout term for analytes that have a fallout contribution
λ_B = Decay term (1/yr) that includes the effect of radioactive decay and leaching from the land surface into surface waters

The values for C_B, M_B, and λ_B have different values for every analyte. The variables C_B and M_B are defined as stochastic variables. The variable for λ_B is defined as a constant. The values for C_B and M_B can be set to a constant, including the constant zero. The value for λ_B can also be set to zero.

The nominal values in the governing concentration equation for the analytes used in the Composite Analysis are provided in Table 2.1. The original data provided by Bruce Napier had concentration units of pCi/L. The data in Table 2.1 have been modified to have units of Ci/m^3. The modification is performed by multiplying the original data by 10^{-9}.

Table 2.1 Nominal Coefficient Values for Background Concentrations in the Columbia River

Analyte ID	C_B	M_B	λ_B	Fallout?
C14	5.3×10^{-11}	0	0	No
Cl36	0	0	0	No
Cs137	0	1.07×10^{-11}	0.023	Yes
Eu152	0	0	0	Very small
H3[(a)]	3.9×10^{-8}	3.04×10^{-8}	0.0562	Yes
I129	0	1.28×10^{-14}	4.41×10^{-8}	Very small
Np237	0	0	0	Very small
Pa231	Not modeled	Not modeled	Not modeled	No
Ra226	Not modeled	Not modeled	Not modeled	No
Se79	0	0	0	Very small
Sr90	0	1.01×10^{-10}	0.0241	Yes
Tc99	0	4.41×10^{-11}	3.28×10^{-6}	Yes

Analyte ID	C_B	M_B	λ_B	Fallout?
U233	0	0	0	No
U235[(b)]	7.9×10^{-12}	0	0	No
U238	1.9×10^{-10}	0	0	No

(a) Recent H3 samples in the Columbia River have about 39 pCi/L rather than the 25 in the general reference.
(b) The U235 value is based on the U238 value and 0.648% relative weight natural abundance

The stochastic distributions associated with the nonzero coefficients defined in Table 2.1 are defined using the following rules:

- **Variable C_B**. The triangular distribution will be used for all values of C_B. The distribution will be symmetric about the midpoint, and the half-range will be 50% of the mid-point. The variable tag will be CB.
- **Variable M_B**. The triangular distribution will be used for all values of C_B. The distribution will be symmetric about the midpoint, and the half-range will be 50% of the mid-point. The variable tag will be MB.
- **Variable λ_B**. This variable is always defined as a constant. For ease of input, a stochastic variable definition will be used. The variable tag will be LB.

2.1.1 Location in the Processing Sequence

The Back_MASS2 code is a preprocessor to the river code MASS2. The system controller (ESP) reads a file containing stochastic information (nominally named “biota.stoch”) and the ESD keyword file and writes a series of “biota.key” files (one for each analyte by realization combination) that are processed one at a time by the Back_MASS2 code. The Back_MASS2 code writes a series of input files that provide background data (for the specific analyte and realization combination) for MASS2. Excerpts from a biota.stoch file are provided in Table 2.2. This file is not read by Back_MASS2; however, it provides the basis for the files prepared for Back_MASS2 by the ESP code.

Table 2.2 Excerpts from a biota.stoch file

```
!  Stochastic Keyword file for the MASS2 & Back_MASS2 codes
!  Purpose:
!    This keyword file is for use in the ESP program.  The ESP reads this
!    file and the ESD keyword file and writes a series of "biota.key' files
!    for use by the Back_MASS2 code.  The Back_MASS2 code writes a series of
!    input files that provide stochastic background data for MASS2.
!    Background from fallout is modeled from 1990 forward
TITLE "Stochastic Keyword Data for MASS2 and Back_MASS2 for the CA Analysis"
SEED 4436546
DEBUG STOCHASTIC="stoch/biota_stoch.out"
!
! River IDs and names for use in the MASS2 background data
RIVER ID="PRD"  NAME="Columbia River at Priest Rapids Dam"
! Suspended sediment loading (kg/m^3) in each river
STOCHASTIC BACKGROUND SOLUTION="sediment" RIVER="PRD"    1 0.00375
!---------------| Median Value Distributions |
! Dissolved contaminant concentrations by analyte and river
! Units are Ci/m^3 for radionuclides and kg/m^3 for chemicals
STOCHASTIC BACKGROUND ANALYTE="C14"   SOLUTION="CB" RIVER="PRD"
```

```
  1 5.300E-11 "Natural background in Ci/m^3"
STOCHASTIC BACKGROUND ANALYTE="C14"   SOLUTION="MB" RIVER="PRD"
  1 0  "Multiplier on leachout term (unitless)"
STOCHASTIC BACKGROUND ANALYTE="C14"   SOLUTION="LB" RIVER="PRD"
  1 0  "Leachout removal constant (1/yr)"
!
STOCHASTIC BACKGROUND ANALYTE="Cs137" SOLUTION="CB" RIVER="PRD"
  1 0  "Natural background in Ci/m^3"
STOCHASTIC BACKGROUND ANALYTE="Cs137" SOLUTION="MB" RIVER="PRD"
  1 1.070E-11 "Multiplier on leachout term (unitless)"
STOCHASTIC BACKGROUND ANALYTE="Cs137" SOLUTION="LB" RIVER="PRD"
  1 2.300E-02 "Leachout removal constant (1/yr)"
!
END
```

2.1.2 How the Code Is Invoked

Back_MASS2 can run under either the Windows or the Linux operating systems. Under the Windows operating system (2000, or XP), Back_MASS2 executes in a DOS box. A run of Back_MASS2 is initiated by entering the following command line:

```
   back_mass2 “Keyfilename”
```

Under the Linux operating system CSCALE is executed through any of the following Bourne Shell or C Shell commands:

```
   back_mass2-1.exe “Keyfilename”
```

For these commands, Back_MASS2.EXE or back_mass2-1.exe is the name of the executable program and “Keyfilename” is the name of an existing control keyword file. Both the name of the executable program and the keyword file may contain path information. If Back_MASS2 is invoked without entering the name of the keyword file, then the code will prompt the user for the file name. If Back_MASS2 cannot find or open the keyword file, then the code will terminate execution after writing an error message to the standard output device.

2.1.3 Memory Requirements

The Back_MASS2 code uses dynamic memory allocation, so the memory requirements depend on the problem being analyzed. However, the code uses minimal memory because operates only on a small amount of scalar data.

2.2 File Definitions

The Back_MASS2 code reads one file. It writes a report file and three additional files for every river defined in the keyword file.

2.2.1 Input Files

The only input file for the Back_MASS2 code is a control keyword file. An example keyword file Back_MASS2 for carbon-14 is provided in Table 2.3. Descriptions for the keywords are provided in Section 2.3.

Table 2.3 Example Keyword File for Back_MASS2

```
$ ESP TEMPLATE File  : MASS2/Biota.Stoch
$ ESP Date Generated : 06-28-2004
$ ESP Program Name   : ESP-Unix
$ ESP Version Number : 1.2
$ ESP Program Date   : 02 Feb 2004
$ Current Run ID     : 20040628083102
TITLE "2004 Composite Analysis 10,000-year (Median Inputs) Assessment"
USER "Eslinger-DWE-WEN"
! Realization
REALIZATION CURRENT=1  TOTAL=1
! Analyte
ANALYTE ID="C14" TYPE="NR" NAME="Carbon-14" HALFLIFE=5715.000
! Time period for the analysis
PERIOD START=1944 STOP=12060 CLOSURE=2070
! River IDs and Names
RIVER ID="PRD" NAME="Columbia River at Priest Rapids Dam"
! Dissolved -- Stochastic (#Analytes x #Rivers)
DISSOLVED ANALYTE="C14" RIVER="PRD" SOLUTION="CB" VALUE=5.300000E-11
DISSOLVED ANALYTE="C14" RIVER="PRD" SOLUTION="MB" VALUE=0.000000E+00
DISSOLVED ANALYTE="C14" RIVER="PRD" SOLUTION="LB" VALUE=0.000000E+00
! SEDIMENT -- Stochastic (#Rivers)
SEDIMENT RIVER="PRD" VALUE=3.750000E-03
END
```

2.2.2 Output Files

The Back_MASS2 code writes a report file and three additional files for every river defined in the keyword file. The report file contains summary information for the code run and is not described further here. Examples of the three additional output files written for use in MASS2 are provided. The files contain concentrations in the water column (Table 2.4), concentrations in particulate matter (Table 2.5), and sediment loading (Table 2.6). The first two lines in each of these tables is actually one line in the data file, but the line is so long that it wraps to the next line in the word processor. Many of the entries in (Table 2.4) were deleted to shorten the file for presentation.

Table 2.4 Example Dissolved Concentration File for MASS2

```
# Back_MASS2 2.10.A.0 : Incoming dissolved concentration
for Cs137 at Columbia River at Priest Rapids Dam
01-01-1943 00:00:00  0.00000E+00 /
01-01-1989 00:00:00  0.00000E+00 /
01-01-1990 00:00:00  1.07000E-11 /
01-01-1991 00:00:00  1.04567E-11 /
```

```
01-01-1992 00:00:00  1.02189E-11 /
01-01-1993 00:00:00  9.98659E-12 /
…
01-01-5413 00:00:00  0.00000E+00 /
01-01-5414 00:00:00  0.00000E+00 /
…
01-01-12061 00:00:00  0.00000E+00 /
```

Table 2.5 Example Particulate Concentration File for MASS2

```
# Back_MASS2 2.10.A.0 : Incoming particulate concentration for
Cs137 at Columbia River at Priest Rapids Dam
01-01-1943 00:00:00 0.0 /
01-01-12061 00:00:00 0.0 /
```

Table 2.6 Example Sediment Loading File for MASS2

```
# Back_MASS2 2.10.A.0 : Incoming Sediment (kg/m^3) for
Columbia River at Priest Rapids Dam
01-01-1943 00:00:00  3.75000E-03 /
01-01-12061 00:00:00  3.75000E-03 /
```

2.3 Keyword Definitions for Back_MASS2

2.3.1 ANALYTE Keyword for Back_MASS2

The ANALYTE keyword identifies an analyte for use in the MASS2 code. Only one ANALYTE keyword is allowed for a run of the Back_MASS2 code. The following is this keyword's syntax:

```
   ANALYTE [ID=“quote 1”] [TYPE=“quote 2”] [NAME=“quote 3”] {HALFLIFE=N1}
```

The modifiers associated with the ANALYTE keyword are defined in Table 2.7.

Table 2.7 Modifiers Associated with the ANALYTE Keyword in Back_MASS2

Modifier	Description
ID	The quote string associated with the ID modifier is an analyte identification string up to six characters in length. The analyte identification string is case sensitive, and spaces or hyphens change the definition. The analyte ID must match one of the analytes defined in the ESD keyword file.

Modifier	Description
TYPE	The quote string associated with the TYPE modifier string is a two-character analyte type indicator. The following are the valid entries for this string: • NR – if the analyte is a radioactive element or an inorganic compound containing a radionuclide • NS – if the analyte is a stable (nonradioactive) element or inorganic compound • OR – if the analyte is an organic compound containing a radionuclide • OS – if the analyte is an organic compound, containing a stable (nonradioactive) elemental analyte or compound
NAME	The quote string associated with the NAME modifier is an analyte name or description up to 72 characters in length.
HALFLIFE	The numerical entry associated with the HALFLIFE modifier is the half-life of the analyte. This value has units of years. Entry of this modifier is necessary when defining a radioactive analyte but should be omitted for nonradioactive analytes.

The following ANALYTE keyword defines the analyte carbon-14 (not organic, radioactive) with a half-life of 5715 years.

```
   ANALYTE ID="C14" TYPE="NR" NAME="Carbon-14" HALFLIFE=5715.000
```

2.3.2 DISSOLVED Keyword for Back_MASS2

The DISSOLVED keyword identifies the background concentrations for an analyte in the water column (at the upstream boundary) in a river used in the MASS2 code. The dissolved concentrations for multiple rivers are defined by making multiple entries of the DISSOLVED keyword. The following is this keyword’s syntax:

```
   DISSOLVED [ANALYTE=“Quote1”] [RIVER=“Quote2”]
      [SOLUTION=“Quote3”] [VALUE=N1]
```

The ID for the analyte is defined in a quote string associated with the modifier ANALYTE and is limited to six characters in length (see the ANALYTE keyword in Section 2.3.1). The ID for the river is defined in a quote string associated with the modifier ID and is limited to six characters in length (see the RIVER keyword in Section 2.3.6). The quote string associated with the modifier SOLUTION takes on one of three values: CB, MB, or LB. The definitions of these parameters for the background solution are defined in Section 2.1. The following three example DISSOLVED keywords define the background concentration for C14 in the Columbia River at Priest Rapids Dam.

```
   DISSOLVED ANALYTE="C14" RIVER="PRD" SOLUTION="CB" VALUE=5.300000E-11
   DISSOLVED ANALYTE="C14" RIVER="PRD" SOLUTION="MB" VALUE=0.000000E+00
   DISSOLVED ANALYTE="C14" RIVER="PRD" SOLUTION="LB" VALUE=0.000000E+00
```

Concentrations may be defined for more than one river in a single run of the Back_MASS2 code. A suite of three DISSOLVED keywords are required to define the background concentration for each river.

2.3.3 END Keyword for Back_MASS2

The END keyword signifies the end of all keyword data. Nominally it will be the last keyword in the keyword file. All data in the keyword file after the END keyword will be ignored. The following is this keyword's syntax:

```
END
```

2.3.4 PERIOD Keyword for Back_MASS2

The PERIOD keyword identifies the start and stop times for the entire simulation. The following is this keyword's syntax:

```
PERIOD [START=N1]  [STOP=N2]
```

The modifier START and the associated value N1 identify the year for the start of the simulation period. The modifier STOP and the associated value N2 identify the year for the end of the simulation period. Start and stop years should be entered as whole numbers with the stop year no smaller than the start year. The following is an example PERIOD keyword that simulates from 1944 through 3050:

```
PERIOD START=1944 STOP=3050
```

2.3.5 REALIZATION Keyword for Back_MASS2

The REALIZAT keyword identifies the current realization number and the total number of realizations to be simulated. The following is this keyword's syntax:

```
REALIZATION [CURRENT=N1] [TOTAL=N2]
```

The number of the current realization is identified by the modifier CURRENT. The total number of realizations for the analysis definition is identified by the modifier TOTAL. The following is an example REALIZAT keyword requesting Back_MASS2 to process data for realization 23 out of 100 realizations:

```
REALIZAT CURRENT=23 TOTAL=100
```

2.3.6 RIVER Keyword for Back_MASS2

The RIVER keyword identifies the ID and name of a river used in the MASS2 code. Multiple rivers are defined by making multiple entries of the RIVER keyword. The following is this keyword's syntax:

```
RIVER [ID=”quote 1”] [NAME=”quote 2”]
```

The ID for the river is defined in a quote string associated with the modifier ID and is limited to six characters in length. The descriptive name of the river is defined in the quote string associated with the modifier NAME and is limited to 48 characters in length. The following is an example RIVER keyword defining the Columbia River at Priest Rapids Dam.

```
   RIVER ID="PRD" NAME="Columbia River at Priest Rapids Dam"
```

2.3.7 SEDIMENT Keyword for Back_MASS2

The SEDIMENT keyword identifies the suspended sediment loading for water (at the upstream boundary) in a river used in the MASS2 code. The sediment loading for multiple rivers are defined by making multiple entries of the SEDIMENT keyword. The following is this keyword's syntax:

```
   SEDIMENT [RIVER=”quote”] [VALUE=N1]
```

The ID for the river is defined in a quote string associated with the modifier ID and is limited to six characters in length (see the RIVER keyword in Section 2.3.6). The suspended sediment loading (in kg/m^3) in the river is defined in the numerical value associated with the modifier VALUE. The following is an example SEDIMENT keyword defining the suspended sediment loading in the Columbia River at Priest Rapids Dam.

```
   SEDIMENT RIVER="PRD" VALUE=3.750000E-03
```

2.3.8 TITLE Keyword for Back_MASS2

The TITLE keyword is used to define a single-line problem title. The problem title will be written to output files. If the title is not supplied, then the program will error terminate. The following is this keyword's syntax:

```
   TITLE [“quote”]
```

The title is entered in a quote string, which must be enclosed in double quotation marks. Titles up to 200 characters long are supported. The following example defines a title for a run of the code:

```
   TITLE “Example title line for the human impacts code.”
```

2.3.9 USER Keyword for Back_MASS2

The USER keyword is used to identify the user of the program. The user name will be written to output files. If the user name is not supplied, then the program will error terminate. The following is this keyword's syntax:

```
   USER [“quote”]
```

The user name is entered in a quote string, which must be enclosed in double quotation marks. User names up to 16 characters long are supported. The following example defines John Q. Public as the user running the code:

```
   USER “John Q. Public”
```

3.0 CONSUME – Ecological Species Consumption Data

3.1 Overview

The CONSUME program reads a text file containing ecological species information, including a full predation matrix, and reformats the data into CONSUME keyword records for use in ECEM (see Eslinger et al. [2004b], Volume 2, Section 3.0). The predation matrix (the fraction of the intake for an ecological specie that comes from consuming other species) is also checked for errors during this process. Typically, the underlying data for ECEM are maintained in a spreadsheet. This program converts data from the spreadsheet into a format that can be inserted into an ECEM keyword file.

3.1.1 Location in the Processing Sequence

The CONSUME program is a preprocessor to the ECEM code.

3.1.2 Memory Requirements

The CONSUME code uses dynamic memory allocation. It is expected that most, if not all, of the runs of the CONSUME code will require less than 2 MB of memory.

3.2 File Definitions

The CONSUME code reads one file and writes one file.

3.2.1 Input Files

The CONSUME code reads a text data file containing ecological species information in a comma separated format. An example input data file for 12 ecological species is provided in Table 3.1. The first line in the file is an integer (N_S) giving the number of species to process. This input is followed a definition line for each of the N_S species. Each line contains the following:

- The species ID (up to 6 characters)
- The type of the species; valid entries for this string are:
  - TA – if the species is a terrestrial animal
  - TP – if the species is a terrestrial plant
  - QA – if the species is an aquatic animal
  - QP – if the species is an aquatic plant
- The species location; valid entries for this string are:
  - RIPARIAN – riparian zone (river shore)
  - UPLAND – upland
  - AQUATIC – river
- The species long name (up to 72 characters)

The definition lines are followed by consumption lines for each of the N_S species. The order of the consumption lines must match with the order the species are defined in the file. Each line contains

N_S+2entries. The first N_S entries give the fraction of the intake for the species that comes from each of the other species. Valid entries are in the range 0 to 1. Plants always have all zeros, and animal species typically have entries that sum to 1. An exception to this rule is the adult salmon that moves up the river without consuming any food. The entry in the N_S+1 position is the amount of sediment consumed by the species, expressed as a fraction of intake. This value should be zero for all species that are not aquatic animals. Valid values are in the range 0 to 1. The entry in the N_S+2 position is the amount of soil consumed by the species, expressed as a fraction of intake. This value should be zero for all species that are not terrestrial animals. Valid values are in the range 0 to 1.

Table 3.1 Example Input File for CONSUME

```
12
"RCTWOD","TP","RIPARIAN","black cottonwood"
"RCRESS","TP","RIPARIAN","Columbia yellowcress"
"RSEDGE","TP","RIPARIAN","dense sedge"
"RFERN ","TP","RIPARIAN","fern"
"RFUNGI","TP","RIPARIAN","fungi"
"HRVMSE","TA","RIPARIAN","Harvest mouse"
"RMLBRY","TP","RIPARIAN","mulberry"
"RREEDC","TP","RIPARIAN","reed canarygrass"
"RRUSHS","TP","RIPARIAN","rushes"
"RSWCLV","TP","RIPARIAN","Sweet Clover (Melalotus alba)"
"RARTPD","TA","RIPARIAN","terrestrial arthropods"
"RRTULE","TP","RIPARIAN","tule"
0,0,0,0,0,0,0,0,0,0,0,0,0,0
0,0,0,0,0,0,0,0,0,0,0,0,0,0
0,0,0,0,0,0,0,0,0,0,0,0,0,0
0,0,0,0,0,0,0,0,0,0,0,0,0,0
0,0,0,0,0,0,0,0,0,0,0,0,0,0
0,0,0.3,0,0,0,0,0.3,0,0.1,0.3,0,0,0.02
0,0,0,0,0,0,0,0,0,0,0,0,0,0
0,0,0,0,0,0,0,0,0,0,0,0,0,0
0,0,0,0,0,0,0,0,0,0,0,0,0,0
0,0,0,0,0,0,0,0,0,0,0,0,0,0
0.1,0.1,0.1,0.1,0.1,0,0.2,0.1,0.1,0,0,0.1,0,0.5
0,0,0,0,0,0,0,0,0,0,0,0,0,0
```

3.2.2 Output Files

The CONSUME code writes one data file. This file is a text file that is always named “consume.rpt” that contains the reformatted consumption data, an echo of the predation matrix, and any error messages. Excerpts from a report file are provided in Table 3.2. The typical mode of operation is to extract the CONSUME keyword records from this file and paste them into an input file for the ECEM (see Eslinger et al. [2004b], Volume 2, Section 3.0) code.

Table 3.2 Excerpts from a CONSUME Report File

```
! Consumption formatted as CONSUME keywords for ECEM
!   Program: Consume
!   Version: 1.0.003
!   Revised: 23 Aug 2004
!   User:    Anonymous User
!   Date:    08-23-2004
!   Time:    16:38:51
!   File:    small.dat
!
CONSUME ID="HRVMSE" PREY
  "RSEDGE"  0.30000
  "RREEDC"  0.30000
  "RSWCLV"  0.10000
  "RARTPD"  0.30000
  SOILING   0.02000
!
CONSUME ID="RARTPD" PREY
  "RCTWOD"  0.10000
  "RCRESS"  0.10000
  "RSEDGE"  0.10000
  "RFERN "  0.10000
  "RFUNGI"  0.10000
  "RMLBRY"  0.20000
  "RREEDC"  0.10000
  "RRUSHS"  0.10000
  "RRTULE"  0.10000
  SOILING   0.50000
```

3.3 Code Execution

CONSUME can run under either the Windows or Linux operating system. Under the Windows operating system (Releases 98, NT, 2000, or XP), CONSUME executes in a DOS box. A run of CONSUME is initiated by entering the following command line:

```
CONSUME “Data File”
```

Under the Linux operating system CONSUME is executed through any of the following Bourne Shell or C Shell commands:

```
Consume-1.exe “Data File”
```

For these commands, CONSUME.EXE or consume-1.exe is the name of the executable program. One argument (file name) must be provided on the command line. The argument is the name of the data file to be processed.

4.0 CRGRAB – Columbia River Data Grabber

4.1 Overview

The CRGRAB (Columbia River Data Grabber) data extractor allows the user to collect and summarize the analyte mass or activity inputs to the Columbia River.

4.1.1 Location in the Processing Sequence

The CRGRAB program reads data files written by the GWDROP code. These are the same data files that the MASS2 code uses to input the analyte source term for transport in the river model. Data for more than one analyte or realization can be gathered in the same run of CRGRAB.

4.1.2 How the Code Is Invoked

CRGRAB only runs under the Linux operating system. CRGRAB is executed through any of the following Bourne Shell or C Shell commands:

```
   crgrab-1.exe “Keyfilename”
```

For these commands, crgrab-1.exe is the name of the executable program and “Keyfilename” is the name of the environmental settings definition (ESD) keyword file. Both the name of the executable program and the ESD keyword file may contain path information. If CRGRAB is invoked without entering the name of the keyword file, then the code will prompt the user for the file name. If CRGRAB cannot open the ESDD keyword file, then the code will terminate execution after writing an error message to the standard output device.

4.1.3 Memory Requirements

The CRGRAB code uses dynamic memory allocation, so the memory requirements depend on the problem being analyzed. It is expected that most, if not all, of the runs of the CRGRAB code will require less than 32 MB of memory.

4.2 File Definitions

The CRGRAB code reads two keyword files, a small number of MASS2 grid coordinate files, and potentially a large number of files containing inputs for MASS2. The code writes two or three output files, depending on the case definition.

4.2.1 Input Files

The CRGRAB code reads the ESD keyword file and a CRGRAB keyword file. In addition, CRGRAB reads a number of MASS2 grid coordinate files, and potentially a large number of files containing inputs for MASS2. The user only modifies the CRGRAB keyword file, the other files are read only for supporting information.

4.2.1.1 CRGRAB Keyword File

The CRGRAB keyword file is always named crgrab.key and contains control information for extracting data. An example file is provided in Table 4.1. Detailed definitions of the keywords are provided in Section 4.3. The actual run of the code uses grid files with extensions ".000" through ".056". Some of the file names were deleted for presentation purposes.

Table 4.1 Example Keyword File for the CRGRAB Code

```
!  Test keyword file for the crgrab data extractor
TITLE "Test the crgrab code"
USER "Paul W. Eslinger"
ANALYTE LIST "H3" "I129" "CCl4"
EXTRACT INFLUX BACKGROUND
LOCATION
GRABPATH "/home/ANALYSIS4/CA1_median/crgrab/"
REALIZATION LIST 1
IGNORE 4000
TIME ANNUAL TOTAL
!  MASS2 Grid Coordinate Files
MASS2GRID
  "/home/ANALYSIS4/CA1_median/mass2/hanfnad83m-pt.000"
  "/home/ANALYSIS4/CA1_median/mass2/hanfnad83m-pt.001"
  "/home/ANALYSIS4/CA1_median/mass2/hanfnad83m-pt.002"
 ...
  "/home/ANALYSIS4/CA1_median/mass2/hanfnad83m-pt.054"
  "/home/ANALYSIS4/CA1_median/mass2/hanfnad83m-pt.055"
  "/home/ANALYSIS4/CA1_median/mass2/hanfnad83m-pt.056"
END
```

4.2.1.2 Other Input Files

CRGRAB reads the ESD keyword file to obtain analyte information and analysis path information. The format of this file is described Section 2.0 of Eslinger et al. (2004a), so it is not discussed in any further detail here. CRGRAB also reads a number of MASS2 grid coordinate files. The names of these files are provided in the CRGRAB keyword file.

In addition, CRGRAB reads potentially a large number of files containing mass flux inputs for MASS2. For example, data for one realization of Cs137 in the 2004 Composite Analysis is contained in approximately 1450 data files. The general format of these files is the following:

- Line 1: Header line
- Line 2: time, cumulative value
- Line 3: time, cumulative value
- …
- Line n: time, cumulative value

The time format is mm-ddd-yyyy hh:mm:ss, where the year can be four, five or six digits long, if necessary. The cumulative value has units of kg if the file contains releases for a nonradioactive analyte, units of curies if the file contains releases for a radioactive analyte, and units of cubic meters if the file contains water volumes.

The naming convention for the river release files is built into CRGRAB. A typical file name associated with an analyte (Cs137 for example, without path name) is TMS-14219-Cs137.DAT. In general, the file name is built from the string "TMS-" followed by a CFEST node number ID, followed by a dash and the analyte ID, followed by the extension ".DAT".

4.2.2 Output Files

The CRGRAB code writes two to four three output files, depending on the case definition. The report file is named crgrab.rpt and is written for every run of the code. It contains summary information from the code run and is not described in further detail.

The annual release file contains annual and cumulative release to the river by year for every requested analyte and is always named "Influx_Annual.csv". Excerpts from this file for the three analytes Tc99, H3 and U238 are provided in Table 4.2. The file is written in comma separated format to facilitate import into a spreadsheet or graphics program. The first line of the file is shown as wrapped in the table but actually occupies only one line of the file. The cumulative release up to the year 2000 is provided in a separate file, always named "Influx_Annual_Locs.csv", as shown in Table 4.3. In addition, the total release to the river by releasing location can be output under control of the LOCATION keyword in the file named "Influx_Total_Locs.csv". Excerpts from an example output file of total releases by location are provided in Table 4.4.

Table 4.2 Annual River Release File fom CRGRAB

```
"Realization","Year","Tc99 Annual","Tc99 Cumul.","H3 Annual","H3 Cumul.","U238
Annual","U238 Cumul."
1,1944, 9.18482E-11, 9.18482E-11, 1.18844E-09, 1.18844E-09, 1.12149E-18, 1.12149E-18
1,1945, 3.88735E-03, 3.88735E-03, 1.83568E-01, 1.83568E-01, 1.37958E-05, 1.37958E-05
1,1946, 1.98659E-02, 2.37533E-02, 9.15834E-01, 1.09940E+00, 5.14122E-03, 5.15501E-03
1,1947, 2.77167E-02, 5.14700E-02, 1.21544E+00, 2.31484E+00, 4.36316E-02, 4.87866E-02
1,1948, 3.26185E-02, 8.40885E-02, 1.35563E+00, 3.67048E+00, 7.57750E-02, 1.24562E-01
1,1949, 3.05275E-02, 1.14616E-01, 1.20208E+00, 4.87255E+00, 8.16468E-02, 2.06208E-01
1,1950, 3.02840E-02, 1.44900E-01, 1.21135E+00, 6.08390E+00, 1.08975E-02, 2.17106E-01
1,1951, 3.25044E-02, 1.77404E-01, 1.53951E+00, 7.62341E+00, 1.08975E-02, 2.28003E-01
1,1952, 3.50948E-02, 2.12499E-01, 2.03253E+00, 9.65593E+00, 1.09275E-02, 2.38931E-01
1,1953, 3.62911E-02, 2.48790E-01, 6.50826E+00, 1.61642E+01, 1.08975E-02, 2.49828E-01
1,1954, 3.87063E-02, 2.87496E-01, 6.79331E+00, 2.29575E+01, 1.09373E-02, 2.60766E-01
1,1955, 4.08380E-02, 3.28334E-01, 7.33052E+00, 3.02880E+01, 2.53792E-02, 2.86145E-01
1,1956, 4.03984E-02, 3.68733E-01, 1.00113E+01, 4.02994E+01, 2.54488E-02, 3.11594E-01
1,1957, 3.71861E-02, 4.05919E-01, 1.15777E+01, 5.18770E+01, 2.53792E-02, 3.36973E-01
1,1958, 2.34013E-02, 4.29320E-01, 1.26491E+01, 6.45261E+01, 2.53794E-02, 3.62352E-01
1,1959, 1.33992E-02, 4.42719E-01, 1.44867E+01, 7.90128E+01, 2.53793E-02, 3.87732E-01
...
1,1997, 6.01274E-01, 2.54193E+01, 1.76168E+02, 2.89322E+04, 3.25530E-01, 3.39689E+00
1,1998, 2.72972E-01, 2.56923E+01, 1.39264E+02, 2.90715E+04, 3.25528E-01, 3.72242E+00
1,1999, 5.13283E-01, 2.62056E+01, 1.66100E+02, 2.92376E+04, 3.25531E-01, 4.04795E+00
1,2000, 5.34088E-01, 2.67397E+01, 1.69791E+02, 2.94074E+04, 1.44196E-01, 4.19215E+00
```

Table 4.3 Cumulative Release File from CRGRAB

```
"Realization","Year","Tc99","H3","U238"
1,2000, 2.67397E+01, 2.94074E+04, 4.19215E+00
```

Table 4.4 Total Release by Releasing Location from CRGRAB

```
"Realization","Year","Easting","Northing","Tc99","H3","U238"
1,2000,    574615.4,    154506.0, 0.00000E+00, 0.00000E+00, 0.00000E+00
1,2000,    574576.0,    154543.9, 0.00000E+00, 0.00000E+00, 0.00000E+00
1,2000,    574536.7,    154581.7, 0.00000E+00, 0.00000E+00, 0.00000E+00
1,2000,    574497.3,    154619.5, 0.00000E+00, 0.00000E+00, 0.00000E+00
1,2000,    574457.9,    154657.4, 0.00000E+00, 0.00000E+00, 0.00000E+00
1,2000,    574418.6,    154695.2, 0.00000E+00, 0.00000E+00, 0.00000E+00
1,2000,    574379.3,    154733.1, 0.00000E+00, 0.00000E+00, 0.00000E+00
1,2000,    574339.9,    154770.9, 0.00000E+00, 0.00000E+00, 0.00000E+00
1,2000,    574710.1,    154562.7, 0.00000E+00, 0.00000E+00, 0.00000E+00
1,2000,    574686.3,    154605.6, 0.00000E+00, 0.00000E+00, 0.00000E+00
1,2000,    574659.7,    154645.9, 0.00000E+00, 0.00000E+00, 0.00000E+00
1,2000,    574629.6,    154683.8, 0.00000E+00, 0.00000E+00, 0.00000E+00
…
1,2000,    582534.5,    147463.5, 1.05315E-06, 1.08056E-04, 5.93517E-09
1,2000,    582371.4,    147186.1, 1.19300E-06, 1.16438E-04, 6.43102E-09
1,2000,    582447.3,    147193.6, 1.35335E-06, 1.32088E-04, 7.29537E-09
1,2000,    582516.3,    147201.0, 1.50082E-06, 1.46482E-04, 8.09034E-09
1,2000,    582500.8,    146901.5, 1.38131E-05, 1.47514E-03, 5.95506E-08
1,2000,    582537.4,    146916.5, 1.23623E-05, 1.31756E-03, 5.36062E-08
1,2000,    582573.6,    146931.1, 1.10899E-05, 1.17955E-03, 4.83707E-08
1,2000,    582632.1,    146611.0, 1.63086E-05, 2.08297E-03, 4.54770E-08
1,2000,    582662.5,    146641.1, 1.00104E-05, 1.23631E-03, 3.06861E-08
1,2000,    582691.9,    146670.5, 5.53580E-06, 6.26356E-04, 1.98411E-08
1,2000,    582766.8,    146433.2, 3.93529E-06, 4.57893E-04, 4.48185E-09
1,2000,    582796.3,    146445.6, 1.90956E-08, 7.61659E-07, 2.48644E-11
1,2000,    582824.9,    146456.9, 2.15329E-08, 8.58875E-07, 2.80380E-11
```

4.3 Keyword Definitions for CRGRAB

Most of the keywords for the CRGRAB code can be entered in any order. The following restrictions apply on keyword order:

- The REPORT keyword must be the first keyword in the file.
- The END keyword must be the last keyword in the file.

4.3.1 ANALYTE Keyword for CRGRAB

The ANALYTE keyword is used to define the analytes to be used in the simulation. The following is this keyword’s syntax:

```
ANALYTE [ALL | LIST=“quote 1” … “quote n”]
```

The analytes requested must be a subset of the analytes for which environmental data were computed and stored by the inventory, release, and transport modules. If the modifier ALL is present then data is extracted for every analyte defined in the ESD keyword file. If the LIST modifier is present then it must be followed by one or more quote strings that contain analyte ID's. The following example ANALYTE keyword specifies that data will be extracted for the three analytes H3, I129, and C14.

```
ANALYTE LIST "H3" "I129" "CCl4"
```

4.3.2 DEBUG Keyword for CRGRAB

The DEBUG keyword is used to activate dumping of intermediate calculations to the report file. It should be used sparingly and with only one or two realizations; otherwise, the report file size can exceed the maximum allowable size of 2.1 gigbytes. The following is this keyword's syntax:

```
DEBUG [modifier 1] {modifier 2} … {modifier 5}
```

Multiple DEBUG records can be entered with combinations of modifiers, or a single record can be entered containing all of the modifiers. The modifiers can be entered in any order. Table 4.5 describes the modifiers associated with the DEBUG keyword.

Table 4.5 Modifiers Associated with the DEBUG Keyword in CRGRAB

Modifier	Description
GRID	Intermediate outputs on grid-related data extractions and calculations.
INFLUX	Intermediate outputs on reading and processing influx data.
TIMES	Intermediate outputs on time-registering data.
TOTAL	Intermediate outputs on calculations for total releases.
LOCATE	Intermediate outputs on calculations for radioactive analytes.

The following entries are examples of the use of the DEBUG keyword:

```
DEBUG TIMES INFLUX GRID
DEBUG TOTAL
```

4.3.3 END Keyword for CRGRAB

The END keyword signifies the end of all keyword data. Nominally it will be the last keyword in the keyword file. All data in the keyword file after the END keyword will be ignored. The following is this keyword's syntax:

```
END
```

4.3.4 EXTRACT Keyword for CRGRAB

The EXTRACT keyword is used to define the type of data to be extracted. The following is this keyword's syntax:

```
EXTRACT [INFLUX]
```

The single modifier INFLUX indicates that contaminant influx to the river is to be extracted. The data extractor has not been upgraded to extract water volumes, although water volumes are sent to the river. An example use of the EXTRACT keyword is the following:

```
EXTRACT INFLUX
```

4.3.5 GRABPATH Keyword for CRGRAB

The GRABPATH keyword is used to define the directory where output files will be written. The following is this keyword's syntax:

```
GRABPATH “quote1”
```

The single quote string contains the pathname of a directory where the output files containing extracted data will be written. The path must terminate with a "/" character. An example use of the GRABPATH keyword is the following:

```
GRABPATH "/home/ANALYSIS4/CA1_median/crgrab/"
```

4.3.6 IGNORE Keyword for CRGRAB

The IGNORE keyword directs the CRGRAB code to ignore any data after a specified year. The following is this keyword's syntax:

```
IGNORE [N1]
```

The single numerical value entered on this keyword is an integer calendar year. All data after this year will be ignored. An example use of the IGNORE keyword is the following:

```
IGNORE 4000
```

4.3.7 LOCATION Keyword for CRGRAB

The LOCATION keyword is used to calculate the total release to the river by releasing location. The following is this keyword's syntax:

```
LOCATION [TOTAL]
```

The single modifier TOTAL indicates that total contaminant influx to the river is to be extracted. An example use of the LOCATION keyword is the following:

```
LOCATION TOTAL
```

4.3.8 MASS2GRID Keyword for CRGRAB

The MASS2GRID keyword is used to identify the suite of grid files used in the MASS2 code. The following is this keyword's syntax:

```
MASS2GRID [“quote 1”] {“quote 2” … “quote n”}
```

The name of each grid file is identified in a quote string. Typically the grid file names will include the pathname. At least one grid file is required. An example use of the MASS2GRID keyword for three grid files is the following:

```
MASS2GRID
  "/home/ANALYSIS4/CA1_median/mass2/hanfnad83m-pt.000"
  "/home/ANALYSIS4/CA1_median/mass2/hanfnad83m-pt.001"
  "/home/ANALYSIS4/CA1_median/mass2/hanfnad83m-pt.002"
```

4.3.9 ONLY Keyword for CRGRAB

The ONLY keyword narrows the focus of the data extraction to a single TMS (flux release) file. The following is this keyword's syntax:

```
ONLY [“quote1”]
```

The name of the single desired TMS file is identified in a quote string. An example use of the ONLY keyword is the following:

```
ONLY “/home/ANALYSIS4/CA1_median/mass2/C14/1/cfest/TMS-6499-C14.DAT"
```

4.3.10 REALIZATION Keyword for CRGRAB

The REALIZAT keyword identifies the realizations to be processed. The following is this keyword's syntax:

```
REALIZATION [ ALL | LIST N1 N2 N3 ... ]
```

The realizations requested must be a subset of the analytes for which environmental data were computed and stored by the inventory, release, and transport modules. If the modifier ALL is present then data is extracted for every realization defined in the ESD keyword file. If the LIST modifier is present then it must be followed by one or more numerical values that identify the realizations to process. Example uses of the REALIZAT keyword to extract data for all realizations and for just realizations 2, 3, and 91 are the following:

```
REALIZATION ALL
REALIZATION LIST 3 2 91
```

4.3.11 TIME Keyword for CRGRAB

The TIME keyword identifies whether data are output accumulated to calendar year or summed for the entire simulation. The following is this keyword's syntax:

```
TIME {ANNUAL} {TOTAL}
```

If the modifier ANNUAL is present then data are output accumulated by calendar year. If the modifier TOTAL is present then data are output summed for the entire simulation. At least one of the ANNUAL of TOTAL modifiers must be used. The following example keyword identifies that data are to be output on both an annual and a simulation total basis:

```
TIME ANNUAL TOTAL
```

4.3.12 TITLE Keyword for CRGRAB

The TITLE keyword is used to define a single-line problem title. The problem title will be written to output files. If the title is not supplied, then the program will error terminate. The following is this keyword's syntax:

```
TITLE [“quote”]
```

The title is entered in a quote string, which must be enclosed in double quotation marks. Titles up to 200 characters long are supported. The following example defines a title for a run of the code:

```
TITLE “Example title line for the human impacts code.”
```

4.3.13 USER Keyword for CRGRAB

The USER keyword is used to identify the user of the program. The user name will be written to output files. If the user name is not supplied, then the program will error terminate. The following is this keyword's syntax:

```
USER [“quote”]
```

The user name is entered in a quote string, which must be enclosed in double quotation marks. User names up to 16 characters long are supported. The following example defines John Q. Public as the user running the code:

```
USER “John Q. Public”
```

4.3.14 VERBOSE Keyword for CRGRAB

The presence of the optional VERBOSE keyword initiates additional output to the report file. The following is this keyword’s syntax:

```
VERBOSE
```

5.0 CSCALE – Concentration Data Scaling

5.1 Overview

The ECDA program creates environmental concentration data accumulator (ECDA) files and initializes the contents. Other programs (GWDROP, CRDROP, AIRDROP, RIPSAC, and SOIL) then fill in the concentration values for different media. The CSCALE utility program can be used to calculate concentration values in an ECDA file as a linear function of data in other ECDA files. The linear combination of data can be a function of time. The same linear combination is used for all media in a given file. Two methods can be used:

- **Method #1.** The data in the output ECDA file is a multiple of the data in a single input file. For example, uranium-238 is run as a radionuclide through the inventory, release, and transport codes of the SAC. This utility could then be used to write a concentration data file for uranium as an element (chemical) for further use in the impacts modules.

- **Method #2.** The data in the output ECDA file is the sum of multiples of the data in two input files. For example, uranium-235 and uranium-238 are run as radionuclides through the inventory, release, and transport codes of the SAC. This utility could then be used to write a concentration data file for uranium as an element (chemical) for further use in the impacts modules. The output concentration is the sum of uranium-235 times its multiplier and uranium-238 times its multiplier.

All ECDA files used by this utility program must have an analyte defined in the ESD keyword file and the file must be set up by the ECDA program.

5.1.1 Location in the Processing Sequence

This code must be run after all transport calculations have been finished. The transport calculations may include running the RIPSAC and SOIL codes depending on the problem being analyzed.

5.1.2 How the Code Is Invoked

CSCALE can run under either the Windows or the Linux operating systems. Under the Windows operating system (2000, or XP), CSCALE executes in a DOS box. A run of CSCALE is initiated by entering the following command line:

```
CSCALE “Keyfilename”
```

Under the Linux operating system CSCALE is executed through any of the following Bourne Shell or C Shell commands:

```
cscale-1.exe “Keyfilename”
```

For these commands, CSCALE.EXE or cscale-1.exe is the name of the executable program and “Keyfilename” is the name of an existing control keyword file. Both the name of the executable program and the keyword file may contain path information. If CSCALE is invoked without entering the name of the keyword file, then the code will prompt the user for the file name. If CSCALE cannot find or open the keyword file, then the code will terminate execution after writing an error message to the standard output device.

5.1.3 Memory Requirements

The CSCALE code uses dynamic memory allocation, so the memory requirements depend on the problem being analyzed. However, the code uses minimal memory because it reads and writes only a single record at a time from each of the input and output ECDA files.

5.2 File Definitions

There are three or more input files for every run of the code. There are two output files for every run of the code.

5.2.1 Input Files

The input files contain control information, the concentration data to be converted, and a concentration index file. More than one input concentration data file can be used.

5.2.1.1 Keyword Control File

The CSCALE program is controlled by the entries in a keyword file. Table 5.1 contains an example keyword file for the CSCALE code. Definitions for the individual keywords is provided in Section 5.3.

Table 5.1 Example Keyword File for the CSCALE Code

```
!
! Report File (Must be first Entry)
REPORT "Test.Rpt"
!
! Other identification information
TITLE " SAC Rev. 1 version of the CScale Code for multiple input files"
USER  "Carmen Arimescu"
!
EXECUTE
!DEBUG
!
! File names
FILE MAP    "\Sac\AAAA\ECDA_Map.CSV"
FILE OUTPUT "\SAC\AAAA\CON_U.DAT"
!
ANALYTE ID="U" UNITS="kg"
!
FILE INPUT "\SAC\AAAA\CON_U235.DAT"
! Multipliers for the data in the first concentration file
! Per the units given above, this also converts Ci to kg
SCALE
```

```
  1944 0.001
  3050 0.5
 12050 0.001
!
FILE INPUT "\SAC\AAAA\CON_U238.DAT"
! Multipliers for the data in the second concentration file
! Per the units given above, this also converts Ci to kg
SCALE
   1944  0
   2194  0.00526044
   4444  0.0513772
   7194  0.104849
  11944  0.190206
!
! End of Inputs
END
```

5.2.1.2 Input ECDA Files

At least one input concentration data (ECDA) file must be identified that will serve as the source file for computing the scaled concentrations. This code assumes the file was generated by the ECDA program, and that it contains concentration data entered by other codes of the SAC computational suite. Multiple input concentration data files may be used.

5.2.1.3 ECDA Record Map File

This input file is the record map file associated with the concentration data files. This file is an ASCII file written by the ECDA program. The same map file applies to all input and output concentration data files for a single code run.

5.2.2 Output Files

There are two output files for every run of the code. These files contain a report of the run progress and the scaled concentration data.

5.2.2.1 Report File

The report file is an ASCII file containing information about the run of the CSCALE program. It is written for every run of the code. All error messages (except for those indicating the report file could not be opened) are written to this file.

5.2.2.2 Concentration Data File

The resulting scaled concentration data file written by the code is in the same format as the input concentration data files. The header lines in the file have been modified to indicate the new concentration units. The data have all been scaled by the multipliers defined by the SCALE keyword (see Section 5.3.7).

5.3 Keywords Definitions for CSCALE

Most of the keywords for the CSCALE code can be entered in any order. The following restrictions apply on keyword order:

- The REPORT keyword must be the first keyword in the file.
- The END keyword must be the last keyword in the file.
- A SCALE keyword entry must be associated with a FILE keyword entry that defines an input file. The scaling factors are associated with the input file by their placement in the keyword file. For example, the second entry of a SCALE keyword defines the scaling factors for the second input file.

5.3.1 ANALYTE Keyword for CSCALE

The ANALYTE keyword is used to define the output analyte name and its associated concentration units. The following is this keyword's syntax:

```
ANALYTE [ID="quote1"]  [UNITS="quote2"]
```

The analyte ID and output units are entered in quote strings, which must be enclosed in double quotes. Units up to 2 characters long are supported.

Exactly one ANALYTE keyword is required for every run of the code. The modifiers associated with the ANALYTE keyword are given in Table 5.2.

Table 5.2 Modifiers Associated with the ANALYTE Keyword

Modifier	Description
ID	The quote string (up to six characters in length) associated with this modifier identifies the output analyte.
UNITS	The quote string associated with this modifier identifies the concentration data units for the output data file. The only allowable units are "Ci" or "kg". These units correspond to concentrations of of Ci/m^3 or kg/m^3 for water media and Ci/kg or kg/kg for soil or sediment media.

The following example supports converting radionuclide concentration in curies to concentration in kilograms for uranium:

```
ANALYTE ID="U"  UNITS="kg"
```

5.3.2 DEBUG Keyword for CSCALE

The DEBUG keyword is used to initiate intermediate data output statements. The following is this keyword's syntax:

```
DEBUG
```

This keyword should not be used when a large data set is being processed because it may result in a very large output file. There are no modifiers or quote strings associated with the DEBUG keyword.

5.3.3 END Keyword for CSCALE

The END keyword signifies the end of all keyword data. It should be the last keyword in the keyword file. All data in the keyword file after the END keyword will be ignored. The following is this keyword's syntax:

```
END
```

5.3.4 EXECUTE Keyword for CSCALE

The EXECUTE keyword signifies that the user wishes to perform problem execution. If this keyword is not entered the inputs are checked for consistency, but the problem will not be executed. This is useful if the run being set up is expected to take a significant amount of computation time. The syntax for this keyword record is the following:

```
EXECUTE
```

5.3.5 FILE Keyword for CSCALE

The FILE keyword is used to enter the names of input and output files except for the report file. The following is this keyword's syntax:

```
FILE [modifier1= "quote 1"] {modifier2="quote 2"} {modifier3="quote 3"}
```

The file names are entered in quote strings, which must be enclosed in double quotes. Path names up to 200 characters long are supported. The file name associated with a modifier must be entered before the next modifier is entered.

At least one FILE keyword is required for every run of the code. At least three files must be defined for every run of the code. The modifiers associated with the FILE keyword are given in Table 5.3.

Table 5.3 Modifiers Associated with the FILE Keyword

Modifier	Description
INPUT	The quote string associated with this modifier identifies an input concentration data file. The input file is assumed to be a concentration file created by the ECDA code. This file contains a set of concentration data that are to be scaled. Every input file must be declared using a separate entry of the FILE keyword. Every entry of an input file must be followed by an associated SCALE keyword before the next input file is defined.
OUTPUT	The quote string associated with this modifier identifies the output concentration data file. This output file will be a file in the same format as the input concentration file or files; however, the concentration data will all be scaled by the associated multipliers, and the data unit definitions in the header lines will be modified.
MAP	The quote string associated with this modifier identifies the record number map associated with the input concentration data file. This file is an input file nominally generated by the ECDA code.

The following example entries form a complete set of FILE keywords for a run of the code where uranium-238 concentrations are converted to uranium concentrations:

```
FILE  OUTPUT “Human_U.dat”
FILE  MAP “Human_Map.csv”
FILE  INPUT “Human_U238.dat”
```

The following example entries form a complete set of FILE keywords for a run of the code where uranium-235 and uranium-238 concentrations are converted to uranium concentrations and summed together:

```
FILE  OUTPUT “Human_U.dat”
FILE  MAP “Human_Map.csv”
FILE  INPUT “Human_U235.dat”
FILE  INPUT “Human_U238.dat”
```

5.3.6 REPORT Keyword for CSCALE

The REPORT keyword is used to define the name of the output report (log) file. It must be the first keyword entered in the keyword file. The following is this keyword’s syntax:

```
REPORT ["quote"]
```

The name of the report file is entered in a quote string. An example REPORT keyword record is the following:

```
REPORT "/SAC/SystemCodes/ECEM/Test1.rpt"
```

5.3.7 SCALE Keyword for CSCALE

The SCALE keyword defines the scaling factors for the concentration data. The following is this keyword’s syntax:

```
SCALE [value1, value2 … valueN]
```

A SCALE keyword entry must be associated with a FILE keyword entry that defines an input file. The scaling factors are associated with the input file by their placement in the keyword file. For example, the second entry of a SCALE keyword defines the scaling factors for the second input file.

The following keyword record defines time-dependent scaling factors to use with an input data file.

```
SCALE !NUMBER=41 TABLE
   1944  0
   2194  0.00526044
   2694  0.0156986
   3694  0.0362475
   4694  0.0563674
   5694  0.0760673
   6944  0.100115
   8194  0.123536
```

```
 9444  0.146348
10444  0.164169
11944  0.190206
```

5.3.8 TITLE Keyword for CSCALE

The TITLE keyword is used to define a single-line problem title. The problem title will be written to output files. If the title is not supplied the program will error terminate. The following is this keyword's syntax:

```
TITLE ["quote"]
```

The title is entered in a quote string, which must be enclosed in double quotes. Titles up to 200 characters long are supported. The following example defines a title for a run of the code:

```
TITLE "Example title line for the CScale computer code."
```

5.3.9 USER Keyword for CSCALE

The USER keyword is used to identify the user of the program. The user name will be written to output files. If the user name is not supplied the program will error terminate. The following is this keyword's syntax:

```
USER ["quote"]
```

The user name is entered in a quote string, which must be enclosed in double quotes. User names up to 16 characters long are supported. The following example defines John Q. Public as the user running the code:

```
USER "John Q. Public”
```

6.0 ECDA – Environmental Concentration Data Accumulator Setup

6.1 Overview

The ECDA code creates the binary environmental concentration data accumulator (ECDA) files and initializes the contents for each environmental medium. In addition, the ECDA code creates the stochastic data libraries of soil distribution coefficients (Kd’s) and river bank seep water dilution factors.

6.1.1 Location in the Processing Sequence

The ECDA code typically is run before any other code in an assessment. It must be run before any of the following codes can execute: AIRDROP, GWDROP, CRDROP, SOIL, RIPSAC, or any of the impacts codes.

6.1.2 How the Code Is Invoked

ECDA can run under either the Windows or the Linux operating system. Under the Windows operating system (Releases 98, NT, 2000, or XP), ECDA executes in a DOS box. A run of ECDA is initiated by entering the following command line:

```
ECDA “Keyfilename”
```

Under the Linux operating system ECDA is executed through any of the following Bourne Shell or C Shell commands:

```
ecda-1.exe “Keyfilename”
```

For these commands, ECDA.EXE or ecda-1.exe is the name of the executable program and “Keyfilename” is the name of a control keyword file. Both the name of the executable program and keyword file may contain path information. If ECDA is invoked without entering the name of the keyword file, then the code will prompt the user for the file name. The keyword file, which should be prepared using an editor that can handle ASCII files without leaving embedded control codes, contains text control information describing the run. If ECDA cannot open the keyword file, then the code will terminate execution after writing an error message to the standard output device.

6.1.3 Memory Requirements

The ECDA program has minimal memory requirements. A run of the code to create files using 1000 locations, 100 realizations, and 15 analytes required less than 4 MB of memory.

6.2 File Definitions

The ECDA program reads one input file and writes multiple output files. These files are described in the following sections.

6.2.1 Input Files for ECDA

The ECDA program reads the ESD keyword file that contains the basic control information for the entire assessment. An example file is provided in Table 6.1.

Table 6.1 Example Keyword File for ECDA

```
! Example keywords for ECDA
REPORT "ESD_CPO.rpt"
TITLE  "Central Plateau Optimization"
USER   "PWE - WEN"
DEBUG ECDA
REALIZAT 1
PERIOD START=1944 STOP=3050 CLOSURE=2070
ANALYTE ID="I129"  NAME="Iodine-129" TYPE="NR" AIR="IODINE" COMPUTE
  MOLWGT=1.289050E+02  HALFLIFE=1.570000E+07  SPECIFIC=1.767954E-04
  DFIMM=1.039647E-01   DFSED=6.930000E-20     GAMMA=2.500000E-02
  MOLDIFF=1.050000E-05 GASDIFF=8.340000E-02   HENRY=1.033515E+02
TIMES 1945  1950  1955  1960  1965  1970  1975  1980  1985  1990
!
ECHO KDSOIL ! Echo SORP-water Kd values to the ECDA report file
FILE KDSOIL NAME="/home/CPO/ecda/KDSOIL.dat" SEED=232323.0 !CREATE
KDSOIL ID="KDI"  1 0.0001316 "Soil-water Kd iodine (L/g)" UNITS="L/g"
KDSOIL ID="KDTc" 1 0.0  "Soil-water Kd technetium (L/g)" UNITS="L/g"
!
ECHO DILUTE ! Echo dilution data to the ECDA report file
FILE DILUTE NAME="/home/CPO/ecda/DILUTE_CPO.Dat" SEED=123445.0 CREATE
DILUTE ID="DF5m" 1  4.8303E-01 "Dilution factor" UNITS="none"
!
FILLECDA GWAT FIXED =  0.0
FILLECDA PWAT FIXED =  0.0
FILLECDA SWAT FIXED =  0.0
FILLECDA SEEP FIXED = -1.0
FILLECDA SEDI FIXED =  0.0
FILLECDA SORP FIXED = -1.0
FILLECDA SODR FIXED = -1.0
FILLECDA SOGW FIXED = -1.0
FILLECDA SOSW FIXED = -1.0
FILLECDA AIRC FIXED =  0.0
FILLECDA AIRD FIXED =  0.0
FILE HEADER NAME="/home/CPO/ecda/Sacview_CPO.hdr" CREATE
FILE I_ECDA  NAME="/home/CPO/ecda/ECDA_CPO.map"   CREATE
FILE C_ECDA ANALYTE="Tc99"  NAME="/home/CPO/ecda/Tc99.dat" CREATE
FILE C_ECDA ANALYTE="I129"  NAME="/home/CPO/ecda/I129.dat" CREATE
FILE C_ECDA ANALYTE="U"     NAME="/home/CPO/ecda/U.dat"    CREATE
!
LOCATION ID="UH0001" NAME="UnsuitableForAgricul"
  EASTING=576022 NORTHING=154367
  GWAT  TYPE = "UPLAND"  LOCUS = "HANFORD"
  APSD     = 4.00E-02  POROSITY= 3.50E-01  FOC    = 1.0E+00
  VEGCOV   = 5.00E-01  NECF    = 1.00E+00  RHOS   = 1.50E+00
  TEMP     = 2.85E+02  MSWIND  = 3.44E+00  MZWIND = 3.44E+00
  IRG_SWAT ="QHP025"   AREA    = 1457376.1 SRH  = 1.8E-02
!
LOCATION ID="RHP001" NAME="RiparianVernitaBridge"
  EASTING=557379.2 NORTHING=144876.4
  GWAT  TYPE="RIPARIAN"  LOCUS="HANFORD"  MILE=389.00
  APSD    = 4.00E-02  COXYGEN = 1.10E-02  POROSITY= 3.50E-01
  NECF    = 1.00E+00  RHOS    = 1.50E+00  SRH     = 1.8E-02
  MZWIND  = 3.44E+00  AREA    = 9360      VEGCOV  = 5.00E-01
  MSWIND  = 3.44E+00  FOC  = 1.0E+00      TEMP = 2.85E+02
!
LOCATION ID="QHP025" NAME="AquaticBenton100BC"
  EASTING=566217.6 NORTHING=145715.6 SWAT SEDI PWAT
  TYPE="AQUATIC"  LOCUS="HANFORD"  MILE=383.20
  COXYGEN = 1.10E-02 AREA = 2.5E+03
!
LOCATION ID="AH0001" NAME="Air 100 D Area" EASTING=574436.00
  NORTHING=151194.98 GWAT SOGW SOSW SODR AIRC AIRD TEMP=2.85E+02
  TYPE="UPLAND"  LOCUS="HANFORD", IRG_SWAT ="QHP136" VEGCOV=5.00E-01
  APSD    = 4.00E-02  POROSITY= 3.50E-01  FOC  = 1.0E+00
  NECF    = 1.00E+00  RHOS    = 1.50E+00  SRH  = 1.8E-02
  MSWIND  = 3.44E+00  MZWIND  = 3.44E+00  AREA = 2500
!
END
```

6.2.2 Output Files for ECDA

The ECDA code writes a number of output files. A report file is always written The number of other files written depend on the input options. The following list summarizes the output files:

- **Report file.** A report file, named “ecda.rpt” is written for every run of the code. This file is a text file containing information about the run progress. Error messages are written to this file as well.
- **ECDA files.** A separate ECDA file is written for every analyte if the CREATE and C_ECDA modifiers are present on the FILE keyword. The file format is described in Section 2.2.1 of (Eslinger et al. 2004a). These binary files contain the media concentrations calculated by the transport codes that are saved for use in the impacts codes.
- **Map index file.** This file is written if a file name is entered in the keyword file (FILE keyword, I_ECDA modifier). The file format is described in Section 2.2.2 of (Eslinger et al. 2004a). This file contains indexing information to quickly locate data for specific times and locations in the binary ECDA files.
- **SacView header file.** This file is written if a file name is entered in the keyword file (FILE keyword, I_ECDA modifier). The file format is described in Section 2.2.3 of (Eslinger et al. 2004a). This file is used in the SacView data extractor to facilitate retrieving and displaying data.
- **Dilution file.** This file is written if the CREATE modifier is present in the keyword file (FILE keyword, DILUTE modifier). The file format is described in Section 2.4.1 of (Eslinger et al. 2004a). This file contains a library of generated dilution factors that describe the mixing of groundwater and river water in the riparian zone modeling that are used in the RIPSAC code.

- **Kdsoil file.** This file is written if the CREATE modifier is present in the keyword file (FILE keyword, KDSOIL modifier). The file format is described in Section 2.4.2 of (Eslinger et al. 2004a). This file contains a library of generated soil-water partition coefficients (Kd values) that are used in a number of other SAC programs.

6.3 Keyword Definitions for ECDA

The ECDA code gets all of its control information from keywords in the ESD keyword file. In general, keywords for the ECDA code can be entered in any order. The only restriction is that the END keyword must be the last keyword in the file. The following keyword descriptions have been modified from those in Section 2.1 of (Eslinger et al. 2004a) to restrict the definition to information used by the ECDA code.

6.3.1 ANALYTE Keyword for ECDA

The ANALYTE keyword is used to define the analytes to be used in the simulation. The following is this keyword's syntax:

```
ANALYTE [ID="quote 1"] [TYPE="quote 2"] [NAME="quote 3"]
```

A separate ANALYTE keyword must be entered for every analyte to be included in the simulation. Table 6.2 provides the modifiers associated with the ANALYTE keyword.

Table 6.2 Modifiers Associated with the ANALYTE Keyword for ECDA

Modifier	Description
ID	The quote string associated with the ID modifier is an analyte identification string up to six characters in length. The analyte identification string is case sensitive, and spaces or hyphens change the definition. All data in the analyte identification strings must satisfy the following conventions: • Only the first entry in the analyte identification string is capitalized. • No embedded spaces or hyphens are used, even for radionuclides. • Individual elements are defined using the standard element abbreviation.
TYPE	The quote string associated with the TYPE modifier string is a two-character analyte type indicator. The following are the valid entries for this string: • NR – if the analyte is a radioactive element or an inorganic compound containing a radionuclide • NS – if the analyte is a stable (nonradioactive) element or inorganic compound • OR – if the analyte is an organic compound containing a radionuclide • OS – if the analyte is an organic compound, containing a stable (nonradioactive) elemental analyte or compound
NAME	The quote string associated with the NAME modifier is an analyte name or description up to 72 characters in length.

The following ANALYTE keywords select the analytes tritium, uranium-233 and the chemical uranium for analysis:

```
ANALYTE ID="H3"  NAME="Tritium"  TYPE= "NR"
ANALYTE ID="U235"  NAME="Uranium-235"  TYPE="NR"
ANALYTE ID="U"  NAME="Uranium"  TYPE="NS"
```

6.3.2 DEBUG Keyword for ECDA

The DEBUG keyword initiates output of detailed information on the processing by the code. The following is this keyword's syntax:

```
DEBUG {ECDA}
```

This keyword has no effect if the modifier ECDA is not present. The following keyword will initiate debug outputs:

```
DEBUG ECDA
```

6.3.3 DILUTE Keyword for ECDA

The DILUTE keyword is used to enter the definition of a statistical distribution for stochastic water dilution variables used in the riparian zone water mixing model. The following is this keyword's syntax:

```
DILUTE [ID="quote1"] [Dist_Index Parameters] {TRUNCATE U1 U2}
   {LABEL="quote2"}  [UNITS="quote3"]
```

The quote string associated with the ID modifier is a unique character string of up to 20 characters that will be used to identify this stochastic variable for subsequent uses. It is case sensitive and embedded spaces are significant. The quote string associated with the optional modifier LABEL contains a description for the stochastic variable that can be up to 64 characters long. An entry for quote2 is not required, although it is used for labeling purposes if present. However, if the modifier LABEL is present the associated quote string must be entered as well. The quote string associated with the UNITS modifier contains a units descriptor for the data. The strings "none" or "unitless" should be used if the variable is unitless. Section 1.0 contains further information about generating stochastic values.

A dilution factor that is triangular on the triple (0.2, 0.5, 0.99) could use the following keyword entry:

```
DILUTE ID="ID#1" 6 0.2 0.5 0.99 LABEL="Example dilution factor"
   UNITS="none"
```

The stochastic values generated from information on the DILUTE keyword are used only in the RIPSAC code.

6.3.4 ECHO Keyword for ECDA

The ECHO keyword is used to initiate output of summary information code when processing the statistical distributions defined by KDSOIL and DILUTE keywords. The following is this keyword’s syntax:

```
ECHO {KDSOIL} {DILUTE}
```

If the KDSOIL modifier is present then variable definitions and summary statistics on generated values will be written to the report file for every distribution specified on a KDSOIL keyword. If the DILUTE modifier is present then variable definitions and summary statistics on generated values will be written to the report file for every distribution specified on a DILUTE keyword. Example uses of this keyword are the following:

```
ECHO KDSOIL
ECHO DILUTE
ECHO DILUTE KDSOIL
```

6.3.5 END Keyword for ECDA

The END keyword signifies the end of all keyword data. It should be the last keyword in the keyword file. Any data in the keyword file after the END keyword will be ignored. The following is this keyword’s syntax:

```
END
```

There are no modifiers or quote strings associated with the END keyword.

6.3.6 FILE Keyword for ECDA

The FILE keyword is used to enter the names of many of the files used in a simulation run. The following is this keyword’s syntax:

```
FILE [NAME=“quote 1”] {ANALYTE=“quote 2”}
   [ HEADER | KDSOIL | DILUTE | C_ECDA | I_ECDA ] {CREATE}
```

The file names must be entered with the complete path. One FILE keyword is required for every analyte for which concentrations are to be generated. Every file definition requires the entry of a separate FILE keyword. Table 6.3 provides the file type modifiers associated with the FILE keyword.

Table 6.3 Modifiers Associated with the FILE Keyword for ECDA

Modifier	Description
NAME	The quote string associated with the NAME modifier is a file name (including path) up to 200 characters in length.
ANALYTE	If the file name is associated with the C_ECDA modifier, the ANALYTE modifier must also be entered to indicate which analyte is to be associated with the ECDA concentration data file.
HEADER	This modifier indicates that the FILE keyword is defining a header file for use by the SACView graphical user interface that allows extraction of human-readable concentration data.
KDSOIL	This modifier indicates that the FILE keyword is defining a file to contain stochastic realizations of all of the random variables defined using the KDSOIL keyword.
DILUTE	This modifier indicates that the FILE keyword is defining a file to contain stochastic realizations of all of the random variables defined using the DILUTE keyword.
C_ECDA	This modifier indicates that the FILE keyword is defining an ECDA concentration data file. Concentrations for each analyte are contained in separate files. The ANALYTE modifier is used to associate an analyte with this file.
I_ECDA	This modifier indicates that the FILE keyword is defining a record index file for mapping into all ECDA concentration files.
CREATE	When this modifier is present, the file will be created. If the CREATE modifier is not present, no actions occur for that file.

The following example entries define the concentration files for a suite of analytes:

```
FILE C_ECDA ANALYTE="H3" NAME="/test/ecda/H3_median.dat"  CREATE
FILE C_ECDA ANALYTE="C14" NAME="/test/ecda/C14_median.dat"  CREATE
FILE C_ECDA ANALYTE="I129" NAME="/test/ecda/I129_median.dat" CREATE
FILE C_ECDA ANALYTE="Tc99" NAME="/test/ecda/Tc99_median.dat" CREATE
```

If present, the CREATE flag causes the following actions:

- Deletion of any existing file by that name and creation of a new file.
- If the file is associated with the HEADER, DILUTE, KDSOIL, INFILT, or I_ECDA modifiers, new idata are written to the file.
- If the file is associated with the C_ECDA modifier, the concentration data are initialized to the values identified by the FILLECDA keyword.

The following example entries define the files for soil-water Kd values and the water dilution factors for the river-shore module:

```
FILE KDSOIL NAME="KDSOIL.CSV" SEED=23232.0 CREATE
FILE DILUTE NAME="DILUTE.CSV" SEED=12345.0 CREATE
```

The following example entries define the SACView Header file and ECDA record number index files:

```
   FILE HEADER NAME="/test/ecda/Sacview_median.hdr" CREATE
   FILE I_ECDA  NAME="/test/ecda/ECDA_median.map"  CREATE
```

6.3.7 FILLECDA Keyword for ECDA

The FILLECDA keyword controls filling ECDA files with fixed or random concentrations as they are initialized. The following is this keyword's syntax:

```
   FILLECDA [GWAT|SWAT|AIRC|AIRD|PWAT|SEDI|SORP|SEEP|SODR|SOGW|SOSW]
      [FIXED=N1 | RAND ]
```

The value N_1 is the single value to be used to initialize the EDCA concentration file for the specified media. Concentrations can be set to zero, or an invalid value. A separate FILLECDA keyword must be used for each media. Table 6.4 describes the modifiers associated with the FILLECDA keyword.

Table 6.4 Modifiers Associated with the FILLECDA Keyword in the ESD File

Modifier	Description
GWAT	Presence of this optional modifier indicates that groundwater concentrations will be initialized with the specified value.
SWAT	Presence of this optional modifier indicates that surface water concentrations will be initialized with the specified value.
AIRC	Presence of this optional modifier indicates that atmospheric concentrations will be initialized with the specified value.
AIRD	Presence of this optional modifier indicates that atmospheric deposition rates will be initialized with the specified value.
PWAT	Presence of this optional modifier indicates that river bottom pore water concentrations will be initialized with the specified value.
SEDI	Presence of this optional modifier indicates that sediment concentrations (on the river bottom) will be initialized with the specified value.
SORP	Presence of this optional modifier indicates that riparian zone soil concentrations (on the land surface) will be initialized with the specified value.
SEEP	Presence of this optional modifier indicates that seep water concentrations (on the land surface) will be initialized with the specified value.
SODR	Presence of this optional modifier indicates that upland soil concentrations from non-irrigated soils will be initialized with the specified value.
SOGW	Presence of this optional modifier indicates that upland soil concentrations using groundwater for irrigation will be initialized with the specified value.
SOSW	Presence of this optional modifier indicates that upland soil concentrations using surface water for irrigation will be initialized with the specified value.

Modifier	Description
RAND	This modifier initializes the values to the media to a random number on the interval (0,1).
FIXED	The numerical value associated with this modifier is used to initialize all entries for the selected media.

The following example entries assign values to use in initialization of the ECDA files:

```
FILLECDA GWAT FIXED =  0.0
FILLECDA PWAT FIXED =  0.0
FILLECDA SWAT FIXED =  0.0
FILLECDA SEEP FIXED = -1.0
FILLECDA SEDI FIXED =  0.0
FILLECDA SORP FIXED = -1.0
FILLECDA SODR FIXED = -1.0
FILLECDA SOGW FIXED = -1.0
FILLECDA SOSW FIXED = -1.0
FILLECDA AIRC FIXED =  0.0
FILLECDA AIRD FIXED =  0.0
```

6.3.8 KDSOIL Keyword in ECDA

The KDSOIL keyword is used to enter the definition of a statistical distribution for the solid-aqueous distribution coefficient (K_d) to be used for calculating soil concentrations from groundwater concentrations. The following is this keyword's syntax:

```
KDSOIL [ID="quote1"] [Dist_Index Parameters] {TRUNCATE U1 U2}
   {LABEL="quote2"}  [UNITS="quote3"]
```

The quote string associated with the ID modifier is a unique character string of up to 20 characters that will be used to identify this stochastic variable for subsequent uses. It is case sensitive and embedded spaces are significant. The quote string associated with the optional modifier LABEL contains a description for the stochastic variable that can be up to 64 characters long. An entry for quote2 is not required, although it is used for labeling purposes if present. However, if the modifier LABEL is present the associated quote string must be entered as well. The quote string associated with the UNITS modifier contains a units descriptor for the data. The strings "none" or "unitless" should be used if the variable is unitless. Section 1.0 contains further information about generating stochastic values.

A K_d that is triangular on the triple (0.2, 0.5, 0.99) could use the following keyword entry:

```
KDSOIL ID="ID#1" 6 0.2 0.5 0.99 LABEL="Example Kd" UNITS= "L/g"
```

The data entered by this keyword are used in the river model MASS2 and in the riparian zone model RIPSAC.

6.3.9 LOCATION Keyword for ECDA

The LOCATION keyword identifies the locations where concentrations will be generated for use in the impacts modules. The following is this keyword's syntax:

```
LOCATION [ID="quote1"] [EASTING=N1] [NORTHING=N2] {GWAT} {SWAT}
   {AIRC} {AIRD} {PWAT} {SEDI} {SORP} {SEEP} {SODR} {SOGW} {SOSW}
   [NAME="quote2"] {LOCUS="quote3"}
```

Table 6.5 provides the modifiers and associated data for the LOCATION keyword.

Table 6.5 Modifiers Associated with the LOCATION Keyword in the ESD File

Modifier	Description
ID	The location identification string is entered using the ID modifier. This string is limited to 6 characters and must be unique. It is used to associate other data with a specific location.
EASTING	This entry is associated with the easting coordinate for the location. These coordinates are expressed in terms of the Lambert projection of the Washington State Plane North American Datum of 1983, expressed in meters.
NORTHING	This entry is associated with the northing coordinate for the location. These coordinates are expressed in terms of the Lambert projection of the Washington State Plane North American Datum of 1983, expressed in meters.
NAME	The quote string associated with the NAME modifier contains a descriptive name of up to 64 characters in length that is used for labeling purposes.
LOCUS	The quote string associated with the LOCUS modifier is used in the MOBILEHOME code to identify a river bank boundary for ecological species. The two valid options are "HANFORD" and "FARSIDE".
GWAT	Presence of this optional modifier indicates that groundwater concentrations will be computed at this location.
SWAT	Presence of this optional modifier indicates that surface water concentrations will be computed at this location.
AIRC	Presence of this optional modifier indicates that atmospheric concentrations will be computed at this location.
AIRD	Presence of this optional modifier indicates that atmospheric deposition rates will be computed at this location.
PWAT	Presence of this optional modifier indicates that river bottom pore water concentrations will be computed at this location.
SEEP	Presence of this optional modifier indicates that seep water concentrations (on the land surface) will be computed at this location.
SODR	Presence of this optional modifier indicates that upland soil concentrations from non-irrigated soils will be initialized with the specified value.

Modifier	Description
SORP	Presence of this optional modifier indicates that riparian zone soil concentrations (on the land surface) will be computed at this location.
SEDI	Presence of this optional modifier indicates that sediment concentrations (on the river bottom) will be computed at this location.
SOGW	Presence of this optional modifier indicates that soil concentrations using groundwater for irrigation will be computed at this location.
SOSW	Presence of this optional modifier indicates that soil concentrations using surface water for irrigation will be computed at this location. Refer to the IRG_SWAT modifier to specify the surface water location.

The following example keywords define three impact locations:

```
LOCATION ID="HL0151" NAME="Upland location"
   EASTING=594737.5  NORTHING=127827.4  GWAT
LOCATION ID="HL0417" NAME="Riparian zone location"
   EASTING=557375.3  NORTHING=144885.2  GWAT SORP SEEP
LOCATION ID="HL0413" NAME="Richland municipal water intake"
   EASTING=595445.1  NORTHING=109753.5  SWAT PWAT SEDI
```

6.3.10 PERIOD Keyword for ECDA

The PERIOD keyword identifies the start and stop times for the entire simulation. The following is this keyword's syntax:

```
PERIOD [START=year1]  [STOP=year2]  [CLOSURE=year3]
```

The modifier START and the associated value $year_1$ identify the start year for the simulation. The start of the simulation period must be 1944 or later or the inventory code will error terminate. The modifier STOP and the associated value $year_2$ identify the end year for the simulation. Start and stop years should be entered as whole numbers with the stop year no smaller than the start year. The modifier CLOSURE and the associated value $year_3$ identify the year that site closure occurs. The year of site closure cannot be smaller than the start year. The following example PERIOD keyword simulates from 1944 through 3050 with site closure occurring at 2050:

```
PERIOD START=1944 STOP=3050 CLOSURE 2050
```

6.3.11 REALIZAT Keyword for ECDA

The REALIZAT keyword identifies the number of realizations to be simulated. The following is this keyword's syntax:

```
REALIZAT N1
```

The number of realizations is given by the single numerical entry N_1. The valid number of realizations is 1 to 9999. Run times and disk storage requirements are directly proportional to the number of realizations. The following is an example REALIZAT keyword requesting 25 realizations:

```
REALIZAT 25
```

6.3.12 TITLE Keyword for ECDA

The TITLE keyword is used to define a single-line problem title. The problem title will be written to output files. If the title is not supplied, then the program will error terminate. The following is this keyword's syntax:

```
TITLE [“quote”]
```

The title is entered in a quote string, which must be enclosed in double quotation marks. Titles up to 200 characters long are supported. The following example defines a title for a run of the code:

```
TITLE “Example title line for the human impacts code.”
```

6.3.13 USER Keyword for ECDA

The USER keyword is used to identify the user of the program. The user name will be written to output files. If the user name is not supplied, then the program will error terminate. The following is this keyword's syntax:

```
USER [“quote”]
```

The user name is entered in a quote string, which must be enclosed in double quotation marks. User names up to 16 characters long are supported. The following example defines John Q. Public as the user running the code:

```
USER “John Q. Public”
```

7.0 ECDA_ASCII – Environmental Concentration Data Accumulator File Conversions

7.1 Overview

The ECDA_ASCII program converts binary environmental concentration files into text format. The sole purpose of the conversion is to create a file that is human readable in a text editor.

7.1.1 Location in the Processing Sequence

The ECDA_ASCII can be executed any time after the ECDA files have been created. However, other programs such as AIRDROP, GWDROP, CRDROP, RIPSAC, and SOIL write concentration data to these files. Thus, ECDA_ASCII is usually run after all of the other codes have been executed.

7.1.2 How the Code Is Invoked

ECDA_ASCII can run under either the Windows or the Linux operating system. Under the Windows operating system (Releases 98, NT, 2000, or XP), ECDA_ASCII executes in a DOS box. A run of ECDA_ASCII is initiated by entering the following command line:

```
   ECDA_ASCII
```

Under the Linux operating system ECDA_ASCII is executed through any of the following Bourne Shell or C Shell commands:

```
   ecda_ascii.exe
```

For these commands, ECDA_ASCII .EXE or ecda_ascii.exe is the name of the executable program.

7.1.3 Memory Requirements

The ECDA_ASCII code uses dynamic memory allocation. However, only one line of the input file is in memory at any given time, so the memory requirements are minimal. It is expected that most, if not all, of the runs of the ECDA_ASCII code will require less than 2 MB of memory.

7.2 File Definitions

The ECDA_ASCII code reads one file and writes one file.

7.2.1 Input Files

The ECDA_ASCII code reads an environmental concentration data accumulator (ECDA) file. This file is in binary format and is created by the ECDA program. Other programs such as AIRDROP, GWDROP, CRDROP, RIPSAC, and SOIL write concentration data to these ECDA files.

7.2.2 Output Files

The ECDA_ASCII code writes one data file. This file is a text translation of the binary ECDA file. An example of the first 20 lines of the output text file when only the groundwater media is selected is provided in Table 7.1. The numbers in the left column are record numbers that are not in the original data set. The file starts with an analyte identifier followed by the units of the 11 possible media. The data for multiple realizations are written on each line. This file only contained one realization of data.

Table 7.1 Excerpts from an ECDA_ASCII Output File

```
File: N:\CA1_median\ecda\Eu152_CA1_median.dat
       1 Eu152
       2 Ci/m^3
       3 Ci/m^3
       4 Ci/m^3
       5 Ci/m^3
       6 Ci/kg
       7 Ci/kg
       8 Ci/kg
       9 Ci/kg
      10 Ci/kg
      11 Ci/m^3
      12 Ci/m^2/yr
      13    1945 UH0001 GWAT  0.00000E+00
      19    1945 UH0002 GWAT  0.00000E+00
      25    1945 UH0003 GWAT  0.00000E+00
      31    1945 UH0004 GWAT  0.00000E+00
      37    1945 UH0005 GWAT  0.00000E+00
      43    1945 UH0006 GWAT  0.00000E+00
      49    1945 UH0007 GWAT  0.00000E+00
```

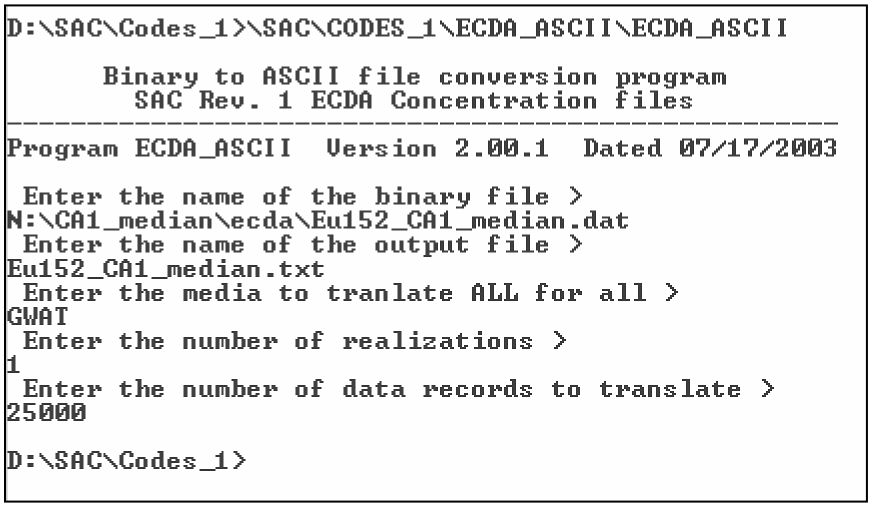
7.3 Code Execution

The ECDA_ASCII program runs in an interactive mode. A screen image from a run of the code on a Windows machine is provided in Figure 7.1. The user must provide answers to five prompts:

1. The name of the binary ECDA file must be provided. A full pathname is required unless the ECDA file is in the same directory where the ECDA_ASCII code was invoked.
2. The name of the output file must be provided. The user can select any name. An extension of ".txt" is recommended.
3. The media to translate must be provided (either ALL or one of the 4 character abbreviations provided below). The entry ALL will translate all media. The individual media options are:
   - GWAT – concentrations in groundwater (Ci/m^3 or kg/m^3)
   - SEEP – concentrations in seep water (Ci/m^3 or kg/m^3)
   - SWAT – concentrations in surface water (river) (Ci/m^3 or kg/m^3)
   - PWAT – concentrations in river bottom pore water (Ci/m^3 or kg/m^3)
   - SEDI – concentrations in river bottom sediment ($Ci/kg_{sediment}$ or $kg_{analyte}/kg_{sediment}$)
   - SORP – concentrations in riparian zone soil (land surface) (Ci/kg_{soil} or $kg_{analyte}/kg_{soil}$)

- SODR – concentrations in upland soil (land surface) with no irrigation (Ci/kg_{soil} or $kg_{analyte}/kg_{soil}$)
- SOGW – concentrations in upland soil (land surface) with groundwater irrigation (Ci/kg_{soil} or $kg_{analyte}/kg_{soil}$)
- SOSW – concentrations in upland soil (land surface) with surface water irrigation (Ci/kg_{soil} or $kg_{analyte}/kg_{soil}$)
- AIRC – concentrations in air (Ci/m^3 or kg/m^3)
- AIRD – air deposition rates ($Ci/m^2/yr$ or $kg/m^2/yr$)

4. The number of realizations for the simulation must be provided.
5. The number of data records to be translated must be provided. Care must be taken when translating a large ECDA file such that the output file does not exceed 2.1 gigbytes in size. If the output file exceeds that size the program will error terminate.



```
D:\SAC\Codes_1>\SAC\CODES_1\ECDA_ASCII\ECDA_ASCII

      Binary to ASCII file conversion program
        SAC Rev. 1 ECDA Concentration files
----------------------------------------------------
Program ECDA_ASCII  Version 2.00.1  Dated 07/17/2003

 Enter the name of the binary file >
N:\CA1_median\ecda\Eu152_CA1_median.dat
 Enter the name of the output file >
Eu152_CA1_median.txt
 Enter the media to tranlate ALL for all >
GWAT
 Enter the number of realizations >
1
 Enter the number of data records to translate >
25000

D:\SAC\Codes_1>
```

Figure 7.1 Screen Image of an ECDA_ASCII Run

8.0 FCDA_ASCII – Food Concentration Data File Conversions

8.1 Overview

The FCDA_ASCII program converts binary food concentration files into text format. The sole purpose of the conversion is to create a file that is human readable in a text editor.

8.1.1 Location in the Processing Sequence

The FCDA_ASCII can be executed any time after the ECEM (see Eslinger et al. [2004b], Volume 2, Section 3.0) code has created the food concentration data files.

8.1.2 How the Code Is Invoked

FCDA_ASCII can run under either the Windows or the Linux operating system. Under the Windows operating system (Releases 98, NT, 2000, or XP), ECDA_ASCII executes in a DOS box. A run of FCDA_ASCII is initiated by entering the following command line:

```
FCDA_ASCII
```

Under the Linux operating system FCDA_ASCII is executed through any of the following Bourne Shell or C Shell commands:

```
fcda_ascii.exe
```

For these commands, FCDA_ASCII .EXE or fcda_ascii.exe is the name of the executable program.

8.1.3 Memory Requirements

The FCDA_ASCII code uses dynamic memory allocation. However, only one line of the input file is in memory at any given time, so the memory requirements are minimal. It is expected that most, if not all, of the runs of the FCDA_ASCII code will require less than 2 MB of memory.

8.2 File Definitions

The FCDA_ASCII code reads one file and writes one file.

8.2.1 Input Files

The FCDA_ASCII code reads an food concentration data accumulator (FCDA) file. This file is in binary format and is created by the ECEM program.

8.2.2 Output Files

The FCDA_ASCII code writes one data file. This file is a text translation of the binary FCDA file. An example of the first 20 lines of the output text file for a file is provided in Figure 8.1. The numbers in the left column are record numbers that are not in the original data set. The file starts with an analyte identifier followed by species identifier and the location type identifier. This file only contained one realization of data.

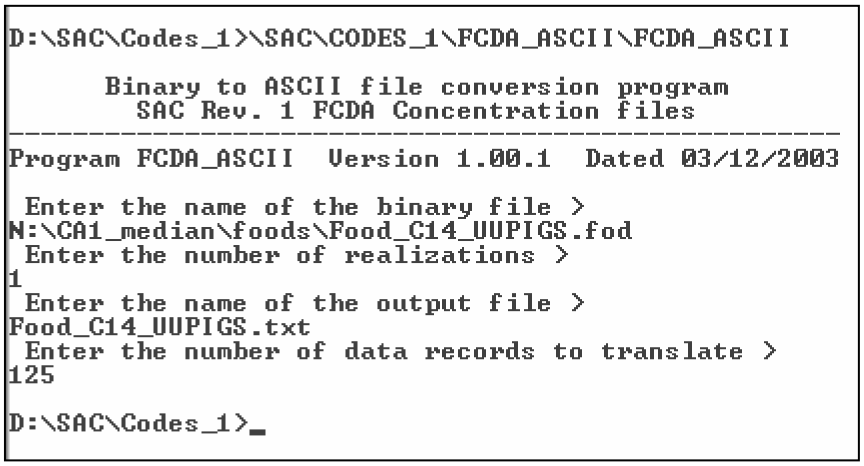
Table 8.1 Excerpts from an FCDA_ASCII Output File

```
  File: N:\CA1_median\foods\Food_C14_UUPIGS.fod
        1 C14
        2 UUPIGS
        3 UPLAND
        4    1945 UH0001  1.85791E+00
        5    1945 UH0001  1.85791E+00
        6    1945 UH0001  1.85791E+00
        7    1945 UH0002  1.85791E+00
        8    1945 UH0002  1.85791E+00
        9    1945 UH0002  1.85791E+00
       10    1945 UH0003  1.85791E+00
       11    1945 UH0003  1.85791E+00
       12    1945 UH0003  1.85791E+00
       13    1945 UH0004  1.39627E+00
       14    1945 UH0004  1.39627E+00
       15    1945 UH0004  1.39627E+00
       16    1945 UH0005  1.39627E+00
       17    1945 UH0005  1.39627E+00
       18    1945 UH0005  1.39627E+00
       19    1945 UH0006  1.85791E+00
```

8.3 Code Execution

The FCDA_ASCII program runs in an interactive mode. A screen image from a run of the code on a Windows machine is provided in Figure 8.1. The user must provide answers to four prompts:

1. The name of the binary FCDA file must be provided. A full pathname is required unless the FCDA file is in the same directory where the FCDA_ASCII code was invoked.
2. The number of realizations for the simulation must be provided.
3. The name of the output file must be provided. The user can select any name. An extension of ".txt" is recommended.
4. The number of data records to be translated must be provided. Care must be taken when translating a large FCDA file such that the output file does not exceed 2.1 gigbytes in size. If the output file exceeds that size the program will error terminate.



```
D:\SAC\Codes_1>\SAC\CODES_1\FCDA_ASCII\FCDA_ASCII

      Binary to ASCII file conversion program
        SAC Rev. 1 FCDA Concentration files
----------------------------------------------------
Program FCDA_ASCII  Version 1.00.1  Dated 03/12/2003

 Enter the name of the binary file >
N:\CA1_median\foods\Food_C14_UUPIGS.fod
 Enter the number of realizations >
1
 Enter the name of the output file >
Food_C14_UUPIGS.txt
 Enter the number of data records to translate >
125

D:\SAC\Codes_1>_
```

Figure 8.1 Screen Image of an FCDA_ASCII Run

9.0 FILLECDA – Concentration Data File Generation

9.1 Overview

The ECDA program creates environmental concentration data accumulator (ECDA) files and initializes the contents. Other programs (GWDROP, CRDROP, AIRDROP, RIPSAC, and SOIL) then fill in the concentration values for different media. The FILLECDA program provides the capability to insert user-specified concentration values into any record in an ECDA file.

9.1.1 Location in the Processing Sequence

The FILLECDA program can be used any time after the ECDA files are created by the ECDA program.

9.1.2 How the Code Is Invoked

FILLECDA can run under either the Windows or the Linux operating system. Under the Windows operating system (Releases 98, NT, 2000, or XP), FILLECDA executes in a DOS box. A run of FILLECDA is initiated by entering the following command line:

```
   FILLECDA “Keyfilename”
```

Under the Linux operating system FILLECDA is executed through any of the following Bourne Shell or C Shell commands:

```
   fillecda-1.exe “Keyfilename”
```

For these commands, FILLECDA.EXE or fillecda-1.exe is the name of the executable program and “Keyfilename” is the name of a control keyword file. Both the name of the executable program and the keyword file may contain path information. If FILLECDA is invoked without entering the name of the keyword file, then the code will prompt the user for the file name. The keyword file, which should be prepared using an editor that can handle ASCII files without leaving embedded control codes, contains text control information describing the run. If FILLECDA cannot open the keyword file, then the code will terminate execution after writing an error message to the standard output device.

9.1.3 Memory Requirements

The FILLECDA program has minimal memory requirements. The memory is small because data for only a single record of an ECDA file is generated and written at any one time.

9.2 File Definitions

The FILLECDA program reads three input files and writes to two or more output files. These files are described in the following paragraphs.

9.2.1 Input Files

9.2.1.1 FILLECDA Keyword File

The FILLECDA keyword file contains control information and also defines the concentration values to be inserted in the ECDA files. An example file is provided in Table 9.1. Detailed definitions of the keywords are provided in Section 9.3.

Table 9.1 Example Keyword File for the FILLECDA Program

```
! Keyword file for the FillECDA program
REPORT "Test.Rpt"                ! Report file name (first keyword)
TITLE "Test the FillECDA Code"   ! Title for labeling purposes
USER "Paul W. Eslinger"          ! User name for labeling purposes
FILE ESD "ESD.Key"               ! Name of the ESD keyword file
REALIZAT 1                       ! Number of realizations
DATA LOCATION="EL0001" ANALYTE="C14"  YEAR=10000 MEDIA="GWAT" VALUES=1
DATA LOCATION="EL0001" ANALYTE="I129" YEAR=10000 MEDIA="SEEP" VALUES=1.1
DATA LOCATION="EL0001" ANALYTE="U236" YEAR=10000 MEDIA="SORP" VALUES=1.2
DATA LOCATION="EL0002" ANALYTE="U238" YEAR=10000 MEDIA="SWAT" VALUES=11
DATA LOCATION="EL0002" ANALYTE="Tc99" YEAR=10000 MEDIA="SWAT" VALUES=11.1
DATA LOCATION="EL0002" ANALYTE="Tc99" YEAR=10000 MEDIA="PWAT" VALUES=11.2
DATA LOCATION="EL0002" ANALYTE="Tc99" YEAR=10000 MEDIA="SDAT" VALUES=11.3
DATA LOCATION="EL0003" ANALYTE="Tc99" YEAR=10000 MEDIA="AIRC" VALUES=11.4
DATA LOCATION="EL0003" ANALYTE="Tc99" YEAR=10000 MEDIA="AIRD" VALUES=11.5
END
```

9.2.1.2 Other Input Files

The FILLECDA program also reads two other input files. These files are the ESD keyword file and the ECDA index file. The ESD keyword file contains the names of the concentration files that are to be modified. The ECDA map file contains index data related to writing to the ECDA concentration files.

9.2.2 Output Files

The code writes to two or more output files. These files are a report file and one or more ECDA concentration files.

9.2.2.1 FILLECDA Report File

The report file contains summary information from the run of the FILLECDA code. This file contains a record of the code version, time and date of run execution, input and output file names, run information, and a summary table of results. This file also contains any error messages generated by the FILLECDA code.

9.2.2.2 Concentration Files

The FILLECDA program writes some data to an ECDA file for every analyte identified on a DATA keyword in the FILLECDA keyword file.

9.3 Keyword Definitions for FILLECDA

In general, the keywords for FILLECDA can be entered in any order. The following restrictions apply on keyword order:

- The REPORT keyword must be the first keyword in the file.
- The END keyword must be the last keyword in the file.

9.3.1 DATA Keyword for FILLECDA

The DATA keyword is used to define the data values to be inserted into one or more ECDA files. The following is this keyword's syntax:

```
DATA [LOCATION="quote"] [MEDIA="quote"] [ANALYTE="quote"] [YEAR=number]
   [VALUES=number, …, number]
```

This keyword identifies the data for all realizations of concentration for a specific analyte, location, media, and time combination. More than one DATA keyword can be entered. The modifiers for the DATA keyword are defined in Table 9.2.

Table 9.2 Modifiers Associated with the DATA Keyword in FILLECDA

Modifier	Description
ANALYTE	The quote string associated with the ANALYTE modifier contains the analyte ID for the concentration data values. The analyte ID must be one of the set of analytes defined in the ESD keyword file.
LOCATION	The quote string associated with the LOCATION modifier contains the location ID for the concentration data values. The location ID must be one of the set of locations defined in the ESD keyword file.
MEDIA	The quote string associated with the MEDIA modifier contains the media ID for the concentration data values. The media ID must be one of the following: GWAT, SEEP, SWAT, PWAT, SEDI, SORP, SODR, SOGW, SOSW, AIRC, or AIRD.
YEAR	The numerical value associated with the YEAR modifier identifies the calendar year at which the concentration data apply. The year entered must be one of the years defined on the TIMES keyword in the ESD keyword file.
VALUES	The numerical values associated with the VALUES modifier contain the concentration values to be entered in the ECDA file. There must be as many values entered as there are realizations defined in the ESD keyword file.

If a DATA keyword does not identify a valid analyte, location, media, and time combination in the ECDA data file, then the information on that DATA keyword is discarded and an error message is written to the report file. Example entries for the DATA keyword for a case of two realizations are the following:

```
DATA LOCATION="EL01" ANALYTE="I129" YEAR=10000 MEDIA="SEEP" VALUES=1.1 1.3
DATA LOCATION="EL01" ANALYTE="U236" YEAR=10000 MEDIA="SORP" VALUES=1.2 1.4
DATA LOCATION="EL02" ANALYTE="U238" YEAR=10000 MEDIA="SWAT" VALUES=11 12
DATA LOCATION="EL02" ANALYTE="Tc99" YEAR=10000 MEDIA="SWAT" VALUES=11.1 10
DATA LOCATION="EL02" ANALYTE="Tc99" YEAR=10000 MEDIA="PWAT" VALUES=11.2 12
```

9.3.2 END Keyword for FILLECDA

The END keyword signifies the end of all keyword data. Nominally it will be the last keyword in the keyword file. All data in the keyword file after the END keyword will be ignored. The following is this keyword's syntax:

```
END
```

9.3.3 FILE Keyword for FILLECDA

The FILE keyword is used to enter the name of the ESD keyword file. The following is this keyword's syntax:

```
FILE [ESD=“quote 1”]
```

The file name is entered in a quote string, which must be enclosed in double quotation marks. Path names up to 200 characters long are supported. An example of the use of this keyword follows:

```
FILE ESD=“/home/ANALYSIS/practice-median/ESD_practice-median.key”
```

9.3.4 REALIZAT Keyword for FILLECDA

The REALIZATION keyword defines the number of realizations to process. The following is this keyword's syntax:

```
REALIZATION value1
```

This keyword gives the number of realizations of data that will be entered on each DATA card. The number must match exactly with the number of realizations in the ESD keyword file. The following keyword record sets the number of realizations to 10:

```
REALIZAT 10
```

9.3.5 REPORT Keyword for FILLECDA

The REPORT keyword is used to define the name of the output report (log) file. It must be the first keyword entered in the keyword file. The following is this keyword’s syntax:

```
REPORT [“quote”]
```

The name of the report file is entered in a quote string. File names up to 200 characters long are supported, and path names can be included. The following is an example REPORT keyword record:

```
REPORT “/SAC/SystemCodes/Cultural/Test1.rpt”
```

9.3.6 TITLE Keyword for FILLECDA

The TITLE keyword is used to define a single-line problem title. The problem title will be written to output files. If the title is not supplied, then the program will error terminate. The following is this keyword’s syntax:

```
TITLE [“quote”]
```

The title is entered in a quote string, which must be enclosed in double quotation marks. Titles up to 200 characters long are supported. The following example defines a title for a run of the code:

```
TITLE “Example title line for the human impacts code.”
```

9.3.7 USER Keyword for FILLECDA

The USER keyword is used to identify the user of the program. The user name will be written to output files. If the user name is not supplied, then the program will error terminate. The following is this keyword’s syntax:

```
USER [“quote”]
```

The user name is entered in a quote string, which must be enclosed in double quotation marks. User names up to 16 characters long are supported. The following example defines John Q. Public as the user running the code:

```
USER “John Q. Public”
```

10.0 GWGRAB – Groundwater Transport Module Data Extractor

10.1 Overview

The GWGRAB code provide an efficient means for analyzing inputs, outputs, mass balance, and other parameters associated with the Groundwater Transport module of the SAC Rev. 1. GWGRAB can detect the parameters of an assessment, locate all necessary inputs and results in the SAC directory structure, and summarize results in a number of ways that can be imported into spreadsheets or graphing programs.

10.1.1 Location in the Processing Sequence

The GWGRAB reads data written by the groundwater flow and transport module. The GWGRAB code must be run after the CFEST code. GWGRAB can also be used to monitor the progress of CFEST runs.

10.1.2 How the Code Is Invoked

GWGRAB only runs under the Linux operating system. GWGRAB is executed through the following Bourne Shell or C Shell commands:

```
gwgrab-1.exe “ESDKeyfilename” “GWGRABKeyFileName”
```

For the command, gwgrab-1.exe is the name of the executable program, and “ESDKeyfilename” is the name of the ESD keyword file providing overall control to the SAC simulation and “GWGRABKeyFileName” is the name of the GWGRAB control keyword file. If GWGRAB is invoked without two filenames the code will terminate execution after writing an error message to the standard output device. GWGRAB must be invoked from the root directory of a SAC analysis set and the ESD keyword file name must be the local file name (no path information). The keyword file name can include path information.

10.1.3 Memory Requirements

The GWGRAB program can require a large amount of memory. Dynamic memory allocation is used and memory use depends on the number of realizations and number of analytes. A typical run with 13 analytes and 100 realizations required on the order of 65 megabytes of memory.

10.2 File Definitions

The GWGRAB program requires two input files to control data extraction. The program also writes two or more output files for every run.

10.2.1 Input Files

The GWGRAB program requires two input files. The first file is the environmental settings definition (ESD) keyword file providing overall control for the SAC problem. This file contains information used by GWGRAB such as the number of realizations and analyte definitions. The second input file is the

GWGRAB control keyword file that gives specific information concerning the particular data to be extracted and reported. This file may be named with any name so that any number of control keyword files may be created at any one time. However, only one control file may be used per run of the GWGRAB program

10.2.2 Output Files

Output files include a log file and up to five additional files. The log file will be named by the same name as the GWGRAB control keyword file with the extension of ".log." The other five files, if written, contain analyte influx data, analyte storage data, analyte outgoing flux data, decay information, and error messages.

10.3 Keyword Descriptions for the GWGRAB Control Keyword File

All user instructions regarding data extraction are defined in the GWGRAB keyword control file. Keywords may be listed in any order. However, all information after the END keyword will be ignored. Therefore, it is necessary that the END keyword be the last keyword listed.

10.3.1 ANALYTE Keyword for GWGRAB

The ANALYTE keyword is used to define the analyte or analytes to process. The following is this keyword's syntax:

```
ANALYTE [ID="quote1" {"quote2"} … {"quoteN"} |ALL]
```

Several ANALYTE keywords may be entered or a single ANALYTE keyword with more than one ID can be used. If the modifier ALL is used then the report is generated for all analytes defined in the ESD keyword file. If the modifier ID is used it must be followed by one or more quote strings containing analyte ID's. Each analyte ID is limited to six characters in length and must identify one of the analytes defined in the ESD control file. The following set of three analyte keywords have the same effect as the single fourth keyword.

```
ANALYTE ID= "H3"
ANALYTE ID= "Sr90"
ANALYTE ID= "Cs137"
ANALYTE ID= "H3" "Cs137"  "Sr90"
```

10.3.2 END Keyword for GWGRAB

The END keyword signifies the end of all keyword data. Nominally it will be the last keyword in the keyword file. All data in the keyword file after the END keyword will be ignored. The following is this keyword's syntax:

```
END
```

10.3.3 EXEDIR Keyword for GWGRAB

The EXEDIR keyword identifies the directory location for program executable files. The following is this keyword's syntax:

```
EXEDIR [“quote”]
```

The EXEDIR keyword is used to identify the locations where programs reside that get executed through calls from GWGRAB. The programs needed are part of the CFEST suite of codes. The following is a sample keyword:

```
EXEDIR “/home/CODES/bin”
```

10.3.4 EXTRACT Keyword for GWGRAB

The EXTRACT keyword defines the types of data to be extracted. Several EXTRACT keywords with different modifiers may be included in one keyword file but a single EXTRACT keyword with more than one modifier is also acceptable. The following is this keyword's syntax:

```
EXTRACT [INQUIRE | BALANCE | CHECK]
```

The modifiers for the EXTRACT keyword are defined in Table 10.1.

Table 10.1 Modifiers Associated with the EXTRACT keyword for GWGRAB

Modifier	Description
INQUIRE	Returns information about the SAC run such as which analytes and how many realizations were run.
BALANCE	Calculates the amount of analyte (kg) in the groundwater domain at the SAC-wide balance times (use EXTRACT INQUIRE to determine balance times).
CHECK	Reports the number of completed CFEST time steps for one or more CFEST runs.

10.3.5 PATH Keyword for GWGRAB

The PATH keyword defines the path to the directory where the extracted results are stored. The path must be entered in quotes. The following is this keyword's syntax:

```
PATH  “path name”
```

An example PATH keyword record is the following:

```
PATH "/home/ANALYSIS2/Initial3/gwgrab/"
```

10.3.6 REALIZATION Keyword for GWGRAB

The REALIZATION keyword defines which realization or realizations to report. Several REALIZATION keywords may be entered or a single REALIZATION keyword with more than one numbers may be used. Realization numbers must be entered as positive integers. The following is this keyword's syntax:

```
REALIZATION [N1 {N2} … {Nk}|ALL]
```

If the ALL modifier is entered then data will extracted for all realizations defined in the ESD keyword file. If the ALL modifier is absent then data will be extracted for only the specified realizations. The following two keywords select extractions for realizations 1, 2, 3, 10, 27, and 45

```
REALIZAT 45 2 3
REALIZAT 1 10 27
```

10.3.7 OUTPUT Keyword for GWGRAB

The OUTPUT keyword defines how to format the output data. The following is this keyword's syntax:

```
OUTPUT [RAW]
```

The RAW modifier must be present. Raw data are output on the time steps used in CFEST. An example use of this keyword is the following:

```
OUTPUT RAW
```

10.3.8 TIME Keyword for GWGRAB

The TIME Keyword defines how to present the data with respect to time. The following is this keyword's syntax:

```
TIME [ANNUAL|TOTAL]
```

If the modifier ANNUAL is used then data are summed and registered to calendar years. If the modifier TOTAL is used then data are summed for the entire simulation period. Two examples of this keyword record are the following:

```
TIME ANNUAL
TIME TOTAL
```

10.4 Example Cases for GWGRAB

Two examples uses of GWGRAB are provided in the following sections.

10.4.1 GWGRAB Example – Check Run Status for CFEST

GWGRAB can be used to check the status of the Groundwater Transport (CFEST) module during an active SAC assessment. This task is accomplished by use of the EXTRACT CHECK keyword in the GWGRAB keyword control file. An example of the GWGRAB keyword control file is provided in Table 10.2.

Table 10.2 GWGRAB Example Keyword File - Check Run Status

```
PATH "/home/ANALYSIS2/Initial3/gwgrab/" !Where to put the results
EXEDIR "/home/CODES/bin/" !Where the SAC executables are
EXTRACT CHECK  !Report completed CFEST time steps.
ANALYTE ALL    !Report all analytes.
REALIZATION ALL   !Report all realizations.
TIME ANNUAL       !TIME keyword is not applicable but must be !included
OUTPUT RAW        !Report raw data for each realization.
END
```

The keyword control in Table 10.2 results in the following type of output. Only an excerpt from the output is provided here in order to conserve document space.

```
------------------------------------------------------------------------
 Current CFEST LPROG3 Run Status (Completed Time Steps)
------------------------------------------------------------------------
Analyte     1    2    3    4    5    6    7    8    9   10
------------------------------------------------------------------------
Tc99     1163 1163 1163 1163 1163 1163 1163 1163 1163 1163
out of   1163 1163 1163 1163 1163 1163 1163 1163 1163 1163

H3       1163 1163 1163 1163 1163 1163 1163 1163 1163 1163
out of   1163 1163 1163 1163 1163 1163 1163 1163 1163 1163
```

10.4.2 GWGRAB Example – Extract CFEST Results

GWGRAB can be used to extract mass balance information from CFEST runs. This task is accomplished by use of the EXTRACT BALANCE keyword in the GWGRAB keyword control file. Extracted data are placed into five output file types: influx, storage, outflux, decay, and error. The first four files help examine mass conservation in the runs; the change in storage equals influx minus outflux minus radioactive decay, whereas the error file reflects the accuracy of the calculation by displaying the error ratio (1.00 if mass is 100% conserved). All calculations are done at the end of each CFEST time step. An example of the GWGRAB keyword control file is provided in Table 10.3.

Table 10.3 GWGRAB Example Keyword File – Extract CFEST Results

```
PATH "/home/ANALYSIS2/Initial3/gwgrab/" !Where to put extracted results
EXEDIR "/home/CODES/bin/" !Where the SAC executables are
EXTRACT CHECK        !Report completed CFEST time steps.
ANALYTE “H3”         !Report Tritium only.
REALIZATION 1 2 3 4  !Report realizations 1-4 only.
TIME ANNUAL          !Mass balance reported at predefined times
OUTPUT RAW           !Report raw data for each realization.
END
```

The extraction defined in Table 10.3 results in the following types of output files. Only the first few lines from each file are shown here.

```
#SAC Groundwater Analyte Influx Data, by Realization (kg)
#SAC Data Extracted          9/10/02    12:28:16
#SAC Problem :        SAC Rev. 0 Initial3 Assessment
 Simulated Time (d)     Date & Time    Analyte ID      1          2          3           4
      1.83E+02             7/1/44 15:00      H3       0.00E+00  0.00E+00  0.00E+00   0.00E+00
      3.65E+02            12/31/44 6:00      H3       0.00E+00  0.00E+00  9.03E-19   0.00E+00
      5.48E+02             7/1/45 21:00      H3        1.38E-06   1.39E-06  1.11E-06    1.11E-06
```

```
#SAC Groundwater Analyte Outflux Data, by Realization (kg)
#SAC Data
Extracted                      9/10/02   12:28:16
#SAC Problem :       SAC Rev. 0 Initial3 Assessment
 Simulated Time (d)     Date & Time    Analyte ID       1           2           3           4
      1.83E+02          7/1/44 15:00        H3       0.00E+00    0.00E+00    0.00E+00    0.00E+00
      3.65E+02         12/31/44 6:00        H3       0.00E+00    0.00E+00    9.42E-30    0.00E+00
      5.48E+02          7/1/45 21:00        H3      -4.51E-17   -1.45E-16   -4.40E-18    3.76E-18
```

```
#SAC Groundwater Analyte Decay Data, by Realization (kg)
#SAC Data Extracted          9/10/02       12:28:16
#SAC Problem :        SAC Rev. 0 Initial3 Assessment
 Simulated Time (d)     Date & Time        Analyte ID       1          2          3           4
      1.83E+02            7/1/44 15:00         H3        0.00E+00  0.00E+00  0.00E+00   0.00E+00
      3.65E+02           12/31/44 6:00         H3        0.00E+00  0.00E+00  0.00E+00   0.00E+00
      5.48E+02            7/1/45 21:00         H3        0.00E+00  0.00E+00  2.51E-20   0.00E+00
      7.31E+02           12/31/45 12:00        H3         3.90E-08  3.91E-08  3.12E-08    3.12E-08
```

```
#SAC Groundwater Analyte Balance Data, by Realization (kg)
#SAC Data Extracted          9/10/02      12:28:16
#SAC Problem :         SAC Rev. 0 Initial3 Assessment
  Simulated Time (d)     Date & Time       Analyte ID       1          2          3           4
      1.83E+02            7/1/44 15:00         H3        0.00E+00  0.00E+00  0.00E+00   0.00E+00
      3.65E+02           12/31/44 6:00         H3        0.00E+00  0.00E+00  9.03E-19   0.00E+00
      5.48E+02            7/1/45 21:00         H3         1.38E-06  1.39E-06  1.11E-06    1.11E-06
```

```
#SAC Groundwater Analyte Mass Balance Error Fraction, by Realization
#SAC Data Extracted          9/10/02       12:28:16
#SAC Problem :         SAC Rev. 0 Initial3 Assessment
  Simulated Time (d)     Date & Time       Analyte ID       1          2          3           4
      1.83E+02            7/1/44 15:00         H3        1.00E+00  1.00E+00  1.00E+00   1.00E+00
      3.65E+02           12/31/44 6:00         H3        1.00E+00  1.00E+00  1.00E+00   1.00E+00
      5.48E+02            7/1/45 21:00         H3        1.00E+00  1.00E+00  1.00E+00   1.00E+00
```

11.0 HIGHDOSE – Dose Extraction For ECEM

11.1 Overview

The HIGHDOSE code is designed to extract the largest impact calculated by the ECEM (see Eslinger et al. [2004b], Volume 2, Section 3.0) code over a suite of locations at each of several time steps. For example, it can determine the species with the highest dose along the riparian zone at a set of times.

11.1.1 Location in the Processing Sequence

The HIGHDOSE code must follow a run of the ECEM code.

11.1.2 How the Code Is Invoked

HIGHDOSE can run under either the Windows or the Linux operating system. Under the Windows operating system (Releases 98, NT, 2000, or XP), HIGHDOSE executes in a DOS box. A run of HIGHDOSE is initiated by entering the following command line:

```
   HIGHDOSE “Keyfilename”
```

Under the Linux operating system HIGHDOSE is executed through any of the following Bourne Shell or C Shell commands:

```
   highdose-1.exe “Keyfilename”
```

For these commands, HIGHDOSE.EXE or highdose.exe is the name of the executable program and “Keyfilename” is the name of a control keyword file. Both the name of the executable program and the keyword file may contain path information. If HIGHDOSE is invoked without entering the name of the keyword file, then the code will prompt the user for the file name. The keyword file, which should be prepared using an editor that can handle ASCII files without leaving embedded control codes, contains text control information describing the run. If HIGHDOSE cannot open the keyword file, then the code will terminate execution after writing an error message to the standard output device.

11.1.3 Memory Requirements

The HIGHDOSE program has minimal memory requirements. A run of the code to extract values for all upland locations required less than 2 MB of memory.

11.2 File Definitions

The HIGHDOSE program reads three input files and writes to two output files. These files are described in the following paragraphs.

11.2.1 Input Files

The input files for the HIGHDOSE program are a control keyword file, a species definition file, and a details result file output by the ECEM code.

11.2.1.1 HIGHDOSE Keyword File

The HIGHDOSE keyword file contains control information. An example file is provided in Table 11.1. Detailed definitions of the keywords are provided in Section 11.3.

Table 11.1 Example Keyword File for HIGHDOSE

```
! Keyword file for the FillECDA program
REPORT "Test.Rpt"
TITLE "Testing of the HighDose Code Using a ECEM Rev. 1 run"
USER "Paul W. Eslinger"
FILE SPECIES "Species.csv"
FILE ECEM "Ecem_Det.Csv"
FILE RESULTS "Test.Csv"
REALIZATION 2
MEMORY TABLEROW=3 REALIZATION=25 RECORDS=1000
!VERBOSE
SOLUTION "SUMRAD"
ANALYTE "-Rads-"
SOIL "SOSW"
! The following keywords allow multiple choices
LOCATION "TH0001" "TH0003" "TH0015" "TH0017"
          "TH0002" "TH0005" "TH0006"
TIME 1950  1955  1960  1965
END
```

11.2.1.2 Species Definition File

The HIGHDOSE program requires a species definition file. This file contains the species type, species ID, and species name for every species for which data are provided in the ECEM (see Eslinger et al. [2004b], Volume 2, Section 3.0) results file. This information is required, but is used mainly for labeling purposes. This file is a text file written using a comma separated format. An example file is provided in Table 11.2.

Table 11.2 Example Species Definition File for HIGHDOSE

```
"TA","AMCOOT","American coot"
"TA","AMCOUP","American coot - upland"
"TA","EGGS","Chicken eggs - upland"
"TA","BEAVER","Beaver"
"QA","CARP","Carp"
"TA","COYOTE","Coyote"
"QA","DAPMAG","Daphnia magna"
"TP","DENSDG","Dense sedge"
"TP","DENSUP","Dense sedge - upland"
```

```
"TP","FUNGI","Fungi"
"TP","MULBRY","Mulberry"
"TP","MULBUP","Mulberry - upland"
"TA","MULDER","Mule deer"
"TP","GRAIN","Generic grain species"
"TP","POTATO","Potato"
"TA","MILKCW","Milk cow"
"QP","PERPHY","Periphyton"
"QP","PHYPLK","Phytoplankton"
```

11.2.1.3 ECEM Results File

The third input file for the HIGHDOSE program is nominally generated with a run of the ECEM code. This file contains data output from ECEM under the control of the DETAILS keyword in ECEM.

11.2.2 Output Files

The HIGHDOSE program writes two output files. One file is a report file that contains text information about the run of the code. It is not described further here. The other output file is the results file containing the extracted information. The first few lines from example results file is provided in Table 11.3.

Table 11.3 Example Results File from the HIGHDOSE Code

```
"Code Name:","HighDose"
"Code Version:","1.0.A.02"
"Code Date:","31 Jul 2002"
"Run ID:","20040127124827"
"Run Title:","ECEM Run - Revised Draft Hanford Solid Waste EIS - Background Plus Hanfo"
"User Name:","Paul W. Eslinger"

"Realization 1 is requested."

"Results for location ID = NAPLSH year = 2500 solution = SUMRAD and analyte type = -Rads-"
"Terrestrial Plants"," ","Terrestrial Animals"," ","Aquatic Plants"," ","Aquatic Animals"
"Specie","Dose","Specie","Dose","Specie","Dose","Specie","Dose"
"TULE  ", 9.754E-11,"CLFSWL", 1.749E-08,"PERPHY", 7.722E-09,"MAYFLY", 3.565E-08
"BLCTWD", 8.809E-11,"AMCOOT", 7.938E-09,"WMLFOL", 5.737E-09,"CRYFSH", 1.168E-08
"MULBRY", 8.809E-11,"BUFLHD", 5.317E-09,"PHYPLK", 6.430E-12,"CPBLSN", 7.990E-09
"CYLCRS", 4.674E-11,"COMSNP", 5.012E-09,"      ", 0.000E+00,"SALMJV", 5.533E-09
"RCANGS", 3.362E-11,"RACOON", 3.198E-09,"      ", 0.000E+00,"RTRTJV", 5.533E-09

"Results for location ID = NAPLSH year = 10000 solution = SUMRAD and analyte type = -Rads-"
"Terrestrial Plants"," ","Terrestrial Animals"," ","Aquatic Plants"," ","Aquatic Animals"
"Specie","Dose","Specie","Dose","Specie","Dose","Specie","Dose"
"TULE  ", 5.840E-10,"RACOON", 2.118E-06,"WMLFOL", 5.992E-05,"WOTDTP", 9.941E-05
"BLCTWD", 5.740E-10,"AMCOOT", 1.292E-06,"PERPHY", 3.261E-06,"MAYFLY", 4.508E-06
```

```
"MULBRY", 5.740E-10,"MALLRD", 2.780E-07,"PHYPLK", 1.753E-06,"DAPMAG", 7.106E-07
"CYLCRS", 3.034E-10,"AMKSTL", 2.736E-07,"      ", 0.000E+00,"CPBLSN", 5.668E-07
"RUSHES", 2.979E-10,"WEASEL", 2.349E-07,"      ", 0.000E+00,"HYALLE", 2.558E-07
```

The most typical method for looking at the results of the HIGHDOSE code is to reformat the comma-separated results into a table. An example of such a result is provided in Table 11.4.

Table 11.4 Reformatted Output From the HIGHDOSE Code

Results for location ID = NAPLSH year = 2500 solution = SUMRAD and analyte type = -Rads-							
Terrestrial Plants		**Terrestrial Animals**		**Aquatic Plants**		**Aquatic Animals**	
Specie	**Dose**	**Specie**	**Dose**	**Specie**	**Dose**	**Specie**	**Dose**
TULE	9.75E-11	CLFSWL	1.75E-08	PERPHY	7.72E-09	MAYFLY	3.57E-08
BLCTWD	8.81E-11	AMCOOT	7.94E-09	WMLFOL	5.74E-09	CRYFSH	1.17E-08
MULBRY	8.81E-11	BUFLHD	5.32E-09	PHYPLK	6.43E-12	CPBLSN	7.99E-09
CYLCRS	4.67E-11	COMSNP	5.01E-09		0.00E+00	SALMJV	5.53E-09
RCANGS	3.36E-11	RACOON	3.20E-09		0.00E+00	RTRTJV	5.53E-09

11.3 Keyword Definitions for HIGHDOSE

In general, the keywords for the HIGHDOSE code can be entered in any order. The following restrictions apply on keyword order:

- The REPORT keyword must be the first keyword in the file.
- The END keyword must be the last keyword in the file.

11.3.1 ANALYTE Keyword for HIGHDOSE

The ANALYTE keyword is used to define the analyte (or analytes) for which data will be processed. The following is this keyword’s syntax:

```
ANALYTE [“quote 1”]
```

The single quote string must either be a single analyte ID from the set of the analytes in the ESD keyword file, or the string “-Rads-” if the combined dose over multiple radioactive analytes is desired. Two examples of this keyword are the following:

```
ANALYTE “C14”
ANALYTE “-Rads-”
```

11.3.2 END Keyword for HIGHDOSE

The END keyword signifies the end of all keyword data. All data in the keyword file after the END keyword will be ignored. The following is this keyword’s syntax:

```
END
```

11.3.3 FILE Keyword for HIGHDOSE

The FILE keyword is used to enter the names of all input and output files except for the report file. The following is this keyword's syntax:

```
   FILE [modifier1="quote1"] {modifier2="quote2"} {modifier3="quote3"}
```

The file names are entered in quote strings, which must be enclosed in double quotes. Path names up to 200 characters long are supported. The file name associated with a modifier must be entered before the next modifier is entered. At least one FILE keyword is required for every run of the code.

The name of the ECEM details file is associated with the modifier ECEM. The name of the input file defining ecological species is associated with the modifier SPECIES. The name of the output file from HIGHDOSE is associated with the modifier RESULTS. Example file keywords that define these three files are the following:

```
   FILE SPECIES "Species.csv"
   FILE ECEM "ECEM_SWEIS_det_Bg.Csv"
   FILE RESULTS "High_Rads_Bg.Csv"
```

11.3.4 LOCATION Keyword for HIGHDOSE

The LOCATION keyword is used to define the set of locations for which data will be processed. The following is this keyword's syntax:

```
   LOCATION [“quote 1”] {“quote2” … “quoten”]
```

One or more quote strings must contain valid location ID's from locations specified in the ESD keyword file. Data for a given time step is scanned over the entire set of specified locations. An example LOCATION keyword defining ten locations is the following:

```
   LOCATION
      "UH0001" "UH0002" "UH0003" "UH0004" "UH0005"
      "UH0006" "UH0007" "UH0008" "UH0009" "UH0010"
```

11.3.5 MEMORY Keyword for HIGHDOSE

The MEMORY keyword is used to define some variables that determine the amount of memory used in the code. The following is this keyword's syntax:

```
   MEMORY [TABLEROW=N1] [REALIZAT=N2] [RECORDS=N3]
```

The modifiers associated with the MEMORY keyword are described in Table 11.5.

Table 11.5 Modifiers Associated with the MEMORY Keyword in HIGHDOSE

Modifier	**Description**
TABLEROW	The numerical value associated with the TABLEROW modifier defines the number of rows to use in the output table. A value of 3 to 5 is suggested.
REALIZAT	The numerical value associated with the REALIZAT modifier defines the number of realizations used in the run of ECEM that produced the detailed data file being analyzed
RECORDS	The numerical value associated with the RECORDS modifier defines the number of records that may be matched in the detailed data file for a single year and location Combination. Typically this value can be set to the number of species.

An example keyword where the user specifies five rows for the output table for a deterministic run of ECEM that utilized 100 species is the following:

```
MEMORY TABLEROW=5 REALIZATION=1 RECORDS=100
```

11.3.6 REALIZATION Keyword for HIGHDOSE

The REALIZATION keyword defines the single realization for which data are to be extracted. The following is this keyword's syntax:

```
REALIZATION value1
```

The integer value1 has a minimum value of 1 and a maximum of the number of realizations performed by the ECEM code. The following keyword record extracts data for realization number 10:

```
REALIZAT 10
```

11.3.7 REPORT Keyword for HIGHDOSE

The REPORT keyword is used to define the name of the output report (log) file. It must be the first keyword entered in the keyword file. The following is this keyword's syntax:

```
REPORT [“quote”]
```

The name of the report file is entered in a quote string. File names up to 200 characters long are supported, and path names can be included. The following is an example REPORT keyword record:

```
REPORT “/SAC/C/Test.rpt”
```

11.3.8 SOIL Keyword for HIGHDOSE

The SOIL keyword is used to the solution type for which data will be processed. The following is this keyword's syntax:

```
SOIL [“quote 1”]
```

The single quote string is case sensitive, six characters in length and must be one of the following:

- NONE = Applies to all aquatic locations
- SORP = Riparian locations
- SODR = Dry (non irrigated) upland locations
- SOGW = Groundwater irrigated upland locations
- SOSW = Surface water irrigated upland locations

An example of this keyword is the following:

```
SOLUTION “BURDEN”
```

11.3.9 SOLUTION Keyword for HIGHDOSE

The SOLUTION is used to the solution type for which data will be processed. The following is this keyword’s syntax:

```
SOLUTION [“quote 1”]
```

The single quote string is case sensitive, six characters in length and must be one of the following:

- BURDEN = Body burdens
- DOSRAD = Radioactive dose
- BMTISS = Tissue benchmark solutions
- SUMRAD = Sum of radioactive dose
- CONCEN = Concentrations

An example of this keyword is the following:

```
SOLUTION “BURDEN”
```

11.3.10 TIME Keyword for HIGHDOSE

The TIME keyword identifies the times at which the calculations are to be performed. The following is this keyword’s syntax:

```
TIME [T1] {T2} … {Tn}
```

The numerical entries T1, T2, …, Tn are the times (whole number years) when outputs are desired. These times must be a subset of the times at which environmental data were computed and stored by the inventory, release, and transport modules. Only one TIME card should be entered. The following is an example TIME keyword that requests output for the three years 2020, 2075, and 3014:

```
TIME 2020 2075 3014
```

11.3.11 TITLE Keyword for HIGHDOSE

The TITLE keyword is used to define a single-line problem title. The problem title will be written to output files. The program will error terminate if the title is not supplied. The following is this keyword's syntax:

```
TITLE [“quote”]
```

The title is entered in a quote string, which must be enclosed in double quotation marks. Titles up to 200 characters long are supported. The following example defines a title for a run of the code:

```
TITLE “Example title line for the INVSUM code.”
```

11.3.12 USER Keyword for HIGHDOSE

The USER keyword is used to identify the user of the program. The user name will be written to output files. The program will error terminate if the user name is not supplied. The following is this keyword's syntax:

```
USER [“quote”]
```

The user name is entered in a quote string, which must be enclosed in double quotation marks. User names up to 16 characters long are supported. The following example defines John Q. Public as the user running the code:

```
USER “John Q. Public”
```

11.3.13 VERBOSE Keyword for HIGHDOSE

The presence of the optional VERBOSE keyword initiates additional output to the report file. The following is this keyword's syntax:

```
VERBOSE
```

12.0 HIGHIMPACT – Maximum Impact Extraction For HUMAN

12.1 Overview

The HIGHIMPACT code is designed to extract the largest impact result calculated by the HUMAN code over a suite of locations at each of several time steps. For example, it can provide the largest dose to an individual outside the Hanford core zone as a function of time.

12.1.1 Location in the Processing Sequence

The HIGHIMPACT code must follow a run of the HUMAN code.

12.1.2 How the Code Is Invoked

HIGHIMPACT can run under either the Windows or the Linux operating system. Under the Windows operating system (Releases 98, NT, 2000, or XP), HIGHIMPACT executes in a DOS box. A run of HIGHIMPACT is initiated by entering the following command line:

```
HIGHIMPACT “Keyfilename”
```

Under the Linux operating system HIGHIMPACT is executed through any of the following Bourne Shell or C Shell commands:

```
highimpact-1.exe “Keyfilename”
```

For these commands, HIGHIMPACT .EXE or highimpact-1.exe is the name of the executable program and “Keyfilename” is the name of a control keyword file. Both the name of the executable program and the keyword file may contain path information. If HIGHIMPACT is invoked without entering the name of the keyword file, then the code will prompt the user for the file name. The keyword file, which should be prepared using an editor that can handle ASCII files without leaving embedded control codes, contains text control information describing the run. If HIGHIMPACT cannot open the keyword file, then the code will terminate execution after writing an error message to the standard output device.

12.1.3 Memory Requirements

The HIGHIMPACT program has minimal memory requirements. A run of the code to extract values for all upland locations required less than 2 MB of memory.

12.2 File Definitions

The HIGHIMPACT program reads two input files and writes two output files. These files are described in the following sections.

12.2.1 Input Files

The HIGHIMPACT program reads a control keyword file and a impact details file written by the HUMAN code. The HIGHIMPACT keyword file contains control information. An example file is provided in Table 12.1. Detailed definitions of the keywords are provided in Section 12.3.

Table 12.1 Example Keyword File for the HIGHIMPACT Program

```
! Keyword file for HIGHIMPACT
REPORT "Test.Rpt"
TITLE "Testing of the HighImpact Code Using a HUMAN Rev. 1 run"
USER "Paul W. Eslinger"
FILE HUMAN "Farmer_CA1_median_Dtl.csv"
FILE RESULTS "Test.Csv"
REALIZAT HUMAN=1 SINGLE=1
! Locations can be multiply defined
LOC_ID "UH1082" "UH1083"
! Times can be multiply defined
TIME 1990  1992
! The following solutions must be uniquely defined
S_TYPE "GWAT"
ANA_ID "C14"
R_TYPE "CON"
VERBOSE
END
```

12.2.2 Output Files

The HIGHIMPACT program writes two output files. One file is a report file that contains text information about the run of the code. It is not described further here. The other output file is the results file containing the information about the highest impact over a set of locations for every year. An example results file is provided in Table 12.2.

Table 12.2 Example Results File from the HIGHIMPACT Code

```
"Code Name:","HighImpact"
"Code Version:","1.00.002"
"Code Date:","30 Jul 2004"
"Run ID:","20040730111054"
"Run Title:","Maximum dose to a farmer outside the core zone"
"User Name:","Paul W. Eslinger"
"Solution type:","SINGLE = 1","COMBIN","RAD","CB"
1945,"UH2709", 9.97442E-06
1950,"UH2709", 2.76643E-03
1955,"UH0858", 5.56114E-01
1960,"UH0888", 9.09807E-01
1965,"UH1643", 3.56537E+00
1970,"UH1644", 4.35403E+00
1975,"UH1645", 3.20452E+00
1980,"UH1645", 1.97369E+00
1985,"UH1644", 1.01818E+00
```

```
1990,"UH0873", 9.39636E-01
1991,"UH0873", 9.11681E-01
1992,"UH0873", 8.80937E-01
1993,"UH0873", 8.48686E-01
1994,"UH0873", 8.15049E-01
1995,"UH0873", 7.80510E-01
1996,"UH0873", 7.45544E-01
1997,"UH0873", 7.10520E-01
1998,"UH0873", 6.73069E-01
1999,"UH0873", 6.36624E-01
2000,"UH0873", 6.02480E-01
```

12.3 Keyword Definitions for the HIGHIMPACT Code

In general, the keywords for the HIGHIMPACT code can be entered in any order. The following restrictions apply on keyword order:

- The REPORT keyword must be the first keyword in the file.
- The END keyword must be the last keyword in the file.

The keyword selections require the user to specify a set of possible values for the first five data values on each line in the details file written by the HUMAN code. The relative position of the values in the details file and the associated keyword is the following:

1. TIME – Multiple time definitions are allowed on a single TIME keyword.
2. LOCATION – Multiple location definitions are allowed on a single LOCATION keyword.
3. ANA_ID – A single solution is required for this keyword.
4. R_TYPE – A single solution is required for this keyword.
5. S_TYPE – A single solution is required for this keyword.

Values for the data in positions 3, 4, and 5 depend on the specified solutions. An easy way to choose the correct combination is to examine the results data file that is to be processed.

12.3.1 ANA_ID Keyword for HIGHIMPACT

The ANA_ID keyword is used to define the analyte (or analytes) for which data will be processed. The following is this keyword's syntax

```
ANA_ID [“quote 1”]
```

The single quote string must either be a single analyte ID from the set of the analytes in the ESD keyword file, or the string “-Rads-” if the combined dose over multiple radioactive analytes is desired. Two examples of this keyword are the following:

```
ANA_ID “C14”
ANA_ID “-Rads-”
```

12.3.2 END Keyword for HIGHIMPACT

The END keyword signifies the end of all keyword data. All data in the keyword file after the END keyword will be ignored. The following is this keyword's syntax:

```
END
```

12.3.3 FILE Keyword for HIGHIMPACT

The FILE keyword is used to enter the names of all input and output files except for the report file. The following is this keyword's syntax:

```
FILE [modifier1="quote1"] {modifier2="quote2"}
```

The file names are entered in quote strings, which must be enclosed in double quotes. Path names up to 200 characters long are supported. The file name associated with a modifier must be entered before the next modifier is entered. At least one FILE keyword is required for every run of the code.

The name of the human details file is associated with the modifier HUMAN. The name of the output file from HIGHIMPACT is associated with the modifier RESULTS. Example file keywords that define these two files are the following:

```
FILE HUMAN "Farmer_CA1_median_Dtl.csv"
FILE RESULTS "Farmer_High.csv"
```

12.3.4 LOC_ID Keyword for HIGHIMPACT

The LOC_ID keyword is used to define the set of locations for which data will be processed. The following is this keyword's syntax:

```
LOC_ID [“quote 1”] {“quote2” … “quoten”]
```

One or more quote strings must contain valid location ID's from locations specified in the ESD keyword file. Data for a given time step is scanned over the entire set of specified locations. An example LOC_ID keyword defining ten locations is the following:

```
LOC_ID
   "UH0001" "UH0002" "UH0003" "UH0004" "UH0005"
   "UH0006" "UH0007" "UH0008" "UH0009" "UH0010"
```

12.3.5 REALIZATION Keyword for HIGHIMPACT

The REALIZAT keyword is used to define the realizations which data will be processed. The following is this keyword's syntax:

```
REALIZAT [HUMAN=N1] [MAXIMUM|MEAN|SINGLE=N2]
```

The modifiers associated with the REALIZAT keyword are described in Table 12.3. Only one of the MAXIMUM, MEAN, or SINGLE modifiers are allowed during a run of the code.

Table 12.3 Modifiers Associated with the REALIZAT Keyword in HIGHIMPACT

Modifier	Description
HUMAN	The numerical value associated with the HUMAN modifier defines the number of realizations contained in the details file written by the HUMAN code.
MAXIMUM	This modifier indicates that the maximum over all realizations will be used in determining the highest impact at each location.
MEAN	This modifier indicates that the arithmetic mean over all realizations will be used in determining the highest impact at each location.
SINGLE	This modifier indicates that data from a single realization will be used in determining the highest impact at each location. The realization number to use is provided in the numerical value associated with the SINGLE modifier.

An example keyword where the user specifies using the arithmetic mean for a data file containing 100 realizations of data is the following:

```
REALIZAT HUMAN=100 MEAN
```

An example keyword where the user specifies using the data for realization 23 from a data file containing 100 realizations of data is the following:

```
REALIZAT HUMAN=100 SINGLE=23
```

12.3.6 REPORT Keyword for HIGHIMPACT

The REPORT keyword is used to define the name of the output report (log) file. It must be the first keyword entered in the keyword file. The following is this keyword's syntax:

```
REPORT [“quote”]
```

The name of the report file is entered in a quote string. File names up to 200 characters long are supported, and path names can be included. The following is an example REPORT keyword record:

```
REPORT “/SAC/C/Test.rpt”
```

12.3.7 R_TYPE Keyword for HIGHIMPACT

The R_TYPE keyword is used to partially define the solution type for which data will be processed. The following is this keyword's syntax:

```
R_TYPE [“quote 1”]
```

The single quote string is case sensitive, six characters in length and must be one of the following:

- CAR = Impacts from carcinogenic analytes
- CON = Concentration data
- HAZ = Impacts from hazardous analytes
- RAD = Impacts from radioactive analytes

An example of this keyword is the following:

```
R_TYPE “CON”
```

12.3.8 S_TYPE Keyword for HIGHIMPACT

The S_TYPE keyword is also used to define the solution type for which data will be processed. The following is this keyword’s syntax:

```
S_TYPE [“quote 1”]
```

The single quote string is case sensitive, two to seven characters in length and must be one of the following:

- AIRC = Air concentrations
- AL = Analyte dose or risk
- BIRD = Food concentration, birds
- CB = Combined dose or risk over several analytes or population dose or risk
- DW = Dose from drinking water
- EGGS = Food concentration, eggs
- EX = External exposure dose or risk
- FISH = Food concentration. fish
- FRUIT = Food concentration, fruit
- GRAIN = Food concentration, grain
- GWAT = Groundwater concentrations
- IG = Ingestion dose or risk
- IH = Inhalation dose or risk
- LEAFVEG = Food concentration, leafy vegetables
- MEAT = Food concentration, meat
- MILK = Food concentration, milk
- PD = Population dose
- PR = Population risk
- ROOTVEG = Food concentration, root vegetables
- SEDI = Sediment concentrations
- SEEP = Seep water concentrations
- SODR = Soil concentrations, no irrigation
- SOGW = Soil concentrations, groundwater irrigated
- SORP = Riparian soil concentrations

- SOSW = Soil concentrations, surface water irrigated
- SWAT = Surface water concentrations

An example of this keyword is the following:

```
S_TYPE “CONCEN”
```

12.3.9 TIME Keyword for HIGHIMPACT

The TIME keyword identifies the times at which the calculations are to be performed. The following is this keyword’s syntax:

```
TIME [T1] {T2} … {Tn}
```

The numerical entries T1, T2, …, Tn are the times (whole number years) when outputs are desired. These times must be a subset of the times at which environmental data were computed and stored by the inventory, release, and transport modules. Only one TIME card should be entered. The following is an example TIME keyword that requests output for the three years 2020, 2075, and 3014:

```
TIME 2020 2075 3014
```

12.3.10 TITLE Keyword for HIGHIMPACT

The TITLE keyword is used to define a single-line problem title. The problem title will be written to output files. The program will error terminate if the title is not supplied. The following is this keyword’s syntax:

```
TITLE [“quote”]
```

The title is entered in a quote string, which must be enclosed in double quotation marks. Titles up to 200 characters long are supported. The following example defines a title for a run of the code:

```
TITLE “Example title line for the INVSUM code.”
```

12.3.11 USER Keyword for HIGHIMPACT

The USER keyword is used to identify the user of the program. The user name will be written to output files. The program will error terminate if the user name is not supplied. The following is this keyword’s syntax:

```
USER [“quote”]
```

The user name is entered in a quote string, which must be enclosed in double quotation marks. User names up to 16 characters long are supported. The following example defines John Q. Public as the user running the code:

```
USER “John Q. Public”
```

12.3.12 VERBOSE Keyword for HIGHIMPACT

The presence of the optional VERBOSE keyword initiates additional output to the report file. The following is this keyword's syntax:

```
VERBOSE
```

13.0 HIGHMEDIA – Extraction of Maximum Media Concentrations

13.1 Overview

The HIGHMEDIA code is designed to extract the maximum media concentrations over a set of specified locations and times for data contained in an ECDA file. For example, it can provide the highest groundwater concentration for a single analyte outside the Hanford core zone as a function of time.

13.1.1 Location in the Processing Sequence

The HIGHMEDIA code reads data from an ECDA file. Thus, the transport codes must have been executed and the media entered in the ECDA files before HIGHMEDIA can be used.

13.1.2 How the Code Is Invoked

HIGHMEDIA can run under either the Windows or the Linux operating system. Under the Windows operating system (Releases 98, NT, 2000, or XP), HIGHMEDIA executes in a DOS box. A run of HIGHMEDIA is initiated by entering the following command line:

```
HIGHMEDIA “Keyfilename”
```

Under the Linux operating system HIGHMEDIA is executed through any of the following Bourne Shell or C Shell commands:

```
highmedia-1.exe “Keyfilename”
```

For these commands, HIGHMEDIA.EXE or highmedia-1.exe is the name of the executable program and “Keyfilename” is the name of a control keyword file. Both the name of the executable program and the keyword file may contain path information. If HIGHMEDIA is invoked without entering the name of the keyword file, then the code will prompt the user for the file name. The keyword file, which should be prepared using an editor that can handle ASCII files without leaving embedded control codes, contains text control information describing the run. If HIGHMEDIA cannot open the keyword file, then the code will terminate execution after writing an error message to the standard output device.

13.1.3 Memory Requirements

The HIGHMEDIA program has minimal memory requirements. A run of the code to extract values for all upland locations required less than 2 MB of memory.

13.2 File Definitions

The HIGHMEDIA program reads two input files and writes two output files. These files are described in the following sections.

13.2.1 Input Files

The HIGHMEDIA program reads a control keyword file and an ECDA file. The HIGHMEDIA keyword file contains control information. An example file is provided in Table 13.1. Detailed definitions of the keywords are provided in Section 13.3.

Table 13.1 Example Keyword File for the HIGHMEDIA Program

```
REPORT "CPO_Alternate_High_GWAT.Rpt"
USER "Paul W. Eslinger"
FILE ESD "\SAC\ANALYSES\CPO_Alternate\ESD_CPO_Alternate.key"
!VERBOSE
ENDINIT

CASE
TITLE "Maximum Concentrations Tc99 at Core Zone Boundary"
FILE RESULTS "CPO_Alternate_GWAT_Tc99_CORE.csv"
ANALYTE "Tc99"
REALIZAT SINGLE=1
MEDIA "GWAT"
UNITS OUTPUT="pCi/L" FACTOR = 1.0E9
TIMES ALL
LOCATION LIST ! Boundary locations for Core Zone
 "UH0722" "UH0724" "UH0725" "UH0736" "UH0739" "UH0748" "UH0749"
 "UH0752" "UH0754" "UH0755" "UH0756" "UH0757" "UH0758" "UH0759"
 "UH0762" "UH0764" "UH0765" "UH0766" "UH0767" "UH0768" "UH0769"
ENDCASE

CASE
TITLE "Maximum Concentrations Tc99 at River Shore Boundary"
FILE RESULTS "CA1_median_GWAT_Tc99_CORE_RIVER.csv"
ANALYTE "Tc99"
REALIZAT SINGLE=1
MEDIA "GWAT"
UNITS OUTPUT="pCi/L" FACTOR = 1.0E9
TIMES ALL
LOCATION LIST ! Boundary locations for the river shore (Hanford Side)
 "RHP001","RHP003","RHP004","RHP005","RHP006","RHP007","RHP008"
 "RHP011","RHP013","RHP014","RHP015","RHP016","RHP017","RHP018"
 "RHP021","RHP023","RHP024","RHP025","RHP026","RHP027","RHP028"
ENDCASE

END
```

13.2.2 Output Files

The HIGHMEDIA program writes two or more output files. One file is a text report file that contains information about the run of the code, including any error messages. The other output file (or files) is the results file containing the information about the highest result over a set of locations for every year. A separate output file is written for every case definition. An example results file is provided in Table 13.2.

Table 13.2 Example Results File from the HIGHMEDIA Code

```
"Code Name:","HighMedia"
"Code Version:","2.00.A.1"
"Code Date:","16 Aug 2004"
"Run ID:","20040817110129"
"Run Title:","HighMedia Using a ECDA Files Rev. 1 run"
"User Name:","Paul W. Eslinger"
"File Name:","H:\CA1_median\ecda_save\Tc99_CA1_median.dat"
"Solution type:","MEDIAN"
"Analyte ID:","Tc99"
"Media ID:","GWAT"
"Year","Solution","Location ID","Easting","Northing"
1990, 4.52490E-07,"UH1230", 5.74522E+05, 1.35992E+05
1991, 3.61323E-07,"UH1230", 5.74522E+05, 1.35992E+05
1992, 2.89056E-07,"UH1230", 5.74522E+05, 1.35992E+05
```

13.3 Keyword Definitions for HIGHMEDIA

The keywords for the HIGHMEDIA code are grouped into an initial set and one or more case groups. The initial set must begin with the REPORT keyword and must end with an ENDINIT keyword. Each analysis case must start with a CASE keyword and end with an ENDCASE keyword. The END keyword must be the last keyword in the file.

13.3.1 Keywords in the Initial Group for HIGHMEDIA

The initial group of keywords starts with the REPORT keyword and ends with the ENDINIT keyword. The run will error terminate if these conditions are not met.

13.3.1.1 ENDINIT Keyword for HIGHMEDIA

The ENDINIT keyword signifies the end of the initial section of the keyword data. The following is this keyword's syntax:

```
ENDINIT
```

13.3.1.2 FILE Keyword for HIGHMEDIA

The FILE keyword is used to enter the names of all input and output files except for the report file. The following is this keyword's syntax in the initial keyword section:

```
FILE [ESD="quote1"]
```

The file name is entered in a quote string, which must be enclosed in double quotes. Path names up to 200 characters long are supported. The name of the ESD control keyword file is associated with the modifier ESD. An example keyword is the following:

```
FILE ESD "C:\SAC\SAC_1\Analysis_2004\CA1_Median\ESD_CA1_median.key"
```

13.3.1.3 REPORT Keyword for HIGHMEDIA

The REPORT keyword is used to define the name of the output report (log) file. It must be the first keyword entered in the keyword file. The following is this keyword's syntax:

```
REPORT [“quote”]
```

The name of the report file is entered in a quote string. File names up to 200 characters long are supported, and path names can be included. An example keyword is the following:

```
REPORT “/SAC/C/Test.rpt”
```

13.3.1.4 USER Keyword for HIGHMEDIA

The USER keyword is used to identify the user of the program. The user name will be written to output files. The program will error terminate if the user name is not supplied. The following is this keyword's syntax:

```
USER [“quote”]
```

The user name is entered in a quote string, which must be enclosed in double quotation marks. User names up to 16 characters long are supported. The following example defines John Q. Public as the user running the code:

```
USER “John Q. Public”
```

13.3.1.5 VERBOSE Keyword for HIGHMEDIA

The presence of the optional VERBOSE keyword initiates additional output to the report file. The following is this keyword's syntax:

```
VERBOSE
```

13.3.2 Keywords for Specific Cases for HIGHMEDIA

The definition of an extraction case starts with the CASE keyword and ends with the ENDCASE keyword. Multiple extraction cases can be defined.

13.3.2.1 ANALYTE Keyword for HIGHMEDIA

The ANAYTE keyword is used to define the analyte for which data will be processed for a single case. The following is this keyword's syntax:

```
ANALYTE [“quote 1”]
```

The single quote string must either be a single analyte ID from the set of the analytes in the ESD keyword file. An example of this keyword utilizing the analyte Tc99 is the following:

```
ANALYTE “Tc99”
```

13.3.2.2 CASE Keyword for HIGHMEDIA

The CASE keyword signifies the beginning of the definition of a specific case of keyword data. The following is this keyword’s syntax:

```
CASE
```

13.3.2.3 ENDCASE Keyword for HIGHMEDIA

The ENDCASE keyword signifies the end of the definition of a specific case of keyword data. The following is this keyword’s syntax:

```
ENDCASE
```

13.3.2.4 FILE Keyword for HIGHMEDIA

The FILE keyword is used to enter the names of all input and output files except for the report file. The following is this keyword’s syntax in a case definition:

```
FILE [RESULTS="quote1"]
```

File names are entered in quote strings, which must be enclosed in double quotes. Path names up to 200 characters long are supported. Exactly one FILE keyword is required for every case in the keyword file. The name of the output file from HIGHMEDIA is associated with the modifier RESULTS. An example keyword that defines a results file is the following:

```
FILE RESULTS "Test.Csv"
```

13.3.2.5 LOCATION Keyword for HIGHMEDIA

The LOCATION keyword is used to define the set of locations for which data will be processed for a single case. TThe following is this keyword’s syntax:

```
LOCATION [ALL| LIST [“quote 1”] {“quote2” … “quoten”}]
```

All locations in the ESD keyword file are used when the modifer ALL is present. A list of specific sites can be defined by entering the modifier LIST and a set of quote strings identifying the site ID’s of interest. The specific locations entered must be defined in the ESD keyword file. All locations are included in the first example shown below. The second example defines a list of six specific sites.

```
LOCATION ALL
LOCATION LIST "UH1000" "UH1001" "UH1003" "UH1004" "UH1005" "UH1230"
```

13.3.2.6 MEDIA Keyword for HIGHMEDIA

The MEDIA keyword is used to define the media for which data will be processed for a single case. The following is this keyword's syntax:

```
MEDIA [“quote 1”]
```

The single quote string is case sensitive, four characters in length, and must be one of the following:

- GWAT: concentrations in groundwater (Ci/m^3 or kg/m^3)
- SEEP: concentrations in seep water (Ci/m^3 or kg/m^3)
- SWAT: concentrations in surface water (river) (Ci/m^3 or kg/m^3)
- PWAT: concentrations in river bottom pore water (Ci/m^3 or kg/m^3)
- SEDI: concentrations in river bottom sediment (Ci/kg$_{sediment}$ or kg$_{analyte}$/kg$_{sediment}$)
- SORP: concentrations in riparian zone soil (land surface) (Ci/kg$_{soil}$ or kg$_{analyte}$/kg$_{soil}$)
- SODR: concentrations in upland soil (land surface) with no irrigation (Ci/kg$_{soil}$ or kg$_{analyte}$/kg$_{soil}$)
- SOGW: concentrations in upland soil (land surface) with groundwater irrigation (Ci/kg$_{soil}$ or kg$_{analyte}$/kg$_{soil}$)
- SOSW: concentrations in upland soil (land surface) with surface water irrigation (Ci/kg$_{soil}$ or kg$_{analyte}$/kg$_{soil}$)
- AIRC: concentrations in air (Ci/m^3 or kg/m^3)
- AIRD: air deposition rates (Ci/m^2/yr or kg/m^2/yr)

An example of this keyword that uses concentrations in upland soil with surface water irrigation is the following:

```
MEDIA “SOSW”
```

13.3.2.7 REALIZATION Keyword for HIGHMEDIA

The REALIZAT keyword is used to define the realizations for which data will be processed and the summary technique to be used for a single case. The following is this keyword's syntax:

```
REALIZAT [MAXIMUM | MEAN | MEDIAN | SINGLE=N1]
```

The modifiers associated with the REALIZAT keyword are described in Table 13.3. Only one of the MAXIMUM, MEAN, MEDIAN or SINGLE modifiers are allowed during a run of the code.

Table 13.3 Modifiers Associated with the REALIZAT Keyword in HIGHMEDIA

Modifier	Description
MAXIMUM	This modifier indicates that the maximum over all realizations will be used in determining the highest impact at each location.
MEDIAN	This modifier indicates that the median over all realizations will be used in determining the highest impact at each location.

Modifier	Description
MEAN	This modifier indicates that the arithmetic mean over all realizations will be used in determining the highest result at each location.
SINGLE	This modifier indicates that data from a single realization will be used in determining the highest result at each location. The realization number to use is provided in the numerical value associated with the SINGLE modifier.

Example keywords where the user specifies using the single realization number 5, the arithmetic mean of all realizations, and the median of all realizations are the following:

```
REALIZAT SINGLE=5
REALIZAT MEAN
REALIZAT MEDIAN
```

13.3.2.8 TIMES Keyword for HIGHMEDIA

The TIMES keyword identifies the times (years) at which the calculations are to be performed for a single case. The following is this keyword's syntax:

```
TIMES [ALL | LIST [T1] {T2} … {Tn}]
```

All times in the ESD keyword file are used when the modifer ALL is present. A list of specific times can be defined by entering the modifier LIST and a set of numerical values identifying the years of interest. The specific years entered must be defined in the ESD keyword file. All times are included in the first example shown below. The second example defines a list of six years.

```
TIMES ALL
TIMES LIST 2020 2075 3014 3050 4000 12050
```

13.3.2.9 TITLE Keyword for HIGHMEDIA

The TITLE keyword is used to define a single-line problem title for a single case. The problem title will be written to output files. The program will error terminate if the title is not supplied. The following is this keyword's syntax:

```
TITLE [“quote”]
```

The title is entered in a quote string, which must be enclosed in double quotation marks. Titles up to 200 characters long are supported. The following example defines a title for a run of the code:

```
TITLE “Example title line for the INVSUM code.”
```

13.3.2.10 UNITS Keyword for HIGHMEDIA

The UNITS keyword is used to define the data units for the analyte to be processed. The following is this keyword's syntax:

```
   UNITS [OUTPUT=“quote 1”] [FACTOR=N1]
```

The media data in the ECDA files have a predefined set of units (see the MEDIA keyword in Section 13.3.2.6). This keyword allows the user to change the units upon output. The single quote string associated with the modifier OUTPUT contains the units label for the output data (limit of 10 characters). The numerical value associated with the modifier FACTOR is the multiplicative factor required to change from input units to output units. Use a value of 1 if the input units are to be preserved. Example keywords that use groundwater concentrations of Tc99 and convert the units from Ci/m^3 to pCi/L upon output are the following:

```
   ANALYTE “Tc99”
   MEDIA “GWAT”
   UNITS OUTPUT="pCi/L" FACTOR = 1.0E9
```

13.3.3 Concluding (END) Keyword for HIGHMEDIA

The END keyword signifies the end of all keyword data. All data in the keyword file after the END keyword will be ignored. The following is this keyword's syntax:

```
   END
```

14.0 HTWOS_TDP - Tank Inventory Data Processor

14.1 Overview

This processor is designed to analyze data file provided by the Hanford Tank Waste Operations Simulator (HTWOS) project describing the processing of waste in the 177 tanks at Hanford. The data is provided in a series of files, one file for each year from about 2001 through the end of processing (about 2028). The files give the total amount present at the end of each year, without decay correction. The differences between files for two successive years indicate changes occurring during that year.

This program reads the HTWOS data files, determines actions taken during each year, and generates disposal streams from processing of tank waste. Output from the program is a data file ready for importing into the inventory database. Because the HTWOS files do not include radioactive decay corrections, the HTWOS_TDP program incorporates decay calculations.

The waste disposal actions generated by the HTWOS_TDP program will include the disposal streams indicated in Table 14.1. The table also indicates the source of the data in the HTWOS files to be used to generate each waste stream.

Table 14.1 Waste Streams Generated from Processing of Tank Wastes

Waste Stream	Waste Type	Disposal Site Acronym
Pre-Treatment Streams PVP based on HLW-glass + ILAW-glass + Supplemental LAW Feed		
Cs-Resin	cmntH2	HTWOS-HLW-CSR
PVP – Offgas	gas	HTWOS-PVP-OFF
PVP – HEPA Filters	cmntH2	HTWOS-PVP-HEP
PJV – Offgas	gas	HTWOS-PJV-OFF
PJV - HEPA Filters	cmntH2	HTWOS-PJV-HEP
LERF - Solid Waste	cmntH2	HTWOS-LERF-SOL
LERF – Liquid	Liquid	HTWOS-LERF-LIQ
HLW Offgas	gas	HTWOS-HLW-OFF
Silver Mordenite Beds	cmntH2	HTWOS-HLW-IAG
HLW – HEPA Filters	cmntH2	HTWOS-HLW-HEP
HLW Carbon Absorption Beds	cmntH2	HTWOS-HLW-CSOL
HLW Failed/Spent Melters	glass	HTWOS-HLW-MLT
HLW glass product	Store	HTWOS-HLW-GLS
ILAW Carbon Absorption Beds	cmntH2	HTWOS-ILAW-CSOL
ILAW – Offgas	gas	HTWOS-ILAW-OFF
ILAW – HEPA Filters	cmntH2	HTWOS-ILAW-HEP
ILAW Failed/Spent Melters	glass	HTWOS-ILAW-MLT
ILAW glass product	glass	HTWOS-ILAW-GLS
LAW to ETF solid waste	cmntH2	HTWOS-LAW-SOL
LAW to ETF liquid waste	Liquid	HTWOS-LAW-LIQ
LAW – Offgas	gas	HTWOS-LAW-OFF
LAW – HEPA Filters	cmntH2	HTWOS-LAW-HEP
LAW supplement product	glass	LAW-supplement
CH-TRU from East Area Processing		
CH-TRU East ETF solid waste	cmntH2	HTWOS-CHE-SOL
CH-TRU East ETF liquid waste	Liquid	HTWOS-CHE-LIQ
CH-TRU East Offgas	gas	HTWOS-CHE-OFF

Waste Stream	Waste Type	Disposal Site Acronym
CH-TRU East HEPA Filters	cmntH2	HTWOS-CHE-HEP
CH-TRU East product	store	HTWOS-CHE
CH-TRU from West Area Processing		
CH-TRU West ETF solid waste	cmntH2	HTWOS-CHW-SOL
CH-TRU West ETF liquid waste	Liquid	HTWOS-CHW-LIQ
CH-TRU West Offgas	gas	HTWOS-CHW-OFF
CH-TRU West HEPA Filters	cmntH2	HTWOS-CHW-HEP
CH-TRU West product	store	HTWOS-CHW
RH-TRU Processing		
RH-TRU ETF solid waste	cmntH2	HTWOS-RH-SOL
RH-TRU ETF liquid waste	Liquid	HTWOS-RH-LIQ
RH-TRU Offgas	gas	HTWOS-RH-OFF
RH-TRU HEPA Filters	Gas	HTWOS-RH-HEP
RH-TRU product	Store	HTWOS-RH
Non-WTP-Low-Cs-Feed Processing		
Non-WTP ETF solid waste	cmntH2	HTWOS-NWTP-SOL
Non-WTP ETF liquid waste	Liquid	HTWOS-NWTP-LIQ
Non-WTP Offgas	gas	HTWOS-NWTP-OFF
Non-WTP HEPA Filters	cmntH2	HTWOS-NWTP-HEP
Non-WTP Product	Glass	NON-WTP
Tank Residue Material		
Tank residual waste after tank closure	res	241-“Tank ID”

The disposal stream amounts are estimated using fractions defined for each pollutant to each disposal type as appropriate for the pollutant. The fractions are provided in the HTWOS_TDP.INP control input data file (see Section 14.2.1).

The data in the HTWOS annual files gives the cumulative amount of each contaminant present at the end of the year. For each pair of years, the difference between annual values is the change in amounts, representing the amount of contaminant that goes from one source to one of the seven product streams (as described in the preceding paragraphs). When the tank amounts decrease, there should be a corresponding increase in one or more of the seven product streams.

14.1.1 Location in the Processing Sequence

The HTWOS_TDP code provides data files to be loaded into the inventory database. It must be run before the inventory code can be run.

14.1.2 How the Code Is Invoked

HTWOS_TDP runs in a DOS box under the Windows operating system. A run of HTWOS_TDP is initiated by entering the following command line:

```
HTWOS_TDP
```

The HTWOS_TDP utility code does not require any additional command line inputs. The assumption is used that all files used by the code will reside in the directory where the code was invoked.

14.2 File Definitions

The HTWOS_TDP code reads three or more input files and writes four output files.

14.2.1 Control Data File

Control information is read from a file that always has the name "HTWOS_TDP.INP". The structure of the file is described as follows and an example file is provided in Table 14.2:

Line 1: Number of radionuclides to extract from the HTWOS annual files
For each radionuclide, define a set of lines as follows:
Line n+1: Radionuclide name as listed in the HTWOS file and analyte name to be used for output
Line n+2: processing factors for pre-treatment of HLW + ILAW + LAW supp. The multiplicative factors are for the following waste stream splits:

- HLW Cs-Resins to solid waste
- PT Offgas - PVP
- PT HEPAs - PVP
- PT Offgas - PJV
- PT HEPAS - PJV
- PT LERF - Solid
- PT LERF - Liquid

Line n+3: processing factors for HLW glass. The multiplicative factors are for the following waste stream splits:

- HLW offgas to air
- HLW carbon bed to solid waste
- HLW Spent I/Ag mordenite beds
- HLW HEPAs to solid waste
- HLW failed melters to solid waste
- HLW to glass

Line n+4: processing factors for ILAW glass. The multiplicative factors are for the following waste stream splits:

- ILAW solid waste
- ILAW caustic scrubber offgass
- ILAW HEPAs to solid waste
- ILAW failed/spent melters
- ILAW Glass to solid waste

Line n+5: processing factors for LAW Supplemental. The multiplicative factors are for the following waste stream splits:

- WTP supplemental LAW ETF to solid waste
- WTP supplemental LAW ETF to liquid waste
- WTP supplemental LAW to air release
- WTP supplemental LAW HEPAS to solid
- WTP supplemental product

Line n+6: processing factors for CHTRU - contact handled TRU streams. The multiplicative factors are for the following waste stream splits:

- CHTRU processing to solid waste
- CHTRU processing to liquid waste
- CHTRU processing to air release
- CHTRU processing HEPA to solid
- CHTRU product

Line n+7: processing factors for RHTRU - remote handled TRU streams. The multiplicative factors are for the following waste stream splits:

- RHTRU processing to solid waste
- RHTRU processing to liquid waste
- RHTRU processing to air release
- RHTRU processing HEPA to solid
- RHTRU product

Line n+8: processing factors for LAW Non-WTP supplemental treatment streams. The multiplicative factors are for the following waste stream splits:

- NON-WTP supplemental LAW to solid waste
- NON-WTP supplemental LAW to liquid waste
- NON-WTP supplemental LAW to air release
- NON-WTP supplemental LAW HEPAS to solid
- NON-WTP product

Line N*8+2: Number of chemicals to extract from the HTWOS annual files
For each chemical, define a set of lines as follows:
Line m+1: Radionuclide name as listed in the HTWOS file and analyte name to be used for output
Line m+2: processing factors for pre-treatment of HLW + ILAW + LAW supp.
Line m+3: processing factors for HLW glass
Line m+4: processing factors for ILAW glass
Line m+5: processing factors for LAW Supplemental
Line m+6: processing factors for CHTRU - contact handled TRU streams
Line m+7: processing factors for RHTRU - remote handled TRU streams
Line m+8: processing factors for LAW Non-WTP supplemental treatment streams

Line N*8 + M*8 + 3: Start year of processing and end year of processing of tank waste

Line N*8 + M*8 + 4: Decay reference year for HTWOS data (based on beginning of year)

Next define volume factors for each waste stream as follows:
Line X+2: volume factors for pre-treatment of HLW + ILAW + LAW supp.
Line X+3: volume factors for HLW glass
Line X+4: volume factors for ILAW glass
Line X+5: volume factors for LAW Supplemental
Line X+6: volume factors for CHTRU - contact handled TRU streams
Line X+7: volume factors for RHTRU - remote handled TRU streams
Line X+8: volume factors for LAW Non-WTP supplemental treatment streams

Table 14.2 Example HTWOS_TDP.INP File for HTWOS_TDP

```
2,number of radionuclides,,,,,,
3H,H3,,,,,,
0,6.77794E-06,0,6.25167E-08,0.00E+00,0,0.45166047,PT,1
0.026744635,0,0,0,0,0,0,HLW,2
0,0.044880591,0,0,0,0,0,ILAW,3
0,0.476707463,0,0,0,0,0,LAW ,4
0,9.7E-11,0.00000161,0,0.99999839,0,0,CHTRU,5
0,9.7E-11,0.00000161,0,0.99999839,0,0,RHTRU,6
0,1,0,0,0,0,0,NON-WTP,7
14C,C14,,,,,,
6.6419E-09,7.89639E-06,0,5.55837E-09,0.00E+00,0.000636721,6.36784E-08,PT,1
0.003832443,0,0,0,0,0,0,HLW,2
0,0.397016737,0,0,0,0,0,ILAW,3
0.598446277,5.98506E-05,0,0,0,0,0,LAW ,4
0.0009999,0.0000001,0.00000978,0,0.99899022,0,0,CHTRU,5
0.0009999,0.0000001,0.00000978,0,0.99899022,0,0,RHTRU,6
0.9999,0.0001,0,0,0,0,0,NON-WTP,7
0,Number,of,chemicals,,,,
2001,2028,Start,and,end,years,,,
2000,Decay,referenced,date,for,HTWOS,input,files,
PT,2.59E-06,1000,3.96E-09,1000,1.39E-08,9.96E-06,2.89E-03,
HLW,1000,2.10E-06,5.35E-07,1.90E-07,1.35E-06,0.992,0,
ILAW,1.09E-07,1000,1.45E-07,2.6837E-07,1.17,0,0,0
WTP,2.03E-02,5.904,0,0,0.874,0,0,
CH-TRU,3.19E-03,0.927072,1000,0,1.076,0,0,
RH-TRU,3.19E-03,0.927072,1000,0,0.153,0,0,
NWTP,2.03E-02,5.904,0,0,0.522,,,
F,F,!,addition,flags,for,U,and,Pu
```

14.2.2 HTWOS Annual Data Files

The HTWOS annual data files contain information on seven product streams that form the basis for the final waste stream quantities. These HTWOS product streams are: CH-TRU-East, CH-TRU-West, RH-TRU, Non-WTP-Low-Cs-Feed, HLW-Glass, ILAW-glass, and Supplemental-LAW-Feed. There is a separate file for each year. The range of years is defined in the control data file (line 19 of the example file shown in Table 14.2). The name of the file for calendar year XXXX is “HTWOXXXX.CSV”. For example, the file for 2004 is named “HTWO2004.CSV”.

Each of the HTWOS annual files have an identical structure and use a comma-separated variables format. They each contain the following information:

Lines 1 – 3,	Title information
Line 4.	Blank
Line 5,	Super headings (radionuclides, chemicals, glass oxides)
Line 6.	Column headings (mostly string values)
Lines 7-9,	Data for the first tank for the current year

Lines 10-12, Data for the second tank for the current year
…
Lines 535 - 537, Data for the last tank for the current year
Lines 538 – 540, Data for CH-TRU West area sources
Lines 541 – 543, Data for CH-TRU East area sources
Lines 544 – 546, Data for RH-TRU sources
Lines 547 – 549, Data for Non-WTP Low Cs Feed
Lines 550 – 552, Total values over all tanks
Line 553, Data for capsules
Line 554, Data for HLW-glass
Line 555, Data for ILAW-glass
Line 556, Data for Suplemental LAW feed

Each of lines 7 through 556 has data for 46 radionuclides, 150 chemicals, followed by the volume (gallons), density (g/cc), and mass (kg) of the waste in the tank, capsules, or glass. The data for all but the last four lines have three entries per product stream. The first line gives the quantity in the “SOLID” phase, the next line gives the quantity in the “LIQUID” phase and the last line gives the “TOTAL” for the product stream. The HTWOS processor used only the “TOTAL” lines in the analysis.

14.2.3 Radionuclide Decay Chain Library

A third input file is a library of radionuclide decay chain data. This file always has the name “RMDLIB.DAT”. The format of this file is not described further because the user typically does not modify this file.

14.3 Output Files

The HTWOS_TDP code writes a log file and an error-message file for each run of the code. The log file is always named HTWOS_TDP.OUT and the error-message file is always named HTWOS_TDP.ERR. These files should be examined after every run to determine whether any unexpected data problems were encountered.

The primary output file is always named “HTWOS_TDP.CSV” and contains disposal data that is to be imported into the inventory database. This file uses the comma-separated variables format and contains decay corrected tank leak data. The values generated are the volume of each annual waste stream and the amount of each analyte in the waste stream. Volumes have units of m^3, amounts of radionuclides have units of Ci and amounts of chemicals have units of kg. The tank residual waste is defined to be the amount of material in the tanks as given in the last HTWOS file. Because the values for the seven product streams in the HTWOS files are cumulative values (increasing with time), the difference of the values for two successive years gives the amount processed during the year. The waste streams generated are assumed to be disposed during the year of processing.

The format of the primary output file is as follows:

- The first line is a comment line.
- The second line contains column titles as follows: Dataset , Waste Stream, Attribute, Year, Code, Value.
- The remaining lines contain disposal action amounts or volumes with entries corresponding to the titles listed above. The waste streams from processing the seven HTWOS product streams are given first, starting with the earliest years of processing and ending with the last year of processing. The data for tank residuals is given at the end of the output file, all with the designated last year of processing.

The first 15 lines from an example "HTWOS_TPD.CSV" file is provided in Table 14.3. A production data set yields an output file containing about 9150 lines of data.

Table 14.3 Excerpts from the HTWOS_TDP.CSV File written by HTWOS_TDP

```
! HTWOS processing data for tanks and HLW/ILAW after 2000
"Dataset  ","Waste Stream","Attribute","Year","Code","Value",
 HTWOS, HTWOS-CHE-SOL ,cmntH2 ,2005 , mean,Volume, 1.92100E+00
 HTWOS, HTWOS-CHE-SOL ,cmntH2 ,2005 , mean,  I129, 5.33589E-06
 HTWOS, HTWOS-CHE-SOL ,cmntH2 ,2005 , mean, Cs137, 1.62176E-08
 HTWOS, HTWOS-CHE-SOL ,cmntH2 ,2005 , mean,   C14, 2.34714E-07
 HTWOS, HTWOS-CHE-SOL ,cmntH2 ,2005 , mean, Eu152, 3.34710E-11
 HTWOS, HTWOS-CHE-SOL ,cmntH2 ,2005 , mean, Ra226, 4.60639E-18
 HTWOS, HTWOS-CHE-SOL ,cmntH2 ,2005 , mean, Pa231, 1.20850E-15
 HTWOS, HTWOS-CHE-SOL ,cmntH2 ,2005 , mean,  U233, 2.18547E-11
 HTWOS, HTWOS-CHE-SOL ,cmntH2 ,2005 , mean,  U234, 5.74160E-05
 HTWOS, HTWOS-CHE-SOL ,cmntH2 ,2005 , mean,  U235, 2.35160E-06
 HTWOS, HTWOS-CHE-SOL ,cmntH2 ,2005 , mean, Np237, 2.11883E-07
 HTWOS, HTWOS-CHE-SOL ,cmntH2 ,2005 , mean,  U238, 5.32207E-05
 HTWOS, HTWOS-CHE-SOL ,cmntH2 ,2005 , mean,  Se79, 6.89438E-12
 HTWOS, HTWOS-CHE-SOL ,cmntH2 ,2005 , mean,  Sr90, 2.07973E+00
 HTWOS, HTWOS-CHE-SOL ,cmntH2 ,2005 , mean,  Tc99, 1.05731E-02
```

The HTWOS_TDP code writes an additional data file named "STREAMS.OUT". This file should be ignored.

15.0 HTWOS_TLDP - Hanford Tank Leak Data Processing

15.1 Overview

The HTWOS_TLDP utility code extracts tank leak data from the Hanford Tank Waste Operations Simulator (HTWOS) data files and generates a reformatted data file that is ready for import into the inventory database.

15.1.1 Location in the Processing Sequence

The HTWOS_TLDP code provides data files to be loaded into the inventory database. It must be run before the inventory code can be run.

15.1.2 How the Code Is Invoked

HTWOS_TLDP runs in a DOS box under the Windows operating system. A run of HTWOS_TLDP is initiated by entering the following command line:

```
HTWOS_TLDP
```

The HTWOS_TLDP utility code does not require any additional command line inputs. The assumption is used that all files used by the code will reside in the directory where the code was invoked.

15.1.3 Processing Assumptions

The following assumptions are used in the data processing steps:

- Data values in the HTWOS file have units of Ci for radionuclides and Kg for chemicals.
- The total volume of liquid waste leaked is as provided in the HTWOS tank leak file for each tank.
- The leak occurs during the year specified in the HTWOS file.
- The activity value is the amount decayed to the end of the indicated year.

15.2 File Definitions

The HTWOS_TLDP code reads two input files and writes six output files. Only one of the output files (the one named “HTWOS_LEAK.CSV”) is used in further data processing.

15.2.1 Input Files

Two files are read by the Tank Leak Data Processor code: a control data file and an HTWOS tank leak data file. Both files are text files and the HTWOS tank leak file uses comma separated variable format.

The control data file defines the number and names of radionuclides to be included and the number and names of chemicals to be included. This file is always named “HTWOS_LIST.INP” and an example file is shown in Table 15.1. This file lists the analyte names in two forms: the form provided in the HTWOS tank leak data file (without hyphens) and the form to be written to the disposal action output file. Next the file provides similar information for chemicals to be included in the analysis. This is followed by the decay reference year. The last line read has two logical flags for addition of uranium isotopes and for addition of Pu240 to Pu239.

Table 15.1 Example File HTWOS_LIST.INP for HTWOS_TLDP

```
20                     ! number of radionuclides
3H       H3,  12.35    ! input name, output name, half life (years)
14C    C14    5730.
79Se   Se79,  0.0
90Sr   Sr90,  29.12
99Tc   Tc99,  0.0
129I   I129,  0.0
137Cs Cs137,  30.0
152Eu Eu152,  13.33
231Pa  Pa231  0.0
226Ra  Ra226  1600.
233U   U233,  0.0
234U   U234,  0.0
235U   U235,  0.0
236U   U236,  0.0
238U   U238,  0.0
237Np  Np237, 0.0
239Pu Pu239,  0.0
240Pu Pu240,  6537.
241Pu Pu241,  14.4
241Am Am241,  432.58
2                      ! number of chemicals
Cr(total    CrVI, 0.0  ! input name, output name, chemical halflife 0.0
U(total)       U, 0.0  ! uranium
2001.                  ! reference year for decay ("2000" = Jan. 01, 2001)
F F
```

The column headings in the HTWOS data file are compared to the names of analytes selected in the first file. Data for analytes matching the column headings are written to the output file. The leak amounts are treated as point values.

The second input file contains disposition data (the amounts of analytes leaked and the year leaked for each of the tanks). This file is always named "HTWOS_TLDP.CSV". The first line of the file gives the number of tanks described in the file. The second line gives column headings for each field on the following lines. The third line contains the tank ID and analyte ID's for all analytes in the data file. The remainder of the data file contains three lines of data per tank. Each of the remaining lines contains the following information.

- Tank ID: Short title for tank, to be used as waste stream ID (string, ten characters maximum)
- Waste form: "Solid", "liquid", or "total"
- For each analyte: amount disposed during the leak event
- Tank_Year: Year that each tank leaks (integer, four digits)

Only the total values are passed on to the inventory database. No example is provided for this file because formatting the production data set containing data for 148 tanks and 199 analytes for viewing in a text document was impractical.

15.2.2 Output Files

The HTWOS_TLDP code writes a log file and an error-message file for each run of the code. The log file is always named HTWOS_TLDP.OUT and the error-message file is always named HTWOS_TLDP.ERR. These files should be examined after every run to determine whether any unexpected data problems were encountered.

The primary output file is always named HTWOS_LEAK.CSV and contains disposal data that is to be imported into the inventory database. This file uses the comma-separated variables format and contains decay corrected tank leak data. The first line of the file contains the headings for each column of data. The remaining lines contain the data set name (HTWOS), the tank name, waste type "liquid", the year of disposal, volume or analyte name, and the amount (volume, activity (Ci), or mass (Kg) of each analyte). The first 15 lines of an HTWOS_LEAK.CSV file are shown in Table 15.2.

Table 15.2 Excerpts from the HTWOS_LEAK.CSV File

```
Source,SiteCode,Subsite,Year,Constituent_name,Constituent_value,Dataset
HTWOS, 241-A-101,tle,2015,Volume,3.03E+01,HTWOS_LEAK_7-04
HTWOS, 241-A-101,tle,2015,I129,2.00E-03,HTWOS_LEAK_7-04
HTWOS, 241-A-101,tle,2015,Cs137,1.68E+03,HTWOS_LEAK_7-04
HTWOS, 241-A-101,tle,2015,C14,9.78E-02,HTWOS_LEAK_7-04
HTWOS, 241-A-101,tle,2015,Eu152,4.58E-04,HTWOS_LEAK_7-04
HTWOS, 241-A-101,tle,2015,Ra226,1.92E-09,HTWOS_LEAK_7-04
HTWOS, 241-A-101,tle,2015,Pa231,1.56E-06,HTWOS_LEAK_7-04
HTWOS, 241-A-101,tle,2015,U233,4.65E-05,HTWOS_LEAK_7-04
HTWOS, 241-A-101,tle,2015,U234,2.59E-05,HTWOS_LEAK_7-04
HTWOS, 241-A-101,tle,2015,U235,1.10E-06,HTWOS_LEAK_7-04
HTWOS, 241-A-101,tle,2015,U236,6.17E-07,HTWOS_LEAK_7-04
HTWOS, 241-A-101,tle,2015,Np237,1.00E-03,HTWOS_LEAK_7-04
HTWOS, 241-A-101,tle,2015,U238,2.42E-05,HTWOS_LEAK_7-04
HTWOS, 241-A-101,tle ,2015,Pu239 ,1.60E-02,HTWOS_LEAK_7-04
```

The HTWOS_TLDP code writes two additional data files. One is always named "DISP_TLDP.OUT". This data file should be ignored. The other output file is always named "TANKLIST.OUT" and contains tank names and leak volumes (gal). The data in these two files are not used in further processing.

16.0 Imp_Med – Median Values for Impact Codes

16.1 Overview

A run of the ECEM, HUMAN, or TCERM codes with stochastic parameters set to the median value (50th percentile) of their range requires an input keyword file designed for a single realization. The input keyword files developed for these codes typically contain full stochastic distributions for the input variables. A sequence of computer codes is available that reads a full stochastic keyword file and writes the associated median-values keyword file. The same sequence of calculations applies to all three impacts codes.

16.1.1 Location in the Processing Sequence

Therefore, conversion of the stochastic keyword files into median-values keyword files must occur after all environmental release and transport codes have completed but before the impact code is executed.

16.1.2 How the Code Is Invoked

A sequence of three codes is used to convert the stochastic keyword file into a median-values keyword file. For purposes of discussion, let the stochastic keyword file be named “stochastic.key” and let the median-values keyword file be named “median.key”. The sequence of three codes is invoked as follows:

```
imp_step1.exe “stochastic.key”
stochastic.exe “Step1.Key”
imp_step2.exe “stochastic.key” “median.key”
```

The user does not need to prepare any input files or enter other commands to control this sequence of calculations.

16.1.3 Memory Requirements

Imp_step1 and Imp_step2 have minimal memory requirements.

16.2 Computational Sequence

A sequence of three codes is used to convert the stochastic keyword file into a median-values keyword file. Invocation of the three codes are described in the following sections.

16.2.1 Imp_Step1 Execution

The first step in the processing sequence is to run the IMP_STEP1 code. This code reads the stochastic keyword file and writes a separate keyword file for the STOCHASTIC code. The user does not need to modify the output keyword file.

Under the Windows operating system (Releases 98, NT, 2000, or XP), INV_STEP1 executes in a DOS box. A run of IMP_STEP1 is initiated by entering the following command line:

```
imp_step1 “stochastic.key”
```

Under the Linux operating system IMP_STEP1 is executed through any of the following Bourne Shell or C Shell commands:

```
imp_step1.exe “stochastic.key”
```

For these commands, IMP_STEP1 or imp_step1.exe is the name of the executable program and “stochastic.key” is the name of the stochastic keyword file.

16.2.2 Stochastic Execution

The second step in the processing sequence is to run the STOCHASTIC code. This code reads the file named “Step1.Key” written by the IMP_STEP1 code and writes a file named “Step1.Val” for use in the IMP_STEP2 code.

Under the Windows operating system (Releases 98, NT, 2000, or XP), STOCHASTIC executes in a DOS box. A run of STOCHASTIC is initiated by entering the following command line:

```
STOCHASTIC “Step1.Key”
```

Under the Linux operating system STOCHASTIC is executed through any of the following Bourne Shell or C Shell commands:

```
stochastic.exe “Step1.Key”
```

For these commands, STOCHASTIC or stochastic.exe is the name of the executable program, “Step1.Key” is the name of the input keyword file and “Step1.Val” is the name of the output file of median values.

16.2.3 Imp_Step2 Execution

The final step in the processing sequence is to run the IMP_STEP2 code. This code reads the stochastic keyword file, a file named “Step1.Val” written by the STOCHASTIC code, and writes the median-values keyword file.

Under the Windows operating system (Releases 98, NT, 2000, or XP), INV_STEP2 executes in a DOS box. A run of IMP_STEP2 is initiated by entering the following command line:

```
imp_step2 “stochastic.key” “median.key”
```

Under the Linux operating system IMP_STEP2 is executed through any of the following Bourne Shell or C Shell commands:

```
   imp_step2.exe “stochastic.key” “median.key”
```

For these commands, IMP_STEP2 or imp_step2.exe is the name of the executable program, “stochastic.key” is the name of the stochastic keyword file and “median.daf” is the name of the median-values keyword file.

17.0 INGRAB – Inventory Data Extraction

17.1 Overview

This program reads an inventory results (*.res) file generated by the SAC Rev. 1 inventory code and outputs files of inventory disposal actions, decay corrected accumulated inventory, and waste stream volumes by waste type for all waste sites for a multiple analytes.

17.1.1 Location in the Processing Sequence

INGRAB reads a file output by the inventory code. It can be run any time after the inventory code complete.

17.1.2 How the Code Is Invoked

INGRAB can run under either the Windows or the Linux operating system. Under the Windows operating system (Releases 98, NT, 2000, or XP), INGRAB executes in a DOS box. A run of INGRAB is initiated by entering the following command line:

```
   INGRAB “Keyfilename”
```

Under the Linux operating system, INGRAB is executed through any of the following Bourne Shell or C Shell commands:

```
   ingrab-1.exe “Keyfilename”
```

For these commands, INGRAB.EXE or ingrab-1.exe is the name of the executable program and “Keyfilename” is the name of a control keyword file. Both the name of the executable program and the keyword file may contain path information. If INGRAB is invoked without entering the name of the keyword file, then the code will prompt the user for the file name. The keyword file, which should be prepared using an editor that can handle ASCII files without leaving embedded control codes, contains text control information describing the run. If INGRAB cannot open the keyword file, then the code will terminate execution after writing an error message to the standard output device.

17.1.3 Memory Requirements

The INGRAB program can require a large amount of memory. Dynamic memory allocation is used and memory use depends on the number of realizations, number of waste sites and number of waste streams. If the memory requirements exceed approximately 1 gigabyte the data are subdivided and a number of intermediate scratch files are used. A run of the code using 1042 waste sites, 43 waste streams and 100 realizations required on the order of 1.2 Gb of memory and 35 Gb of scratch disk space.

17.2 File Definitions

The INGRAB program reads two input files and writes four output files. These files are described in the following sections.

17.2.1 Input Files

The INGRAB program reads a control keyword file and a results file written by the INVENTORY code that contains all disposal actions (nominally named “inventory.all”). The INGRAB keyword file contains control information. An example file is provided in Table 17.1. Detailed definitions of the keywords are provided in Section 17.3.

Table 17.1 Example Keyword File for INGRAB

```
REPORT "Ingrab_C14.rpt"
TITLE "Extract C14 for the CA data set on 14 Jun 2004"
USER "Paul W. Eslinger"
FILE ALL    "/home/ANALYSIS4/CA1_median/inv/inventory.all"
FILE AMOUNT "Ingrab_C14.csv"
FILE ACCUM  "IngrabA_C14.csv"
FILE VOLUME "IngrabV_C14.csv"
REALIZAT ALL
ANALYTE   "C14"
HALFLIFE   5.715000E+03
SITE "116-B-1" "116-B-2" PRINT
TYPE ALL PRINT
YEARS SUM 1944 2070
SHOW POSITIVE
END
```

17.2.2 Output Files

The INGRAB program writes three or four output files for every run of the code. The output files are the following:

- **Report File.** The report file contains information about the run progress and any error messages. It should be examined after every run of the code.
- **Inventory File**. The inventory file is a text file in comma separated format that contains the amount disposed for every disposal action (every waste type and every location). This file is organized in chronological order and also contains summary information for all sites and all waste types for each year.
- **Accumulated Inventory File**. The accumulated inventory file is a text file in comma separated format that contains the accumulated (and decay corrected) amounts for every disposal action at every site. This file is organized in chronological order and also contains summary information for all sites and all waste types for each year. This file is optional (see the FILE and YEARS keyword descriptions).

- **Volume File**. The volume file is a text file in comma separated format that contains the volume of waste disposed for every disposal action (every waste type and every location). This file is organized in chronological order and also contains summary information for all sites for each year.

17.3 Keyword Definitions INGRAB

In general, the keywords for the HIGHMEDIA code can be entered in any order. The following restrictions apply on keyword order:

- The REPORT keyword must be the first keyword in the file.
- The END keyword must be the last keyword in the file.

17.3.1 ANALYTE Keyword for INGRAB

The ANALYTE keyword is used to define the analyte for which data will be processed. The following is this keyword’s syntax:

```
ANALYTE [“quote 1”]
```

The single quote string associated must contain an analyte identification string (limit of six characters in length) from the set of the analytes in the inventory results file. An example of this keyword is the following:

```
ANALYTE “H3”
```

17.3.2 END Keyword for INGRAB

The END keyword signifies the end of all keyword data. All data in the keyword file after the END keyword will be ignored. The following is this keyword’s syntax:

```
END
```

17.3.3 EXECUTE Keyword for INGRAB

The optional EXECUTE keyword indicates that the problem defined in the control keyword file should actually be executed. The following is this keyword record’s syntax:

```
EXECUTE
```

If the EXECUTE keyword is not entered the inputs will be checked for errors but no extractions will be performed.

17.3.4 FILE Keyword for INGRAB

The FILE keyword is used to enter the names of all input and output files except for the report file. The following is this keyword's syntax:

```
FILE [ACCUM="quote1"] {ALL="quote2"} {AMOUNT="quote3"} {VOLUME="quote4"}
```

The file names are entered in quote strings, which must be enclosed in double quotes. Path names up to 200 characters long are supported. The file name associated with a modifier must be entered before the next modifier is entered. At least one FILE keyword is required for every run of the code.

The name of the summary inventory file written by the inventory code is associated with the modifier ALL. This file is an input file to INGRAB. The name of the accumulated (and decay corrected) disposal amounts file written by INGRES is associated with the modifier ACCUM. The name of the disposal amounts file written by INGRES is associated with the modifier AMOUNT. The name of the disposal volumes file written by INGRES is associated with the modifier VOLUME. Example file keywords that define these four files are the following:

```
FILE ALL    "/home/ANALYSIS4/CA1_median/inventory/inventory.all"
FILE AMOUNT "Ingrab_C14.csv"
FILE ACCUM  "IngrabA_C14.csv"
FILE VOLUME "IngrabV_C14.csv"
```

17.3.5 HALFLIFE Keyword for INGRAB

The HALFLIFE keyword is used to define the halflife of the analyte for which data will be processed. The following is this keyword's syntax:

```
HALFLIFE [N1]
```

The single numerical value must contain the halflife (units are years) of the analyte being processed. The value of 0 should be used if the analyte is not radioactive. An example of this keyword is the following:

```
HALFLIFE 5.6262E-02
```

17.3.6 REALIZAT Keyword for INGRAB

The REALIZAT keyword defines the realizations to be used in the inventory data extractions. The following is this keyword's syntax:

```
REALIZAT [ALL | N1]
```

If the modifier ALL is present all realizations of data will be processed. If the modifier ALL is not present a single realization number must be entered and the data extraction will occur for only that realization. Two example keywords are the following:

```
REALIZAT ALL
REALIZAT 4
```

The first example uses all realizations in the data set. The second example uses data from only realization number 4.

17.3.7 REPORT Keyword for INGRAB

The REPORT keyword is used to define the name of the output report (log) file. It must be the first keyword entered in the keyword file. The following is this keyword's syntax:

```
REPORT [“quote”]
```

The name of the report file is entered in a quote string. File names up to 200 characters long are supported, and path names can be included. The following is an example REPORT keyword record:

```
REPORT “/SAC/C/Test.rpt”
```

17.3.8 SHOW Keyword for INGRAB

The optional SHOW keyword is used to limit data output to values greater than zero. The following is this keyword's syntax:

```
SHOW POSITIVE
```

The single modifer POSITIVE must be used if the output is to be limited to data values greater than zero. An example of this keyword is the following:

```
SHOW POSITIVE
```

It is recommended that this keyword be used with large data sets. If not, the possibility exists that the output file sizes can exceed the maximum size allowed by the compiler and the run will error terminate.

17.3.9 SITE Keyword for INGRAB

The SITE keyword is used to define the sites included in the data extractions. The following is this keyword's syntax:

```
SITE {PRINT} [ALL | “quote1” … “quoteN”]
```

If the optional PRINT modifier is present then details of all actions will be printed as well as summary disposal actions. Only summary disposal actions will be printed when PRINT is not present. The choice of which sites to include has two options. The first option is to enter the single modifier ALL, thereby invoking outputs for every waste site with disposal data. The other option is to enter a list of site ID's.

The first example keyword given below uses all sites and will output detailed information for every site. The second example uses only three sites and also outputs detailed information for just these three sites.

```
SITE ALL PRINT
SITE “116-B-1” “116-B-2” “116-B-3” “116-B-5” PRINT
```

17.3.10 TITLE Keyword for INGRAB

The TITLE keyword is used to define a single-line problem title. The problem title will be written to output files. The program will error terminate if the title is not supplied. The following is this keyword's syntax:

```
   TITLE [“quote”]
```

The title is entered in a quote string, which must be enclosed in double quotation marks. Titles up to 200 characters long are supported. The following example defines a title for a run of the code:

```
   TITLE “Example title line for the INVSUM code.”
```

17.3.11 TYPE Keyword for INGRAB

The TYPE keyword is used to define the waste types included in the data extractions. The following is this keyword's syntax:

```
   TYPE {PRINT} [ALL | “quote1” … “quoteN”]
```

If the optional PRINT modifier is present then details of all actions will be printed as well as summary disposal actions. Only summary disposal actions will be printed when PRINT is not present. The choice of which waste types to include has two options. The first option is to enter the single modifier ALL, thereby invoking outputs for every waste type with disposal data. The other option is to enter a list of waste ID's.

The first example keyword given below uses all waste types and will output detailed information for every waste type. The second example uses only six specific waste types and also outputs detailed information for these six waste types.

```
TYPE ALL PRINT
TYPE "liquid" "glass" "soil" "cake" "gases" "cement" PRINT
```

17.3.12 USER Keyword for INGRAB

The USER keyword is used to identify the user of the program. The user name will be written to output files. The program will error terminate if the user name is not supplied. The following is this keyword's syntax:

```
   USER [“quote”]
```

The user name is entered in a quote string, which must be enclosed in double quotation marks. User names up to 16 characters long are supported. The following example defines John Q. Public as the user running the code:

```
   USER “John Q. Public”
```

17.3.13 YEARS Keyword for INGRAB

The YEARS keyword supplies data controlling the time period for the data extraction and whether the accumulated inventory file will be written. The following is this keyword's syntax:

```
YEARS {SUM} [N1 N2]
```

Presence of the optional modifier SUM initiates accumulation calculations that will be output in the accumulation inventory file. The two numerical values N1 and N2 give the year range (inclusive of the end years) for all of the extraction calculations. The second year cannot be smaller than the first year. The first year must be 1944 or larger. An example keyword entry is the following:

```
YEARS SUM 1944 2070
```

18.0 INGRES – Inventory Data Extraction (RES Files)

18.1 Overview

This program reads an inventory results (*.res) file generated by the SAC Rev. 1 inventory code and outputs a file of decay corrected accumulated inventory and waste stream volumes by waste type for all waste sites for a single analyte. All functions performed by this program can be performed by the INGRAB code. However, INGRAB reads the file “inventory.all” rather than a specific results file and can extract data for more than one realization at a time.

18.1.1 Location in the Processing Sequence

INGRES reads a file output by the inventory code. It can be run any time after the inventory code complete.

18.1.2 How the Code Is Invoked

INGRES can run under either the Windows or the Linux operating system. Under the Windows operating system (Releases 98, NT, 2000, or XP), INGRES executes in a DOS box. A run of INGRES is initiated by entering the following command line:

```
INGRES “Keyfilename”
```

Under the Linux operating system INGRES is executed through any of the following Bourne Shell or C Shell commands:

```
ingres-1.exe “Keyfilename”
```

For these commands, INGRES.EXE or ingres-1.exe is the name of the executable program and “Keyfilename” is the name of a control keyword file. Both the name of the executable program and the keyword file may contain path information. If INGRES is invoked without entering the name of the keyword file, then the code will prompt the user for the file name. The keyword file, which should be prepared using an editor that can handle ASCII files without leaving embedded control codes, contains text control information describing the run. If INGRES cannot open the keyword file, then the code will terminate execution after writing an error message to the standard output device.

18.1.3 Memory Requirements

The memory used by INGRES program depends on the number of sites and waste streams. A run of the code to using 1000 waste sites and 40 waste streams required on the order of 126 MB of memory.

18.2 File Definitions

The INGRES program reads two input files and writes two output files. These files are described in the following sections.

18.2.1 Input Files

The INGRES program reads a control keyword file and a results file written by the INVENTORY code. The INGRES keyword file contains control information. An example file is provided in Table 18.1. Detailed definitions of the keywords are provided in Section 18.3.

Table 18.1 Example Keyword File for INGRES

```
! Keyword file for INGRES
REPORT "Test.Rpt"
TITLE "Inventory Sums by Site for Testing IngRes"
USER "Paul W. Eslinger"
! The *.res file produced by the inventory code
FILE RES "inv1.res"
! The decay corrected accumulated inventory and volume data
FILE SUM "Test_sum.csv"
! Analyte specific data
ANALYTE ID="H3"  LAMBDA = 5.6262E-02
! Year for output data
YEAR DECAY=2060 START=1944 STOP=2004
! Waste streams to use in the analysis
WASTE ALL
END
```

18.2.2 Output Files

The INGRES program writes two output files. One file is a report file that contains text information about the run of the code. It is not described further here. The other output file is the results file containing the information about the inventory assigned to a set of locations decayed to a common year. The first 24 records from an INGRAB output file are provided in Table 18.2. The last block of records in this file (not shown in the table) is the volume of the inventory by waste stream at every waste site.

Table 18.2 Excerpts from an INGRES Results File

```
"Release by site"
"218-E-ILAW", 0.00000E+00
"218-E-Melter", 0.00000E+00
"atmosphere", 0.00000E+00
"store", 0.00000E+00
"216-U-1%2-Fast", 0.00000E+00
"100-B-15", 0.00000E+00
"100-B-3", 0.00000E+00
"100-B-5", 3.66629E-06
"100-B-8", 0.00000E+00
"100-C-3", 0.00000E+00
"100-C-6", 0.00000E+00
"100-D-23", 0.00000E+00
"100-D-24", 0.00000E+00
"100-D-29", 0.00000E+00
"100-D-3", 4.34891E-03
"100-D-32", 4.28570E-03
```

```
"100-D-40", 4.34891E-03
"100-D-42", 0.00000E+00
"100-D-43", 0.00000E+00
"100-D-45", 0.00000E+00
"100-D-47", 4.34891E-03
"100-D-49", 0.00000E+00
"100-D-53", 0.00000E+00
```

18.3 Keyword Definitions INGRES

In general, the keywords for the INGRES code can be entered in any order. The following restrictions apply on keyword order:

- The REPORT keyword must be the first keyword in the file.
- The END keyword must be the last keyword in the file.

18.3.1 ANALYTE Keyword for INGRES

The ANALYTE keyword is used to define the analyte for which data will be processed. The following is this keyword's syntax:

```
ANALYTE [ID=“quote 1”] [LAMBDA=N1]
```

The single quote string associated with the modifier ID must an analyte identification string (limit of six characters in length) from the set of the analytes in the inventory results file. The numerical value associated with the modifier LAMBDA is the decay constant (units of year^{-1}) for the analyte. Use a value of 0 if the analyte is not radioactive. An example of this keyword is the following:

```
ANALYTE ID="H3"  LAMBDA = 5.6262E-02
```

18.3.2 END Keyword for INGRES

The END keyword signifies the end of all keyword data. All data in the keyword file after the END keyword will be ignored. The following is this keyword's syntax:

```
END
```

18.3.3 FILE Keyword for INGRES

The FILE keyword is used to enter the names of all input and output files except for the report file. The following is this keyword's syntax:

```
FILE [modifier1="quote1"] {modifier2="quote2"}
```

The file names are entered in quote strings, which must be enclosed in double quotes. Path names up to 200 characters long are supported. The file name associated with a modifier must be entered before the next modifier is entered. At least one FILE keyword is required for every run of the code.

The name of the results (*.res) file written by the inventory code is associated with the modifier RES. The name of the output file from INGRES is associated with the modifier SUM. Example file keywords that define these two files are the following:

```
FILE RES "/home/ANALYSIS4/CA1_median/inventory/inv01.res"
FILE SUM "Decayed_inventory.csv"
```

18.3.4 REPORT Keyword for INGRES

The REPORT keyword is used to define the name of the output report (log) file. It must be the first keyword entered in the keyword file. The following is this keyword's syntax:

```
REPORT [“quote”]
```

The name of the report file is entered in a quote string. File names up to 200 characters long are supported, and path names can be included. The following is an example REPORT keyword record:

```
REPORT “/SAC/C/Test.rpt”
```

18.3.5 TITLE Keyword for INGRES

The TITLE keyword is used to define a single-line problem title. The problem title will be written to output files. The program will error terminate if the title is not supplied. The following is this keyword's syntax:

```
TITLE [“quote”]
```

The title is entered in a quote string, which must be enclosed in double quotation marks. Titles up to 200 characters long are supported. The following example defines a title for a run of the code:

```
TITLE “Example title line for the INVSUM code.”
```

18.3.6 USER Keyword for INGRES

The USER keyword is used to identify the user of the program. The user name will be written to output files. The program will error terminate if the user name is not supplied. The following is this keyword's syntax:

```
USER [“quote”]
```

The user name is entered in a quote string, which must be enclosed in double quotation marks. User names up to 16 characters long are supported. The following example defines John Q. Public as the user running the code:

```
USER “John Q. Public”
```

18.3.7 VERBOSE Keyword for INGRES

The presence of the optional VERBOSE keyword initiates additional output to the report file. The following is this keyword’s syntax:

```
VERBOSE
```

18.3.8 WASTE Keyword for INGRES

The WASTE keyword is used to define the waste streams for which data summations will be performed. The following is this keyword’s syntax:

```
WASTE [ ALL | “quote 1” … “quote N” ]
```

The single modifier ALL should be entered only if all waste streams are desired. A specific set of waste streams can be identified by entering their identifiers in quote strings. Two examples of this keyword are provided here. The first example performs data summations for all waste streams. The second example performs data summations for 21 specific waste streams.

```
WASTE ALL
WASTE "soil" "glass" "SF" "soil1f" "soil1n" "soil3f" "soil3n"
   "cmntcs", "cmntg3", "rxcomp", "soilsf" "soilsn" "TRASH" "cmntct"
  "TRUf" "TRUn" "cmntcu" "soiltu" "cmntHT" "cmntH2" "liquid"
```

18.3.9 YEAR Keyword for INGRES

The YEAR keyword is used to define the years for specific data actions. The following is this keyword’s syntax:

```
YEAR [START=“N1”] [STOP=N2] [DECAY=N3]
```

The numerical value associated with the modifier START identifies the (integer) calendar year for the start of the data sums. The numerical value associated with the modifier STOP identifies the (integer) calendar year for the end of the data sums. The numerical value associated with the modifier DECAY identifies the (integer) calendar year to which all radioactive elements will be decay corrected. An example of this keyword is the following:

```
YEAR DECAY=2060 START=1944 STOP=2004
```

19.0 INPROC – Inventory Preprocessing

19.1 Overview

The INPROC code reads a file of disposal action data produced by the SAC inventory database. The information in this file is converted into disposal action keyword for use by the SAC Inventory module. The disposal action keywords are defined as statistical distributions. The INPROC code applies statistical distribution rules, waste type mapping rules, and concentration data fill-in rules in the process of developing the disposal action keywords.

19.1.1 Location in the Processing Sequence

The INPROC code reads data files written by the SAC inventory database. INPROC must be executed before the INVENTORY code can be executed.

The INPROC program interacts with the inventory module of the SAC computer code through data files. In general, the inventory model obtains input data from the following sources:

- Control information from the Keyword Control file and the Environmental Settings file
- Waste stream aggregation names from the Environmental Settings File
- Waste stream aggregation map from the Inventory keyword control file
- Disposal actions from the Master Waste Stream Disposal Action file
- Waste stream selection from the Waste Stream Selection file

The INPROC program generates the master waste stream disposal action file and aggregation keyword records that are placed in the INVENTORY keyword control file. The INPROC program obtains information from the SAC inventory database provided in a file prepared specifically for INPROC using database queries.

19.1.2 How the Code Is Invoked

INPROC can run under either the Windows or the Linux operating system. Under the Windows operating system (Releases 98, NT, 2000, or XP), INPROC executes in a DOS box. A run of INPROC is initiated by entering the following command line:

```
INPROC
```

Under the Linux operating system INPROC is executed through any of the following Bourne Shell or C Shell commands:

```
inproc-1.exe
```

For these commands, INPROC.EXE or inproc-1.exe is the name of the executable program. The name of the executable program may contain path information. INPROC reads an file named inproc.key from the local where the code is invoked. The keyword file, which should be prepared using an editor that can handle ASCII files without leaving embedded control codes, contains text control information describing the run. The code will terminate execution after writing an error message to the standard output device if INPROC cannot open the keyword file.

19.1.3 Memory Requirements

The INPROC code uses dynamic memory allocation, so the memory requirements depend on the problem being analyzed. It is expected that most, if not all, of the runs of the INPROC code will require less than 128 MB of memory.

19.1.4 Inventory Estimation Rules

Methods to estimate analyte mass or activities are provided when data are missing. Several methods implemented in INPROC are summarized here:

- **Isotopic Ratio Method**. Isotopic ratio information can be used to estimate quantities for missing radionuclides, such as isotopes of uranium and plutonium. If one isotope is known, then the other isotopes can be estimated.
- **Fuel Ratio Method**. When values for fission products are unknown, they may be estimated from the values of known fission products. In the initial assessment, four fission products were considered for this method Cs-137, Sr-90, Tc-99, and I-129. The factors used to estimate activities were defined as a function of year of definition.
- **Uranium Mass**. Conversion of uranium quantities from mass to activity can be made using specific activity factors and assumed isotopic ratios (as defined above for the isotopic ratio method). Similar conversions can be made for uranium mass if the activity of an isotope is known.

19.1.5 Statistical Rules

The output parameters from the INPROC program must be expressed as statistical distributions. Because many of the data in the SAC inventory database are stored in the form of point values, rules must be defined to translate the point values into statistical distributions. The rules are supplied to the INPROC program through keywords (see the STATRULE keyword in Section 19.3.8).

The definition of statistical distribution types is limited by the information available on a given parameter. If only one value is given, then assumptions must be made to define a distribution. For example, the single value could be assumed to represent the mean of a normal distribution for which the standard deviation is defined as some factor times the mean. If a minimum and maximum value are known, the distribution could be assumed to be uniform over the interval between the two values. These assumptions are supplied using the STATRULE keyword.

19.2 File Definitions

The INPROC code reads three input files and writes four output files. These files are described in the following sections.

19.2.1 Input Files

The INPROC code reads a control keyword file that is always named “inproc.key”. In addition it reads a file containing aggregate site mappings and another file containing disposal action data. Excerpts from the control keyword file for INPROC are provided in Table 19.1. Example keyword data are provided for each keyword, however the data provided here do not define a complete or internally consistent keyword file.

Table 19.1 Excerpts from a Keyword Control File for INPROC

```
!   INPROC Keyword File
TITLE "2004 Composite Analysis Baseline Inventory"
USER "Paul W. Eslinger"
!
! File definitions
FILE DATA     "INPROC_Input_Terri_rev.csv" ! Input file
FILE SITE     "Aggr_site.csv"              ! Input file
FILE SUMMARY  "CA1-stoch.out"              ! Output file
FILE AGGREGAT "CA1-aggregat.key"           ! Output file
FILE WASTE    "CA1-stoch.daf"              ! Output file
VERBOSE
!
MATCH YEARS
PERIOD START=1944 STOP=12050 CLOSURE=2070
!
! Analyte definitions
ANALYTE ID "U233"  NAME "Uranium-233"      TYPE "NR" COMPUTE OUTPUT
ANALYTE ID "U234"  NAME "Uranium-234"      TYPE "NR" COMPUTE OUTPUT
!
! Subset of the burial ground subsites
ATTRIBUTE "S3N"    TYPE "soil3n" "Cat 3 from onsite"
ATTRIBUTE "UCN"    TYPE "cmntcu" "unsegregated caisson onsite"
ATTRIBUTE "LCN"    TYPE "cement" "Lithium Targets in Caison"
ATTRIBUTE "RTG"    TYPE "store"  "A sealed source; not disposed"
ATTRIBUTE "S"      TYPE "Core"   "Reactor cores"
ATTRIBUTE "NR"     TYPE "River"  "Outfall releases to the river"
ATTRIBUTE "cement" TYPE "cement" "cement"
ATTRIBUTE "upr"    TYPE "liquid" "unplanned release"
!
! Subset of the waste types
WASTYPE NAME "cake"
WASTYPE NAME "capsul"
WASTYPE NAME "cement"
WASTYPE NAME "cmntcs"
WASTYPE NAME "cmntct"
WASTYPE NAME "cmntcu"
WASTYPE NAME "WIPP"
!
! Statistical rules
STATRULE DATASET "SIM_7_Jun_04" INPUT  "VOLUME" PASSTHRU
  APPLY "USER" 0.80 1.2 YEARS ALLTIMES
STATRULE DATASET "SIM_7_Jun_04" INPUT "ANALYTE" PASSTHRU
  APPLY "USER" 0. 0. YEARS ALLTIMES
STATRULE DATASET "ALL" MEAN   "ANALYTE" APPLY "LOGE" 2.00
  YEARS BEFORE 1970 TRUNCATE .01 .99
STATRULE DATASET "ALL" MEAN   "ANALYTE" APPLY "LOGE" 0.25
```

```
  YEARS AFTER 1969  TRUNCATE .01 .99
STATRULE DATASET "ALL" MEAN   "VOLUME"  APPLY "TRIANGLE" 0.8 1.2
  YEARS ALLTIMES
STATRULE DATASET "ALL" POINT  "ANALYTE" APPLY "LOGE" 2.00
  YEARS BEFORE 1970 TRUNCATE .01 .99
STATRULE DATASET "ALL" POINT  "ANALYTE" APPLY "LOGE" 0.25
  YEARS AFTER 1969  TRUNCATE .01 .99
STATRULE DATASET "ALL" POINT  "VOLUME"  APPLY "TRIANGLE" 0.8 1.2
  YEARS ALLTIMES
STATRULE DATASET "ALL" MODE   "ANALYTE" APPLY "LOGE" 2.00
  YEARS BEFORE 1970 TRUNCATE .01 .99
STATRULE DATASET "ALL" MODE   "ANALYTE" APPLY "LOGE" 0.25
  YEARS AFTER 1969  TRUNCATE .01 .99
STATRULE DATASET "ALL" MODE   "VOLUME"  APPLY "TRIANGLE" 0.8 1.2
  YEARS ALLTIMES
!
! Summation rules
SUM TO "U238" FROM 2  "U234" "U238"
!
! Surrogate site rules
SURROGATE UNKSITE="100-D-3" 434.782 1958 SURSITE "118-D-2" 1956 "UTN"
SURROGATE UNKSITE="100-D-3" 434.782 1959 SURSITE "118-D-2" 1956 "UTN"
SURROGATE UNKSITE="100-H-8" 25 1962 SURSITE "116-B-4" 1960 "liquid"
SURROGATE UNKSITE="100-H-8" 25 1963 SURSITE "116-B-4" 1960 "liquid"
SURROGATE UNKSITE="200-W-69" 535 2000 SURSITE "218-W-3A" 1991 "STN"
SURROGATE UNKSITE="216-SX-2" 5882.3 1952 SURSITE "216-A-11" 1964 "liquid"
SURROGATE UNKSITE="241-A-431" 800 1969 SURSITE "276-U" 1959 "cement"
!
! Estimation factors using U238 as the known analyte
ESTIMATE ANALYTE "U234" FROM "U238" FACTOR 0.97  YEARS
  ALLYEARS SITES ALLSITES
ESTIMATE ANALYTE "U235" FROM "U238" FACTOR 0.042 YEARS
  ALLYEARS SITES ALLSITES
!
END
```

A shortened example of an aggregate site mapping file for INPROC is provided in Table 19.2. The first line contains the number of sites and labels for the three columns. INPROC allows aggregation of waste forms (third entry on the data line) from several sites (first entry on the data line) to one output site (second entry on the data line). This option has never been exercised because advances in the VADER code made this aggregation unnecessary. The operational mode is to map each site to itself for a typical waste stream. The site list must be complete with respect to the sites defined in the inventory data file.

Table 19.2 Aggregate Site Mapping File for INPROC

```
8,SiteCode,SiteCode,WasteType
100-B-3,100-B-3,liquid
100-B-5,100-B-5,liquid
100-B-8,100-B-8,liquid
100-B-15,100-B-15,liquid
100-C-3,100-C-3,liquid
100-C-6,100-C-6,liquid
100-D-3,100-D-3,soiltu
100-D-23,100-D-23,soiltu
```

A shortened version of the inventory disposal data is provided in Table 19.3. This example file contains data for three sites (100-H-5, 100-N-66, and 200-E-100) for a few years. The “derived-records” data sets will have statistical rules assigned while the “SIM_7_Jun_04” data set has a user defined distribution that will be passed through without modification. The production data set for the 2004 Composite Analysis contains over 1.5 million records.

Table 19.3 Excerpted Records from an INPROC Inventory Action File

```
"Derived-Records","100-H-5","soil",1953,"Mean","C14",2.08,"dpa4[0]",1865,"none",
"Derived-Records","100-H-5","soil",1953,"Mean","Cs137",2.13,"dpa4[0]",1865,"none",
"Derived-Records","100-H-5","soil",1953,"Mean","Eu152",73.91,"dpa4[0]",1865,"none",
"Derived-Records","100-H-5","soil",1953,"Mean","H3",0.86,"dpa4[0]",1865,"none",
"Derived-Records","100-H-5","soil",1953,"Mean","Sr90",0.32,"dpa4[0]",1865,"none",
"Derived-Records","100-H-5","soil",1953,"Mean","U",1.77857e-04,"dpa4[0]",1865,"none",
"Derived-Records","100-H-5","soil",1953,"Mean","U234",2.38e-07,"dpa4[0]",1865,"none",
"Derived-Records","100-H-5","soil",1953,"Mean","U235",2.49e-09,"dpa4[0]",1865,"none",
"Derived-Records","100-H-5","soil",1953,"Mean","U236",9.78214e-10,"dpa4[0]",1865,"none",
"Derived-Records","100-H-5","soil",1953,"Mean","U238",5.94042e-08,"dpa4[0]",1865,"none",
"Derived-Records","100-H-5","soil",1953,"Mean","Volume",7345.00,"dpa4[0]",1865,"none",
"Derived-Records","100-N-66","S",1998,"Point","Am241",0.30,"dpaA/2003",3941,"none",
"Derived-Records","100-N-66","S",1998,"Point","C14",9550.00,"dpaA/2003",3941,"none",
"Derived-Records","100-N-66","S",1998,"Point","Cl36",75.00,"dpaA/2003",3941,"none",
"Derived-Records","100-N-66","S",1998,"Point","Cs137",40.00,"dpaA/2003",3941,"none",
"Derived-Records","100-N-66","S",1998,"Point","Eu152",58.00,"dpaA/2003",3941,"none",
"Derived-Records","100-N-66","S",1998,"Point","H3",64000.00,"dpaA/2003",3941,"none",
"Derived-Records","100-N-66","S",1998,"Point","Sr90",14.30,"dpaA/2003",3941,"none",
"Derived-Records","100-N-66","S",1998,"Point","Tc99",0.03,"dpaA/2003",3941,"none",
"Derived-records","100-N-66","S",1998,"Point","Volume",500.00,"dpaA/2003",3941,"none",
"SIM_7_Jun_04","200-E-100","liquid",1945,"V0200","U238",3.5874e-10,"UNA200AREA",4360,"None",
"SIM_7_Jun_04","200-E-100","liquid",1945,"V0050","U238",1.8203e-10,"UNA200AREA",4360,"None",
"SIM_7_Jun_04","200-E-100","liquid",1945,"V0450","U238",5.3728e-10,"UNA200AREA",4360,"None",
"SIM_7_Jun_04","200-E-100","liquid",1945,"V0950","U238",1.0543e-09,"UNA200AREA",4360,"None",
"SIM_7_Jun_04","200-E-100","liquid",1945,"V0250","U238",4.0079e-10,"UNA200AREA",4360,"None",
"SIM_7_Jun_04","200-E-100","liquid",1945,"V0100","U238",2.5472e-10,"UNA200AREA",4360,"None",
"SIM_7_Jun_04","200-E-100","liquid",1945,"V0750","U238",7.7905e-10,"UNA200AREA",4360,"None",
"SIM_7_Jun_04","200-E-100","liquid",1945,"V0150","U238",3.1116e-10,"UNA200AREA",4360,"None",
"SIM_7_Jun_04","200-E-100","liquid",1945,"Mean","U238",5.9278e-10,"UNA200AREA",4360,"None",
"SIM_7_Jun_04","200-E-100","liquid",1945,"V0900","U238",9.6264e-10,"UNA200AREA",4360,"None",
"SIM_7_Jun_04","200-E-100","liquid",1945,"V0800","U238",8.3154e-10,"UNA200AREA",4360,"None",
"SIM_7_Jun_04","200-E-100","liquid",1945,"V0995","U238",1.2114e-09,"UNA200AREA",4360,"None",
"SIM_7_Jun_04","200-E-100","liquid",1945,"V0700","U238",7.3086e-10,"UNA200AREA",4360,"None",
"SIM_7_Jun_04","200-E-100","liquid",1945,"V0650","U238",6.874e-10,"UNA200AREA",4360,"None",
"SIM_7_Jun_04","200-E-100","liquid",1945,"V0600","U238",6.4724e-10,"UNA200AREA",4360,"None",
"SIM_7_Jun_04","200-E-100","liquid",1945,"V0550","U238",6.0841e-10,"UNA200AREA",4360,"None",
"SIM_7_Jun_04","200-E-100","liquid",1945,"V0500","U238",5.7285e-10,"UNA200AREA",4360,"None",
"SIM_7_Jun_04","200-E-100","liquid",1945,"V0850","U238",8.9016e-10,"UNA200AREA",4360,"None",
"SIM_7_Jun_04","200-E-100","liquid",1945,"V0400","volume",4.14,"UNA200AREA",4360,"None",
"SIM_7_Jun_04","200-E-100","liquid",1945,"V0200","volume",3.57,"UNA200AREA",4360,"None",
"SIM_7_Jun_04","200-E-100","liquid",1945,"V0350","volume",4.02,"UNA200AREA",4360,"None",
"SIM_7_Jun_04","200-E-100","liquid",1945,"V0300","volume",3.88,"UNA200AREA",4360,"None",
"SIM_7_Jun_04","200-E-100","liquid",1945,"V0250","volume",3.73,"UNA200AREA",4360,"None",
"SIM_7_Jun_04","200-E-100","liquid",1945,"V0150","volume",3.38,"UNA200AREA",4360,"None",
"SIM_7_Jun_04","200-E-100","liquid",1945,"V0100","volume",3.16,"UNA200AREA",4360,"None",
"SIM_7_Jun_04","200-E-100","liquid",1945,"V0050","volume",2.88,"UNA200AREA",4360,"None",
"SIM_7_Jun_04","200-E-100","liquid",1945,"SD","volume",0.89,"UNA200AREA",4360,"None",
"SIM_7_Jun_04","200-E-100","liquid",1945,"V0600","volume",4.61,"UNA200AREA",4360,"None",
"SIM_7_Jun_04","200-E-100","liquid",1945,"Mean","volume",4.38,"UNA200AREA",4360,"None",
"SIM_7_Jun_04","200-E-100","liquid",1945,"V0005","volume",2.40,"UNA200AREA",4360,"None",
"SIM_7_Jun_04","200-E-100","liquid",1945,"V0995","volume",6.35,"UNA200AREA",4360,"None",
"SIM_7_Jun_04","200-E-100","liquid",1945,"V0550","volume",4.49,"UNA200AREA",4360,"None",
"SIM_7_Jun_04","200-E-100","liquid",1945,"V0650","volume",4.73,"UNA200AREA",4360,"None",
"SIM_7_Jun_04","200-E-100","liquid",1945,"V0700","volume",4.87,"UNA200AREA",4360,"None",
"SIM_7_Jun_04","200-E-100","liquid",1945,"V0750","volume",5.02,"UNA200AREA",4360,"None",
"SIM_7_Jun_04","200-E-100","liquid",1945,"V0800","volume",5.18,"UNA200AREA",4360,"None",
"SIM_7_Jun_04","200-E-100","liquid",1945,"V0850","volume",5.37,"UNA200AREA",4360,"None",
"SIM_7_Jun_04","200-E-100","liquid",1945,"V0900","volume",5.59,"UNA200AREA",4360,"None",
"SIM_7_Jun_04","200-E-100","liquid",1945,"V0950","volume",5.87,"UNA200AREA",4360,"None",
"SIM_7_Jun_04","200-E-100","liquid",1945,"V0450","volume",4.26,"UNA200AREA",4360,"None",
"SIM_7_Jun_04","200-E-100","liquid",1945,"V0500","volume",4.38,"UNA200AREA",4360,"None",
```

19.2.2 Output Files

The INGRES program writes four output files. One file is a report file that contains text information about the run of the code. The report file is always named “inproc.out”. Searching for the text strings “error”, “warning”, or “skip” in the report file is useful for helping identify data set inconsistencies.

The disposal action file contains information about every disposal after all statistical, estimation, or surrogate rules have been applied. The information is given as a statistical distribution for volume and statistical distributions for the concentrations of each analyte. There is a separate DISPOSAL keyword for every combination of waste site, waste stream type, and year of disposal. This file is read and processed in the INVENTORY code. The first few lines of a disposal action file are provided in Table 19.4.

Table 19.4 Excerpts from a Disposal Action File Written by INPROC

```
TITLE "2004 Composite Analysis Baseline Inventory"
USER "Paul W. Eslinger"
DISPOSAL WASTEID "100-H-5" TYPE "soil"  YEAR 1953
 VOLUME   6  5.87600E+03 7.34500E+03 8.81400E+03 TRUNCATE  0.000 1.000
 CONTAM1  "H3    "  9 -9.85732E+00 1.26864E+00 TRUNCATE  0.010 0.990
 CONTAM2  "C14   "  9 -8.97413E+00 1.26864E+00 TRUNCATE  0.010 0.990
 CONTAM6  "Sr90  "  9 -1.08459E+01 1.26864E+00 TRUNCATE  0.010 0.990
 CONTAM8  "Cs137 "  9 -8.95037E+00 1.26864E+00 TRUNCATE  0.010 0.990
 CONTAM9  "Eu152 "  9 -5.40365E+00 1.26864E+00 TRUNCATE  0.010 0.990
 CONTAM14 "U235  "  9 -2.95175E+01 1.26864E+00 TRUNCATE  0.010 0.990
 CONTAM15 "U238  "  9 -2.47347E+01 1.26864E+00 TRUNCATE  0.010 0.990
DISPOSAL WASTEID "100-K-2" TYPE "soil"  YEAR 1955
 VOLUME   6  5.87600E+03 7.34500E+03 8.81400E+03 TRUNCATE  0.000 1.000
 CONTAM1  "H3    "  9 -9.85732E+00 1.26864E+00 TRUNCATE  0.010 0.990
 CONTAM2  "C14   "  9 -8.97413E+00 1.26864E+00 TRUNCATE  0.010 0.990
 CONTAM14 "U235  "  9 -2.95175E+01 1.26864E+00 TRUNCATE  0.010 0.990
 CONTAM15 "U238  "  9 -2.47347E+01 1.26864E+00 TRUNCATE  0.010 0.990
```

The summary file provides a summary of the application of the estimation and surrogate data rules. This file contains information about every site, waste type at the site, waste volume, analyte, and year of disposal. The first few records from a summary file are provided in Table 19.5. The numerical values are point values for the respective action. Lines of data where an input distribution was supplied has a last entry of “inp”. Lines of data filled in from a surrogate site has a last entry of “sur”. Lines of data filled in using an estimation rule have a last entry of “est”.

Table 19.5 Excerpts from a Summary Action File Written by INPROC

```
Derived-Records, 100-H-5, soil, 1953, volume,  7.34500E+03, inp
Derived-Records, 100-H-5, soil, 1953, H3    ,  5.23627E-05, inp
Derived-Records, 100-H-5, soil, 1953, C14   ,  1.26645E-04, inp
Derived-Records, 100-H-5, soil, 1953, Sr90  ,  1.94838E-05, inp
Derived-Records, 100-H-5, soil, 1953, Cs137 ,  1.29689E-04, inp
Derived-Records, 100-H-5, soil, 1953, Eu152 ,  4.50015E-03, inp
Derived-Records, 100-H-5, soil, 1953, U235  ,  1.51608E-13, inp
Derived-Records, 100-H-5, soil, 1953, U238  ,  1.81080E-11, inp
Derived-Records, 100-K-2, soil, 1955, volume,  7.34500E+03, sur
Derived-Records, 100-K-2, soil, 1955, H3    ,  5.23627E-05, sur
Derived-Records, 100-K-2, soil, 1955, C14   ,  1.26645E-04, sur
```

```
Derived-Records, 100-K-2, soil, 1955, Sr90  ,  1.94838E-05, sur
Derived-Records, 100-K-2, soil, 1955, Cs137 ,  1.29689E-04, sur
Derived-Records, 100-K-2, soil, 1955, Eu152 ,  4.50015E-03, sur
Derived-Records, 100-K-2, soil, 1955, U235  ,  1.51608E-13, sur
```

The INPROC code produces a suite of WASTEMAP keywords for use in the INVENTORY code. An example of this output file for the first eight WASTEMAP keywords is provided in Table 19.6. All of the WASTEMAP keywords in this file must be inserted in the keyword file for the INVENTORY code before the INVENTORY code is executed.

Table 19.6 Excerpts from a Aggregate Keyword File Written by INPROC

```
TITLE "2004 Composite Analysis Baseline Inventory"
USER "Paul W. Eslinger"
WASTEMAP AGGREGAT "100-H-5" FRACTION 1 1.0 STREAMS
 "100-H-5                       soil  1953"
WASTEMAP AGGREGAT "100-K-2" FRACTION 1 1.0 STREAMS
 "100-K-2                       soil  1955"
WASTEMAP AGGREGAT "100-K-2" FRACTION 1 1.0 STREAMS
 "100-K-2                       soil  1956"
WASTEMAP AGGREGAT "100-K-2" FRACTION 1 1.0 STREAMS
 "100-K-2                       soil  1957"
WASTEMAP AGGREGAT "100-K-2" FRACTION 1 1.0 STREAMS
 "100-K-2                       soil  1958"
WASTEMAP AGGREGAT "100-K-2" FRACTION 1 1.0 STREAMS
 "100-K-2                       soil  1959"
WASTEMAP AGGREGAT "100-K-2" FRACTION 1 1.0 STREAMS
 "100-K-2                       soil  1960"
WASTEMAP AGGREGAT "100-K-2" FRACTION 1 1.0 STREAMS
 "100-K-2                       soil  1961"
```

19.3 Keyword Definitions for INPROC

The main control for program operation is provided in the keyword control file which always has the name inproc.key. Options and parameters are provided in this file using keywords. This section defines the keywords available for the INPROC program.

In general, the keywords for INPROC can be entered in any order. The following restrictions apply on keyword order:

- The REPORT keyword must be the first keyword in the file.
- The END keyword must be the last keyword in the file.

19.3.1 ANALYTE Keyword for INPROC

The ANALYTE keyword identifies contaminants to be included in the analysis and provides parameter values associated with the contaminant. The following is this keyword’s syntax:

```
ANALYTE [ID=“quote 1”] [TYPE=“quote 2”] [NAME=“quote 3”]
   {COMPUTE} {OUTPUT}
```

The single quote string associated with the modifier ID must contain an analyte identification (limit of six characters in length) from the set of the analytes in the inventory results file. The TYPE modifier is followed by a two-character quote string with the following meaning:

NR – analyte is non-organic radionuclide
NS – analyte is non-organic stable element or compound
OR – analyte is organic radionuclide
OS – analyte is organic stable element or compound

The NAME modifier is followed by a descriptive name for the contaminant in a quote string of up to 72 characters. The presence of the optional COMPUTE modifier causes the contaminant to be included in the analysis. As the ANALYTE keyword records are read, the COMPUTE modifier is searched for first. If it is not found, the record is ignored. The presence of the optional modifier OUPUT causes data for the analyte to be output in the disposal action file. In the following five keyword examples tritium will be ignored and U234 will be calculated but not output:

```
ANALYTE ID "H3"   NAME "Tritium"     TYPE "NR"
ANALYTE ID "C14"  NAME "Carbon-14"   TYPE "NR" COMPUTE OUTPUT
ANALYTE ID "U234" NAME "Uranium-234" TYPE "NR" COMPUTE
ANALYTE ID "U235" NAME "Uranium-235" TYPE "NR" COMPUTE OUTPUT
ANALYTE ID "U238" NAME "Uranium-238" TYPE "NR" COMPUTE OUTPUT
```

19.3.2 ATTRIBUTE Keyword for INPROC

The ATTRIBUTE keyword provides cross-reference information relating the input attribute field names to the output waste type names. The following is this keyword's syntax:

```
ATTRIBUTE [“quote 1”] [TYPE “quote 2”]
```

The first quote string for this keyword contains the name of an attribute field that may be supplied on the input disposal action file. The quote string associated with the TYPE modifier is the waste type to be assigned to the output data set whenever the input attribute field occurs. The following six examples illustrate the use of this keyword:

```
ATTRIBUTE "STF"    TYPE "soilsf"  "segregated trench offsite waste"
ATTRIBUTE "STN"    TYPE "soilsn"  "segregated trench onsite"
ATTRIBUTE "TCN"    TYPE "cmntct" "segregated TRU caisson onsite"
ATTRIBUTE "UTN"    TYPE "soiltu" "unsegregated trench onsite"
ATTRIBUTE "tle"    TYPE "liquid" "Liquid releases from tanks"
ATTRIBUTE "res"    TYPE "cake  " "Waste inside the tanks"
```

19.3.3 END Keyword for INPROC

The END keyword signifies the end of all keyword data. All data in the keyword file after the END keyword will be ignored. The following is this keyword's syntax:

```
END
```

19.3.4 ESTIMATE Keyword for INPROC

The ESTIMATE keyword provides factors to estimate concentrations of analytes not present in a waste disposal action. This feature is used when an analyte is expected to be present, but no information is available in the database. The estimate is based on other analytes that are present. The ESTIMATE keyword records can be defined to apply to specific waste sites and years, or they can be applied to all disposal actions. The following is this keyword's syntax:

```
ESTIMATE [ANALYTE “quote1”] [FROM “quote2”] [FACTOR N1]
  [SITES [ALLSITES | {NOTSITES} “quote3” “quote4” … “quoteN”]]
  [YEARS [ALLYEARS|BEFORE N2|AFTER N2|AT N2]]
```

At least one ESTIMATE keyword must be entered even if it is not used in the calculations. The quote string associated with the ANALYTE modifer identifies the analyte for which concentration data are to be estimated. The quote string associated with the FROM modifier identifies the analyte to be used as the source of the estimate. The numerical value N1 associated with the FACTOR keyword is the factor that the concentration of analyte "quote2" is multiplied by to estimate the concentration of analyte "quote1."

The SITES modifier is used to define the sites where the estimation rule is to be applied. If the modifier ALLSITES is present along with the SITES modifier, then specific site IDs are not used, and the method is applied to all sites needing a fill-in rule. When the ALLSITES modifier is not present the strings "quote3" through "quoteN" provide the site IDs where this estimation rule will be applied. If the modifier NOTSITES is present, but not the modifier ALLSITES, then the site names identified in the quote STRINGS are excluded and all other sites are included for application of the ESTIMATE rule.

The years for which the estimation rule applies are provided using the YEARS modifier. If the modifier ALLYEARS is present along with the YEARS modifier, then the factor is applied to all years and the numerical value N2 is not used. Otherwise, specific years of applicability are defined using one of the modifiers BEFORE, AFTER or AT. When these modifiers are used, the value of N2 is used to define the year or range of years for application of the ESTIMATE rule. When BEFORE is present, the rule is applied to all years before N2. When AFTER is present, the rule is applied to all years after N2. When AT is present the rule is applied only to year N2. Only one of the BEFORE, AFTER or AT modifiers can be used (there is an input hierarchy of BEFORE, AFTER, and AT). For example, if BEFORE is present, then AFTER and AT are ignored if they were to be entered. If no year data are provided the code will error terminate.

The order of the entry for multiple ESTIMATE records is important. If more than one method is provided for estimation of the same analyte, then the first estimation rule where the necessary information is available is used. For example, if two methods are given for estimation of Tc99 inventory, the first based on the Sr90 concentration and the second on the I129 concentration, the analysis will use the factor based on the Sr90 concentration (first estimate rule) if a value for Sr90 concentration is given. If no Sr90 concentration is given for the waste site, then the code will attempt to estimate the Tc99 concentration from the I129 concentration (second estimation rule). If there is no concentration for either Sr90 or I129, then no estimate will be made.

For example, the following five ESTIMATE keywords can be used to estimate fission product concentrations from fuel rod failures in N reactor fuel using Cs137 as the known analyte:

```
   ESTIMATE ANALYTE "Se79"  FROM "Cs137" FACTOR 2.476E-7 ALLTIMES
      SITES "116-N-1" "116-N-3"
   ESTIMATE ANALYTE "Sr90"  FROM "Cs137" FACTOR 9.160E-1 ALLTIMES
      SITES "116-N-1" "116-N-3"
   ESTIMATE ANALYTE "Tc99"  FROM "Cs137" FACTOR 1.418E-4 ALLTIMES
      SITES "116-N-1" "116-N-3"
   ESTIMATE ANALYTE "I129"  FROM "Cs137" FACTOR 3.041E-7 ALLTIMES
      SITES "116-N-1" "116-N-3"
   ESTIMATE ANALYTE "Eu152" FROM "Cs137" FACTOR 1.320E-5 ALLTIMES
      SITES "116-N-1" "116-N-3"
```

19.3.5 FILE Keyword for INPROC

The FILE keyword is used to enter the names of all files except the input keyword file and the report file. Those two files always have the names “inproc.key” and “inpoc.out”, respectively. The following is this keyword’s syntax:

```
   FILE modifier1 “quote1” ... modifier5 “quote5”
```

The file names are entered in quote strings, which must be enclosed in double quotes. Path names up to 200 characters long are supported. The modifiers associated with the FILE keyword are given in Table 19.7. The file name associated with a modifier must be entered before the next modifier is entered.

Table 19.7 Modifiers Associated with the FILE Keyword in INPROC

Modifier	Description of Use
AGGREGAT	Output file containing aggregation keyword data output for use in the inventory code.
DATA	Input file containing inventory disposal data (comma separated format).
SITE	Input file containing waste disposal site aggregation rules and default waste types.
SUMMARY	Output file containing point values for all disposal actions and data source indicators.
WASTE	Output file containing stochastic definitions of disposal actions for all waste streams, contaminants, and years.

At least one FILE keyword is required for every run of the code and multiple keyword entries can be used. An example entry is the following:

```
   FILE DATA=INPROC_Input.csv” SITE=“Aggr_site.csv” SUMMARY=“CA1-stoch.out”
      AGGREGAT“CA1-aggregat.key” WASTE=“CA1-stoch.daf”
```

The following set of five entries has the same effect as the single entry above.

```
   FILE DATA     “INPROC_Input.csv”
   FILE SITE     “Aggr_site.csv”
   FILE SUMMARY  “CA1-stoch.out”
   FILE AGGREGAT “CA1-aggregat.key”
   FILE WASTE    “CA1-stoch.daf”
```

19.3.6 MATCH Keyword for INPROC

The MATCH keyword identifies requirements on input data handling for data out of the range of analysis specifications. The following is this keyword's syntax:

```
   MATCH {ANALYTES} {SITES} {TYPES} {YEARS}
```

Requirements can be specified to force agreement with disposal site names, waste types, analyte identifications and years of analysis. If a data set on the input disposal action file has one of these parameters not specified from other input sources, then the code will write an error message and stop execution. This keyword is optional. If this keyword is not used than the INPROC code will ignore any extra data inputs. For example, it will only process data for the analytes specified in the ANALYTE keyword even though the input file contains data on other analytes.

At least one modifier must be present on each MATCH keyword record. The modifiers associated with the FILE keyword are given in Table 19.8.

Table 19.8 Modifiers Associated with the MATCH Keyword in INPROC

Modifier	Description of Use
ANALYTES	When present, this modifier forces the requirement that an analyte in the INPROC input disposal action file must also be specified as being used in INPROC (see ANALYTE keyword description). Detection of an analyte not specified with the ANALYTE keyword will cause run termination if this modifier is present, otherwise the data will be ignored.
SITES	This modifier initiates checking whether a site name in the INPROC input disposal action file is in the master list of waste site names (see the SITE modifier for the FILE keyword). Detection of a site not on the master list will cause run termination if this modifier is present, otherwise the data will be ignored.
TYPES	This modifier initiates checking whether a waste type name in the INPROC input disposal action file is in the master list of waste type names (see the ATTRIBUT keyword). Detection of a waste type not specified using the ATTRIBUT keyword will cause run termination if this modifier is present, otherwise the data will be ignored.
YEARS	This modifier initiates checking whether a disposal action year in the INPROC input disposal action file is in the range of desired years (see the PERIOD keyword). Detection of a year outside the desired range given on the PERIOD keyword will cause run termination if this modifier is present, otherwise the data will be ignored.

Example uses of this keyword to enforce data checking for sites, waste types, and disposal years is the following:

```
   MATCH SITES TYPES
   MATCH YEARS
```

19.3.7 PERIOD Keyword for INPROC

The PERIOD keyword identifies the start and stop times for the entire simulation. The following is this keyword's syntax:

```
   PERIOD [START=year₁]  [STOP=year₂]  [CLOSURE=year₃]
```

The modifier START and the value $year_1$ identify the start of the simulation period. The start of the simulation period must be 1944 or later or the inventory code will error terminate. The modifier STOP and the value $year_2$ identify end of the simulation period. Start and stop years should be entered as whole numbers with the stop year no smaller than the start year. The modifier CLOSURE and the value $year_3$ identify the year that site closure occurs. The year of site closure cannot be smaller than the start year. The following is an example PERIOD keyword that simulates from 1944 through 3050 with site closure occurring at 2050:

```
   PERIOD START=1944 STOP=3050 CLOSURE 2050
```

19.3.8 STATRULE Keyword for INPROC

The STATRULE keyword is used to define statistical distribution rules to convert available parameter information into statistical distributions for output (Table 19.9). The following is this keyword's syntax:

```
   STATRULE  DATASET “setname” {TRUNCATE P1 P2}
      [POINT|MEAN|MEDIAN|MODE|CONSTANT|INPUT][“ANALYTE”|“VOLUME”]
      [[APPLY {PASSTHRU}] “CONSTANT” | “NORMAL” SD | “LOGE” SD |
      “LOG10” SD | “UNIFORM” UMIN UMAX | “DISCRETU” UMIN UMAX |
      “TRIANGLE” FMIN FMAX | “USER” ]]
      [[YEARS] [ALLTIMES|BEFORE T1|AFTER T2|AT T3]]
```

Each STATRULE keyword record provides information for one statistics rule. Multiple rules can be defined. The rules are tested for applicability to a data set in the order they are entered in the keyword file. If the conditions of the rule are satisfied for data representing a given parameter (a POINT value is given in the correct time range, for example), then the rule is used for the data point.

The quote string associated with the DATASET modifier must contain the name of a data set ID in the inventory disposal action data. Use of this modifier allows specification of rules specific to the data sets in the database. The DATASET modifier is mandatory. If a rule is to be applied to all data sets, then the data set name “all” can be entered.

One of the modifiers CONSTANT, INPUT, MEAN, MEDIAN, MODE, or POINT are used to indicate the type of input distribution for which the rule applies. These rules are applied to the data in the inventory disposal data file. The modifier INPUT is used to use the input values directly without modification. When a data input is found that matches the modifier on the statistics rule, then the rule may be applied if other constraints are met (correct data set and year of application, for example). Each of the modifiers CONSTANT, INPUT, MEAN, MEDIAN, MODE, or POINT must be followed by a quote string containing either “ANALYTE” or “VOLUME” but not both. This quote string identifies whether the rule applies to analyte amounts or to volumes. If PASSTHRU is present, the data are treated as concentrations and are passed through to output without modification.

Table 19.9 Statistical Distributions on the STATRULE Keyword for INPROC

Modifier	Associated Parameters and Description of Use
CONSTANT	The data in the inventory data disposal file are treated as constants. No statistical distribution is applied to these data records.
UNIFORM	A uniform distribution is defined using the single datum in the inventory data disposal file. The lower limit is the product of the datum and the value UMIN. The upper limit is the product of the datum and the value UMAX.
DISCRETU	A discrete uniform distribution (range of integer values) is defined using the single datum in the inventory data disposal file. The lower limit is the product of the datum and the value UMIN. The upper limit is the product of the datum and the value UMAX.
LOGU10	A loguniform distribution (base 10) is defined using the single datum in the inventory data disposal file. The lower limit is the product of the datum and the value UMIN. The upper limit is the product of the datum and the value UMAX.
LOGUE	A loguniform distribution (base e) is defined using the single datum in the inventory data disposal file. The lower limit is the product of the datum and the value UMIN. The upper limit is the product of the datum and the value UMAX.
TRIANGLE	The single datum in the inventory data disposal file is treated as the mode of a triangular distribution. The minimum of the triangular distribution is the product of FMIN and the mode. The maximum of the triangular distribution is the product of FMAX and the mode.
NORMAL	The single datum in the inventory data disposal file is treated as coming from a normal distribution. The datum provides the mean of the normal distribution. The numerical value SD is applied as the standard deviation of the distribution.
LOG10	The single datum in the inventory data disposal file is treated as coming from a lognormal distribution (base 10). The datum provides the arithmetic mean of the lognormal distribution. The numerical value SD is applied as the arithmetic standard deviation of the distribution.
LOGE	The single datum in the inventory data disposal file is treated as coming from a lognormal distribution (base e). The datum provides the arithmetic mean of the lognormal distribution. The numerical value SD is applied as the arithmetic standard deviation of the distribution.

An optional specification of years for application of the rule is handled with the YEARS modifier. If YEARS is absent, the rule is applied to all years. Use of the YEARS modifier followed by the ALLTIMES modifer also applies the statistics rule for all times. The numerical value (integer year) associated with the optional BEFORE modifier specifies that the statistics rule will be applied to all years before the year provided. The numerical value (integer year) associated with the optional AT modifier specifies that the statistics rule will be applied to the one year provided. The numerical value (integer year) associated with the optional AFTER modifier specifies that the statistics rule will be applied to all years after the year provided.

If the TRUNCATE modifier is present then the distribution will have the TRUNCATE option activated with the parameter values P1 and P2 being the lower and upper truncation limits. Truncation limits are defined as tail probabilities rather than limits on the data values. The truncation is only valid for the following distributions: uniform, loguniform, normal, lognormal (base e), lognormal (base 10), and triangular.

Two examples of STATRULE keyword records that apply to volumes are the following:

```
   STATRULE DATASET “ALL” POINT “VOLUME” APPLY “TRIANGLE” 0.8 1.2
      TIMES ALLTIMES
   STATRULE DATASET “ALL” MEAN “ANALYTE” APPLY “LOGE” 2.0 TIMES BEFORE 1970
```

The first keyword indicates that volume data with a statistical parameter name of “point” will be assigned a triangular distribution with the point value being the mode, the minimum will be 0.8 times the point value, and the maximum will be 1.2 times the point value. The rule will be applied to all time periods. The second keyword indicates that volume data with a statistical parameter name of “mean” will be assigned a lognormal (base e) distribution with the input value being used as the arithmetic mean and the arithmetic standard deviation will be set to 2.0. The rule will be applied for times before 1970.

The following two examples apply rules to a specific dataset titled “SIM_7_Jun_04”.

```
   STATRULE DATASET "SIM_7_Jun_04" INPUT  "VOLUME" PASSTHRU
      APPLY "USER" 0.80 1.2 YEARS ALLTIMES
   STATRULE DATASET "SIM_7_Jun_04" INPUT "ANALYTE" PASSTHRU
      APPLY "USER" 0. 0. YEARS ALLTIMES
```

The following two examples apply rules to all data sets for different time periods and also illustrate the use of the truncation option.

```
   STATRULE DATASET "ALL" MEAN   "ANALYTE" APPLY "LOGE" 2.00
      YEARS BEFORE 1970 TRUNCATE .01 .99
   STATRULE DATASET "ALL" MEAN   "ANALYTE" APPLY "LOGE" 0.25
      YEARS AFTER 1969  TRUNCATE .01 .99
```

19.3.9 SURROGATE Keyword for INPROC

The SURROGATE keyword provides information on use of surrogate sites to provide analyte concentration estimates for sites having no analyte data. The following is this keyword’s syntax:

```
   SURROGATE UNKSITE=“quote 1” N1 N2 SURSITE=“quote 2” N3 “quote 3”
```

The ID (“quote 1”) of the site for which data are to be generated is associated with the modifier UNKSITE. The two numerical values associated with the modifier UNKSITE are the volume of the waste disposed to the site with missing data (N1, m^3) and the year the disposal action occurs (N2, calendar year). The ID (“quote 2”) of the surrogate site from which analyte data are to be used is associated with the modifier SURSITE . The numerical value associated with SURSITE is the year of disposal for the source surrogate data., and “quote 3” is the waste type of the site providing the surrogate data. If the input data file does not contain data for the reference site, waste type, and year, then the unknown site disposal action is not generated and no error message is written.

An example SURROGATE keyword record is as follows:

```
SURROGAT UNKSITE “216-W-32” 1965 30.0 SURSITE “216-W-5” 1964 “liquid”
```

The record indicates data for site 216-W-5 in year 1964 and waste type “liquid” will be used to estimate the disposal action to site 216-W-32 for year 1965, with the disposal volume being 30 m^3. The waste type for the unknown site is always the same as for the surrogate site (“liquid” in the example).

19.3.10 SUM Keyword for INPROC

The SUM keyword allows radionuclide concentration addition such as summing over all uranium isotopes. The following is this keyword’s syntax:

```
SUM [TO “quote 1”] [FROM N1 “quote 2” {“quote 3”, … “quote N1”}]
```

The record specifies the name of the analyte representing the output analyte, and the names of all analytes to be included in the sum.

The quote string associated with the TO modifier contains the ID for the output analyte. The numerical value associated with the FROM modifier indicates the number of analytes to use in the sum. The quote strings associated with the FROM modifier provide the analyte ID’s used in the sum. An example record that outputs U-238 as the sum of data for U-234 and U-238 is the following:

```
SUM TO “U238” FROM 2 “U234” “U238”
```

19.3.11 VERBOSE Keyword for INPROC

The presence of the optional VERBOSE keyword initiates additional output to the report file. The following is this keyword’s syntax:

```
VERBOSE
```

If this keyword is present the report file will contain a listing of the actions by analyte (data distribution, estimated distribution, or analyte not included) for every data set. In addition, all surrogate sites that use a specific waste stream from a source site are identified.

20.0 Inv_Med – Median Values for Inventory

20.1 Overview

A run of the INVENTORY code with stochastic parameters set to the median value (50th percentile) of their range requires an input disposal action file designed for a single realization. The disposal action file developed by the INPROC computer code contains stochastic distributions for volumes and concentrations and is intended to support multiple realization runs of the INVENTORY code. A sequence of computer codes is available that reads a full stochastic disposal action file and writes the associated median-values disposal action file.

20.1.1 Location in the Processing Sequence

The disposal action file is an output of the INPROC code and an input to the INVENTORY code. Therefore, conversion of the stochastic disposal action file into a median-values disposal action file must occur after a run of the INPROC code but before a run of the INVENTORY code.

20.1.2 How the Code Is Invoked

A sequence of three codes is used to convert the stochastic disposal action file into a median-values disposal action file. For purposes of discussion, let the stochastic disposal action file be named "stochastic.daf" and let the median-values disposal action file be named "median.daf." The sequence of three codes is invoked as follows:

```
inv_step1.exe "stochastic.daf"
stochastic.exe "Step1.Key"
inv_step2.exe "stochastic.daf" "median.daf"
```

The user does not need to prepare any input files or enter other commands to control this sequence of calculations.

20.1.3 Memory Requirements

Inv_step1 and Inv_step2 have minimal memory requirements. Stochastic.exe uses 20 megabytes of memory for a run of 107 years, 988 waste sites, and 40 waste types.

20.2 Computational Sequence

A sequence of three codes is used to convert the stochastic disposal action file into a median-values disposal action file. Invocation of the three codes are described in the following paragraphs.

20.2.1 Inv_Step1 Execution

The first step in the processing sequence is to run the INV_STEP1 code. This code reads the stochastic disposal action file and writes a keyword file for the STOCHASTIC code. The user does not need to modify the output keyword file.

Under the Windows operating system (Releases 98, NT, 2000, or XP), INV_STEP1 executes in a DOS box. A run of INV_STEP1 is initiated by entering the following command line:

```
inv_step1 “stochastic.daf”
```

Under the Linux operating system INV_STEP1 is executed through any of the following Bourne Shell or C Shell commands:

```
inv_step1.exe “stochastic.daf”
```

For these commands, INV_STEP1 or inv_step1.exe is the name of the executable program and “stochastic.daf” is the name of the stochastic disposal action file.

20.2.2 Stochastic Execution

The second step in the processing sequence is to run the STOCHASTIC code. This code reads the file named “Step1.Key” written by the INV_STEP1 code and writes a file named “Step1.Val” for use in the INV_STEP2 code.

Under the Windows operating system (Releases 98, NT, 2000, or XP), STOCHASTIC executes in a DOS box. A run of STOCHASTIC is initiated by entering the following command line:

```
STOCHASTIC “Step1.Key”
```

Under the Linux operating system STOCHASTIC is executed through any of the following Bourne Shell or C Shell commands:

```
stochastic.exe “Step1.Key”
```

For these commands, STOCHASTIC or stochastic.exe is the name of the executable program, “Step1.Key” is the name of the input keyword file and “Step1.Val” is the name of the output file of median values.

20.2.3 Inv_Step2 Execution

The final step in the processing sequence is to run the INV_STEP2 code. This code reads the stochastic disposal action file, a file named “Step1.Val” written by the STOCHASTIC code, and writes the median-values disposal action file.

Under the Windows operating system (Releases 98, NT, 2000, or XP), INV_STEP2 executes in a DOS box. A run of INV_STEP2 is initiated by entering the following command line:

```
inv_step2 “stochastic.daf” “median.daf”
```

Under the Linux operating system INV_STEP2 is executed through any of the following Bourne Shell or C Shell commands:

```
inv_step2.exe “stochastic.daf” “median.daf”
```

For these commands, INV_STEP2 or inv_step2.exe is the name of the executable program, “stochastic.daf” is the name of the stochastic disposal action file and “median.daf” is the name of the median-values disposal action file.

21.0 Inventory – Inventory Data Base

21.1 Overview

The SAC inventory database contains information derived from Hanford records, simulation of past Hanford processes (SIM – Soil Inventory Model), and projection of future waste generation (HTWOS – Hanford Tank Waste Operations Simulator). The structure of the SAC inventory database and the site naming conventions were derived from WIDS (Waste Information Data System). The SAC inventory database uses a modified version of the WIDS Site table as the basis for organizing waste site information. The SAC inventory database was developed using Microsoft Access.

21.1.1 Location in the Processing Sequence

The SAC inventory database provides input for the INPROC utility code. It is the first processing step of any SAC simulation.

21.1.2 Memory Requirements

The current size of the inventory database is 860 megabytes. The memory required to run queries should not exceed 128 megabytes.

21.2 Database Structure

The structure of the SAC Inventory Database was derived from WIDS. WIDS is a database operated by Fluor Hanford to track all Hanford Site solid waste management units, as required by the Tri-Party Agreement (the Hanford Federal Facility Agreement and Consent Order).

The major structure of the Inventory database includes the Site and Waste tables from WIDS, plus a table of inventory values compiled for SAC. The relationships between the Site table and the Waste and Inventory tables are shown in Figure 21.1.

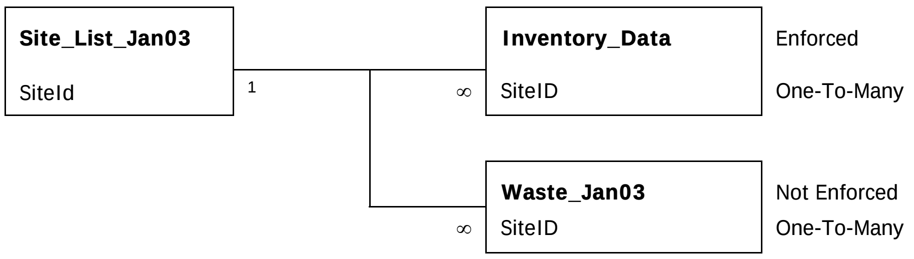


Figure 21.1 Relationships between Site, Waste, and Inventory Tables

Waste site names and unique identifying numbers are adapted from WIDS. The Site table (adapted in this database as Site_List_Jan03) is the foundation of the WIDS database. This table lists each site identified as a potential waste site, categorizes it as to type of site (crib, process facility, unplanned release, etc.), provides descriptions related to location, release, process , and activities at each site. The WASTE table (Waste_Jan03) was downloaded from WIDS in January 2003, primarily to supply descriptive details to aid in determining the relevance of each site and the potential magnitude of contamination associated with each site. The Inventory_Data table, which is unique to the SAC Inventory, is the repository of all collected record data and simulation results used as input to the Inventory code.

The fields in the Site, Inventory, and Waste tables are listed in Tables 21.1, 21.2, and 21.3, respectively.

Table 21.1 Field Descriptions for Inventory Table Site_List_Jan03

Non-WIDS?	**Field Name**	**Type**	**Size**	**Field Description**
	SiteId	Long Integer	4	
	SiteCode	Text	15	
	SiteNames	Text	255	
	SiteType	Text	50	
	SiteStatus	Text	20	
	DesgArea	Text	20	
	OperUnit	Text	10	
	SiteClassificationStatus	Text	20	
	SitePackageStatus	Text	20	
	SiteReclassStatus	Text	25	
	SiteDesc	Memo	-	
	AssocStruct	Memo	-	
	SiteComment	Memo	-	
	ReleaseDesc	Memo	-	
	ReleasePotDesc	Memo	-	
	EnvMonDesc	Memo	-	
	ProcessDesc	Memo	-	
	StartDate	Double	8	
	EndDate	Double	8	
	ValidatedBy	Text	40	
	ValidatedDate	Date/Time	8	
	AssignedToFieldTech	Text	50	
	DateSiteIdAssigned	Date/Time	8	
	InvestigationSubmissionDate	Date/Time	8	
	InvestigationAssignedDate	Date/Time	8	
	LocDesc	Memo	-	
	PrioritizationLevel	Integer	2	
	SiteIdAssignedBy	Text	50	
	SiteActivities	Memo	-	
	PreviousOperableUnit	Text	10	
	ReclassStatusDate	Date/Time	8	
	OperUnitPrintOrder	Double	8	
	SiteCodePrintOrder	Double	8	
X	Site-Volume?	Yes/No	1	Does the site have a waste volume assigned?
X	Site-specific_Inventory?	Yes/No	1	Does the site have site-specific inventory?
X	Inv_consolidated-other_Site?	Yes/No	1	Has inventory been consolidated at another site?
X	Other_Site_containing_Inv	Text	50	Name of other site containing inventory

Non-WIDS?	Field Name	Type	Size	Field Description
X	Screen_class-reclass	Yes/No	1	Does the site pass screening based on classification or reclassification?
X	Screen_SiteType	Yes/No	1	Is the site of a selected site type?
X	Associated_with_Duplicate?	Yes/No	1	Is the site associated with a duplicate?
X	Is_Duplicate?	Yes/No	1	Is this site the duplicate of another?
X	Duplicate_of_SiteCode	Text	50	Site Code of duplicate site
X	Duplicate_info_fields	Text	50	
X	No_Waste_Received?	Yes/No	1	Has this site received no waste?
X	Consolidated_Site?	Yes/No	1	Has the site been consolidated with other(s)?
X	Other_site_name	Text	50	Site of consolidation
X	Selected?	Yes/No	1	Is this site selected for inventory consideration?
X	Attributed_elsewhere?	Yes/No	1	Is the inventory for this site attributed elsewhere?
X	Selected_surrogate?	Yes/No	1	Has a surrogate been selected for site w/o inventory?

Table 21.2 Field Descriptions for Inventory Table Inventory_Data

Name	Type	Size	Description
Index	Long Integer	4	Autonumber field
SiteID	Long Integer	4	Site index from WIDS; additions numbered from 9900
SiteCode	Text	255	Descriptive Site name
Stat_Distribution	Text	50	Distribution parameter
Subsite	Text	50	Waste description or index
Attribute2	Text	50	Additional desc; used only for offsite
Year	Long Integer	4	Year of contaminant deposit or basis
Constituent_name	Text	255	Radionuclide or chemical name
Constituent_value	Double	8	quantity
Constituent_units	Text	50	Ci, kg, or m3; SIM Ci/m3 or kg/m3
Source	Text	50	Origin of data (e.g. records, HTWOS, SIM)
Dataset	Text	50	Data package or file for traceability
Comment	Text	50	Comment – history of value

Note: **Attribute2** is given a value of “none” for sites with waste disposed at Hanford, which includes all SIM sites. For waste with offsite destination (such as WIPP, YUCCA, or SNM storage), a new site name is created in Attribute2. The new name is a combination of the offsite destination and the original SiteCode.

Table 21.3 Field Descriptions for Inventory Table Waste_Jan03

Name	Type	Size	Field Description
WasteId	Long Integer	4	
SiteId	Long Integer	4	
Type	Text	30	
Category	Text	50	
Amount	Double	8	
Units	Text	20	

Name	Type	Size	Field Description
StartDate	Double	8	
EndDate	Double	8	
ReportedDate	Double	8	
PhysicalState	Text	20	
WasteDesc	Memo	-	
WasteObscured	Text	255	
Screen_WasteCategory	Yes/No	1	Does waste category pass screening criteria?
Screen_WasteType	Yes/No	1	Does waste type pass screening criteria?

21.3 Importing Data and Results of Simulations

There are three major sources of values in the inventory database. Information concerning waste disposals or releases at Hanford from published documents, contractor records, or other databases are termed **Record** data. Simulations have been used to estimate disposal of some liquid wastes (the Soil Inventory Model, or **SIM**). Future waste from processing tank waste is projected by Hanford Tank Waste Operations Simulator (**HTWOS**).

21.3.1 Importing RECORD Data

Record data are derived from various Hanford reports or databases and are collected into data packages. These data packages usually contain detailed instructions for adding new records and for deleting obsolete records when necessary. The data package may include a text document containing import instructions, a spreadsheet containing a list of values to be added or deleted, and may have a database to be incorporated into the Inventory_Data table.

21.3.2 Importing SIM Files

Data from the Soil Inventory Model (SIM) (Simpson 2001) form the basis for the inventory for many sites that incurred liquid releases. A pre-processing module SIMS (see Section 1.0) reformats the SIM data into a suite of comma-separated variable files ready for import into the datadbase. The data are grouped into files by area or facility, with filenames such as “Bplant” and “Tfarm”. The files contain a stochastic representation for volume and concentrations (with 23 values from a user defined statistical distribution) for each analyte. Table 21.4 lists the seven fields included in each record from SIM.

Table 21.4 Fields included in SIM input to the Inventory Database

Field Name	Description	Example
Source	General description of the record	SIM5_24_Feb_
SiteCode	Name of a site, usually indicating designated area and type	216-A-25 (216 indicates liquid; -A indicates 200-East)
Subsite	Describes the waste type or release model	always “liquid” for SIM input
Year	Year of waste deposit or discharge	1945

Field Name	Description	Example
Stat_distribution	Name of the statistical distribution parameter	“mean”; “V0950”, the 95th percentile value
Constituent_name	Name of the chemical element, compound, radioisotope, or “volume”	I129, Tc99, U238, U, CCl4 or CrVI, volume
Constituent_value	Concentration of the constituent	A number; units are Ci/m^3 for radioactive analytes and kg/m^3 for other analytes

A field named **Filename** is added to the SIM output listed above to provide traceability to the records and to match the format of records from other sources. **Source** describes the origin of the record in a general way, whereas **Filename** identifies the particular file from which the record was derived. Constituent units are added, because SIM provides values as concentrations. SiteID, a numeric index corresponding to SiteCode, is added out of a sense of conscience because it should have been used as a primary index of SiteCode. A comment field is also included. Table 21.5 lists fields added to SIM input.

Table 21.5 Fields added to SIM input in the Inventory Database

Field Name	Description	Example
Filename	Identifier based on the filename of record origin, added to make the data easier to trace.	“Bplant” or “Tfarm”
Constituent_Units	Concentration units, different from other data sources.	Chemicals: kg/m^3 Radionuclides: Ci/m^3 Volume: m3
SiteID	Index which should be used to designate a unique site, with reference to SiteCode only in the Site table.	348 (216-A-25)
Attribute2	Used to identify wastes to be disposed offsite. Value is “none” for sites with waste disposed at Hanford, which includes all SIM sites.	“none”
Comments	Remarks about the history of the values	

Detailed instructions for creating a table of SIM data to import into inventory database are as follows:

1. Open a new blank database in Access.
2. Import each .csv file, using File; Get External Data; Import, select file from the browser, and follow instructions for importing.
3. Create a new table (e.g. SIM5), in which to combine information from all fields of the individual files (the target table containing the combined SIM list)
    a. Build a query to show all fields of the first SIM file (e.g., 200EPonds)
    b. Use “Make-Table query” and name the new Table (SIM*n*)
    c. Save the query as a “select query”, to use as a template when adding data from other files to the target file. (Name it Add_data, or some such name to distinguish it from Update files)
4. Add a new field, **“Filename”** to the new combined (SIM5) Table:
    a. Select the combined table, and choose design view
    b. Insert a row into the table design, and add “Filename”.

   c. Save and close the design view.
5. Update the new blank Filename field to the name of the file that was imported by creating an "Update Table Query"
   a. Create a **template Update Query** to change the Filename field
   b. Use for Filename criterion = "is null" to select records to revise. Update to the newly imported filename (e.g. 200EPonds)
   c. Name the query (Update_filename).
   d. Save the query as a Select Query to avoid unintentional updates.
6. Import the next new data file (e.g. 200WPonds)
   a. Open the Add_Table query in design view. Select the next table to be added.
   b. Change the Table designation in Design View to the name of the new table.
   c. Select "Append-Query" to add data to the combined, or target table.
   d. Create **template Append Query**; each revision, add new file, change table names in Design View, delete old file from design.
   e. Run the Append Query to add data to SIM5; save query as a "select" query (to avoid running an action query by accident)
7. Repeat steps 3 through 6, using template queries to add the new records and update filenames.

The next step is to modify the **SIM** table as follows:

1. Update the **Subsite** field to "liquid". The **Subsite** file**,** which identifies the release type, must be added to the combined file to conform to the structure of the Inventory_Data table. Currently (by tradition) this information is imported as "**Attribute2**".
2. Update the **Attribute2** field to "none".
3. Update the **Constituent_Units** field to include the data units. There are no units given with the SIM input. Use the following update query for chemical constituents:
   ```
   UPDATE Inventory_Data SET Inventory_Data.Constituent_units = "kg/m3"
   WHERE (((Inventory_Data.Constituent_name)="CCl4" Or
   (Inventory_Data.Constituent_name)="CrVI" Or
   (Inventory_Data.Constituent_name)="NH3" Or
   (Inventory_Data.Constituent_name)="NO2" Or
   (Inventory_Data.Constituent_name)="NO3" Or
   (Inventory_Data.Constituent_name)="U") AND
   ((Inventory_Data.Constituent_units) Is Null));
   ```
   Use the following update query for volume data:
   ```
   UPDATE Inventory_Data SET Inventory_Data.Constituent_units = "m3"
   WHERE (((Inventory_Data.Constituent_name)="volume") AND
   ((Inventory_Data.Constituent_units) Is Null));
   ```
   The remaining analytes are radionuclides so update the data units to Ci/m^3 using the following update query:
   ```
   UPDATE Inventory_Data SET Inventory_Data.Constituent_units = "Ci/m3"
   WHERE (((Inventory_Data.Constituent_units) Is Null));
   ```

21.3.3 Importing HTWOS Files

Data from HTWOS are imported as two comma-separated value files named “HTWOS_TDP.CSV” and “HTWOS_LEAK.CSV”. These files contain HWTOS data already processed by HTWOS_TLDP (Section 1.0) and HTWOS_TDP (Section 1.0). The first three lines of each input file are shown in Table 21.6 to illustrate the title line and the first record of each file.

Table 21.6 Excerpted Data from HTWOS Tank Loss and Secondary Waste Streams Files

```
! HTWOS Tank Leak data after 2000
SiteCode   Attribute            Attribute2   Year    Code      Value
HTWOS       241-S-106            tle          2006   Volume    3.03E+01

! HTWOS processing data for tanks and HLW/ILAW after 2000
Dataset    Waste Stream         Attribute    Year     Code       Value
 HTWOS     HTWOS-CHE-SOL       cmntH2         2005    mean       Volume     1.92E+00
```

These files need to be edited to match the inventory database input structure. The following steps are needed for each file:

1. Open the file in Excel. Save the file as an Excel worksheet (.xls).
2. Delete first line of data.
3. Add a new column 2 with the entry “Dataset” in the first row. The data in following rows should be entered as “HTWOS_TDP_*mm-yy*” or “HTWOS_LEAK_*mm-yy*” where *mm-yy* refers to the month and year of the input file.
4. Add a new column 6 with the data entry “Stat_Distribution” in the first row. The data in the following rows should all be set to “mean.”
5. Change the column headings to match exactly with the entries in Table 21.7.
6. Modify subsite names as needed. In HTWOS_TDP, “cmntH2” must be changed to “cement”.

Table 21.7 Modified Column Titles for the HWTOS Data Set

Column Number	Column Title (First Row)
1	Source
2	Dataset
3	SiteCode
4	Subsite
5	Year
6	Stat_Distribution
7	Constituent_name
8	Constituent_value

Finally, import these files into the inventory database using the following steps:

1. Open database
2. Import HTWOS_LEAK.xls:
    a. File menu
    b. Select “Get External Data”
    c. Select “Import”
    d. Look in: choose directory c:/VZ03-04/DataPackages/HTWOS/

   e. Change file type to .xls; select *filename (HTWOS_LEAK.xls)*
   f. Show worksheets: *select first page of workbook*
   g. Select "**first row contains column headings**"
   h. Where would you like to store your data? Select **New Table**
3. Import HTWOS_TDP.xls:
   a. File menu
   b. Select "Get External Data"
   c. Select "Import"
   d. Look in: choose directory c:/VZ03-04/DataPackages/HTWOS/
   e. Change file type to .xls; select *filename HTWOS_TDP.xls*
   f. Show worksheets: *select first page of workbook*
   g. Select "**first row contains column headings**"
   h. Where would you like to store your data? Select In an existing table: HTWOS_LEAK
4. Edit the structure of HTWOS_LEAK:
   a. Add fields that are in the Inventory_Data table: (SiteID, Attribute2, Constituent_units, Comment).
   b. Save and close table.
5. Add SiteID to existing sites in the HTWOS_LEAK table using the following query:

```
UPDATE Site_List_Jan03 INNER JOIN HTWOS_LEAK ON Site_List_Jan03.SiteCode =
HTWOS_LEAK.SiteCode SET HTWOS_LEAK.SiteID = [Site_List_Jan03].[SiteId];
```

6. Determine whether there are new site codes in the HTWOS table that need to be added to the site list using the following query:

```
SELECT HTWOS_LEAK.SiteCode, HTWOS_LEAK.SiteID
FROM Site_List_Jan03 RIGHT JOIN HTWOS_LEAK ON Site_List_Jan03.SiteId =
HTWOS_LEAK.SiteID
GROUP BY HTWOS_LEAK.SiteCode, HTWOS_LEAK.SiteID
HAVING (((HTWOS_LEAK.SiteID) Is Null));
```

7. Append a list of new sites and SiteIDs to the site list table (Site_List_Jan03) for each new waste stream in the HTWOS file using the following query:

```
INSERT INTO Site_List_Jan03 ( SiteCode, SiteId )
SELECT HTWOS_LEAK.SiteCode, HTWOS_LEAK.SiteID
FROM Site_List_Jan03 RIGHT JOIN HTWOS_LEAK ON Site_List_Jan03.SiteId =
HTWOS_LEAK.SiteID
GROUP BY HTWOS_LEAK.SiteCode, HTWOS_LEAK.SiteID
HAVING (((HTWOS_LEAK.SiteID) Is Null));
```

8. Make table of new site codes from HTWOS_LEAK (combined) table using the following query:

```
SELECT HTWOS_LEAK.SiteCode, HTWOS_LEAK.SiteID INTO [Add_HTWOS_5-05]
FROM HTWOS_LEAK LEFT JOIN Site_List_Jan03 ON HTWOS_LEAK.SiteCode =
Site_List_Jan03.SiteCode
GROUP BY HTWOS_LEAK.SiteCode, HTWOS_LEAK.SiteID
HAVING (((HTWOS_LEAK.SiteID) Is Null));
```

9. Append new site codes (waste streams) from HTWOS to the site list using the following query:

```
INSERT INTO Site_List_Jan03 ( SiteCode )
SELECT [Add_HTWOS_5-05].SiteCode
FROM [Add_HTWOS_5-05];
```

10. The new SiteID (autonumber) is added to the list by ACCESS. Add values for other fields as needed and update Site Names to site code using the following query:

```
UPDATE Site_List_Jan03 SET Site_List_Jan03.SiteNames =
[site_List_Jan03].[SiteCode]
WHERE (((Site_List_Jan03.SiteId)>9917));
```

11. Update the SiteID in Waste table (WasteID= autonumber, updated automatically) using the following query:

```
UPDATE Site_List_Jan03 LEFT JOIN Waste_Jan03 ON Site_List_Jan03.SiteId =
Waste_Jan03.SiteId SET Waste_Jan03.SiteId = [Site_List_Jan03].[SiteId]
WHERE (((Site_List_Jan03.SiteId)>9900) AND ((Waste_Jan03.SiteId) Is
Null));
```

12. Update Waste Type for new HTWOS streams using the following query:

```
UPDATE (Site_List_Jan03 LEFT JOIN Waste_Jan03 ON Site_List_Jan03.SiteId =
Waste_Jan03.SiteId) LEFT JOIN HTWOS_STREAMS ON Site_List_Jan03.SiteCode =
HTWOS_STREAMS.SiteCode SET Waste_Jan03.Type = [HTWOS_STREAMS].[Waste_Type]
WHERE (((Waste_Jan03.SiteId)>9917));
```

13. Update the Inventory_Data Table for waste streams sent offsite to Yucca or WIPP (update Attribute2 using the following query):

```
UPDATE (Site_List_Jan03 LEFT JOIN Inventory_Data ON Site_List_Jan03.SiteId
= Inventory_Data.SiteID) LEFT JOIN HTWOS_STREAMS ON
Site_List_Jan03.SiteCode = HTWOS_STREAMS.SiteCode SET
Inventory_Data.Attribute2 = [HTWOS_STREAMS].[New_Attribute]
WHERE (((Site_List_Jan03.SiteId)>9917) AND ((HTWOS_STREAMS.NEW_Attribute)
Is Not Null));
```

14. Update the HTWOS SiteID from Site table using the following query:

```
UPDATE Site_List_Jan03 INNER JOIN HTWOS_LEAK ON Site_List_Jan03.SiteCode =
HTWOS_LEAK.SiteCode SET HTWOS_LEAK.SiteID = [Site_List_Jan03].[SiteID];
```

15. Update HTWOS input (HTWOS_LEAK) Attribute2 field with New_Attribute from HTWOS_Streams using the following query:

```
UPDATE HTWOS_STREAMS INNER JOIN HTWOS_LEAK ON HTWOS_STREAMS.SiteCode =
HTWOS_LEAK.SiteCode SET HTWOS_LEAK.Attribute2 =
HTWOS_STREAMS].[NEW_Attribute]
WHERE (((HTWOS_STREAMS.NEW_Attribute) Is Not Null));
```

16. Update the Site Table Site_Comment to Disposal Site for new HTWOS Sites using the following query:

```
UPDATE HTWOS_STREAMS INNER JOIN Site_List_Jan03 ON HTWOS_STREAMS.SiteCode
= Site_List_Jan03.SiteCode SET Site_List_Jan03.SiteComment =
[HTWOS_STREAMS].[Disposal_Site];
```

17. Update the volume to m3 using the following query:

```
UPDATE HTWOS_LEAK SET HTWOS_LEAK.Constituent_units = "m3"
WHERE (((HTWOS_LEAK.Constituent_name)="volume"));
```

18. Update Units for other constituents to Ci using the following query:

```
UPDATE HTWOS_LEAK SET HTWOS_LEAK.Constituent_units = "Ci"
WHERE (((HTWOS_LEAK.Constituent_units) Is Null));
```

19. Append HTWOS_LEAK records to Inventory_Data table using the following query:

```
INSERT INTO Inventory_Data ( SiteID, SiteCode, Subsite, [Year],
Stat_Distribution, Attribute2, Constituent_units, Constituent_name,
Constituent_value, Source, Dataset )
SELECT HTWOS_LEAK.SiteID, HTWOS_LEAK.SiteCode, HTWOS_LEAK.Subsite,
HTWOS_LEAK.Year, HTWOS_LEAK.Stat_Distribution, HTWOS_LEAK.Attribute2,
HTWOS_LEAK.Constituent_units, HTWOS_LEAK.Constituent_name,
HTWOS_LEAK.Constituent_value, HTWOS_LEAK.Source, HTWOS_LEAK.Dataset
FROM HTWOS_LEAK;
```

20. Perform a select query on the HTWOS_TDP table, grouped to find unique values, with SiteCode not starting with “241”. Check the list of sites generated for accuracy, then change the query to a “Make Table” query to append the data to the site list table.

21.4 Exporting Records from the Inventory Database

Data exported from the database include inventory data and surrogate site data.

21.4.1 Exporting Data to INPROC

INPROC (Section 19) is the code that processes inventory information that is stored in the inventory database. Separate export procedures are required for waste or materials that 1) will be stored or disposed offsite, 2) have been disposed at Hanford, or 3) will be generated as separate waste streams but will be disposed at a common site. Data files exported from these three procedures must be combined into a single text file for use in the INPROC code.

21.4.1.1 Waste or Materials to be shipped Offsite for storage or disposal

TRU waste, High Level Waste, Spent nuclear fuel, and Special Nuclear materials (SNM) are to be shipped offsite for disposal or storage. The site names for wastes and materials to be shipped offsite include the destination (WIPP or Yucca, or SNM for special nuclear materials, plus the original site name. This combined site name is in the "Attribute2" field. (For wastes remaining at Hanford, SiteCode is the second entry in the query, and the text in the Attribute2 field is "none"). The query to list waste or special nuclear material not kept at Hanford is as follows:

```
SELECT Inventory_Data.Source, Inventory_Data.Attribute2,
Inventory_Data.Subsite, Inventory_Data.Year,
Inventory_Data.Stat_Distribution, Inventory_Data.Constituent_name,
Inventory_Data.Constituent_value, Inventory_Data.Dataset,
Inventory_Data.SiteID
FROM Inventory_Data
WHERE (((Inventory_Data.Attribute2)<>"none"))
ORDER BY Inventory_Data.Attribute2, Inventory_Data.Subsite,
Inventory_Data.Year, Inventory_Data.Constituent_name;
```

The results of this query are then exported in a comma-delimited text file format.

21.4.1.2 Waste or materials disposed at Hanford

The set of analytes used in a particular simulation may be a subset of the inventory data. Data only need to be exported for sites which contain at least one analyte of interest (in addition to volume). Data for most Hanford waste are handled in the following three steps:

- Run the query "Filter Constituents" to filter records from table Inventory_Data to exclude nalytes that are not to be considered (for example, chemicals or plutonium).
- Run the query "1-Constituent_Site-Subsite-Year" to determine the records not excluded (above) for which combinations of site, release model, and year have only one constituent (volume). These site-subsite-year combinations are excluded from further consideration.
- Run a third query, "INPROC-Onsite_excluding_combined" to identify records from the Inventory_Data table, excluding those site-subsite-year combinations that are found in the previous query. Results from this query are exported as a csv file, for input to INPROC.

The query "Filter Constituents" removes constituents that are not to be considered in the SAC model run and reads as follows:

```
SELECT Inventory_Data.Source, Inventory_Data.SiteCode,
Inventory_Data.Subsite, Inventory_Data.Year,
Inventory_Data.Constituent_name, Inventory_Data.Stat_Distribution,
Inventory_Data.Dataset, Inventory_Data.SiteID
FROM Inventory_Data
WHERE (((Inventory_Data.Constituent_name) Not Like "NH*" And
(Inventory_Data.Constituent_name) Not Like "NO*" And
(Inventory_Data.Constituent_name) Not Like "Cr*" And
(Inventory_Data.Constituent_name) Not Like "CCl*" And
(Inventory_Data.Constituent_name) Not Like "Pu*" AND
((Inventory_Data.Stat_Distribution)="mean" Or
(Inventory_Data.Stat_Distribution)="mode" Or
(Inventory_Data.Stat_Distribution)="point"))
ORDER BY Inventory_Data.SiteCode, Inventory_Data.Subsite,
Inventory_Data.Year;
```

Results of the “Filter Constituents” query are used in the “1-Constituent_Site-Subsite-Year” query to determine which site-year-subsite combinations contain no constituents of interest (volume only).

```
SELECT Filter_constituents.SiteCode, Filter_constituents.Subsite,
Filter_constituents.Year, Count(Filter_constituents.Year) AS CountOfYear
FROM Filter_constituents
GROUP BY Filter_constituents.SiteCode, Filter_constituents.Subsite,
Filter_constituents.Year
HAVING (((Count(Filter_constituents.Year))=1));
```

The “INPROC-Onsite_excluding_combined” query takes records from Inventory_Data, excluding those site-subsite-year combinations that are found in “1-Constituent_Site-Subsite-Year”. Results from this query are exported as a csv file for input to INPROC. This query applies to wastes that are not from future processing (HTWOS_TDP) because the future processed wastes are either disposed offsite or are separate waste streams that are combined and disposed at a common location. Neither of these cases is suitable for input to the codes that perform the vadose zone and groundwater simulations. A query to build an input file for HTWOS combined wastes is given in the next section. The condition that Attribute2 = “none” is necessary to prevent the waste disposed offsite from appearing in the results. Additional conditions are required for the HTWOS results, to avoid duplicating waste streams that are sent to one of the combined sites (218-E-RCRA, 600-211 or 291-WTP).

```
SELECT Inventory_Data.Source, Inventory_Data.SiteCode,
Inventory_Data.Subsite, Inventory_Data.Year,
Inventory_Data.Stat_Distribution, Inventory_Data.Constituent_name,
Inventory_Data.Constituent_value, Inventory_Data.Dataset,
Inventory_Data.SiteID, Inventory_Data.Attribute2, Inventory_Data.Comment
FROM Inventory_Data LEFT JOIN [1-Constituent_Site-Subsite-Year] ON
(Inventory_Data.Year = [1-Constituent_Site-Subsite-Year].Year) AND
(Inventory_Data.Subsite = [1-Constituent_Site-Subsite-Year].Subsite) AND
(Inventory_Data.SiteCode = [1-Constituent_Site-Subsite-Year].SiteCode)
WHERE (((Inventory_Data.Constituent_name) Not Like "CCl*" And
(Inventory_Data.Constituent_name) Not Like "Cr*" And
(Inventory_Data.Constituent_name) Not Like "NO*" And
(Inventory_Data.Constituent_name) Not Like "NH*" And
(Inventory_Data.Constituent_name) Not Like "Pu*") AND
((Inventory_Data.Dataset) Like "HTWOS*") AND
((Inventory_Data.Attribute2)="none") AND (Not
(Inventory_Data.Comment)="218-E-RCRA" And Not
(Inventory_Data.Comment)="600-211" And Not (Inventory_Data.Comment)="291-
WTP") AND (([1-Constituent_Site-Subsite-Year].Year) Is Null)) OR
(((Inventory_Data.Constituent_name) Not Like "CCl*" And
(Inventory_Data.Constituent_name) Not Like "Cr*" And
(Inventory_Data.Constituent_name) Not Like "NO*" And
(Inventory_Data.Constituent_name) Not Like "NH*" And
(Inventory_Data.Constituent_name) Not Like "Pu*") AND
((Inventory_Data.Dataset) Like "HTWOS*") AND
((Inventory_Data.Attribute2)="none") AND ((Inventory_Data.Comment) Is
Null) AND (([1-Constituent_Site-Subsite-Year].Year) Is Null)) OR
(((Inventory_Data.Constituent_name) Not Like "CCl*" And
(Inventory_Data.Constituent_name) Not Like "Cr*" And
(Inventory_Data.Constituent_name) Not Like "NO*" And
(Inventory_Data.Constituent_name) Not Like "NH*" And
(Inventory_Data.Constituent_name) Not Like "Pu*") AND
((Inventory_Data.Dataset) Not Like "HTWOS*") AND
((Inventory_Data.Attribute2)="none") AND (([1-Constituent_Site-Subsite-
Year].Year) Is Null))
ORDER BY Inventory_Data.SiteCode, Inventory_Data.Subsite,
Inventory_Data.Year, Inventory_Data.Constituent_name;
```

The results of this query are then exported in a comma-delimited text file format.

21.4.1.3 HTWOS waste streams disposed at common Sites

Waste streams (from waste treatment facilities) from HTWOS_TDP are either disposed offsite (WIPP or Yucca) or are sent to a solid waste facility (218-E-RCRA), a liquid waste facility (600-211), or are released to the atmosphere (291-WTP). The waste to be disposed at Hanford must first be combined (by year and waste type) before it is disposed to the combined disposal site. The “HTWOS_Combined” query is as follows:

```
SELECT Inventory_Data.Source, Inventory_Data.Comment,
Inventory_Data.Subsite, Inventory_Data.Year,
Inventory_Data.Stat_Distribution, Inventory_Data.Constituent_name,
Sum(Inventory_Data.Constituent_value) AS SumOfConstituent_value,
Inventory_Data.Dataset
FROM Inventory_Data
GROUP BY Inventory_Data.Source, Inventory_Data.Comment,
Inventory_Data.Subsite, Inventory_Data.Year,
Inventory_Data.Stat_Distribution, Inventory_Data.Constituent_name,
Inventory_Data.Dataset, Inventory_Data.Attribute2
HAVING (((Inventory_Data.Source) Like "HTWOS*") AND
((Inventory_Data.Subsite)="gas" Or (Inventory_Data.Subsite)="glass" Or
(Inventory_Data.Subsite)="liquid" Or (Inventory_Data.Subsite)="cement")
AND ((Inventory_Data.Attribute2)="none"))
ORDER BY Inventory_Data.Comment, Inventory_Data.Subsite,
Inventory_Data.Year, Inventory_Data.Constituent_name;
```

This query must be followed by the “INPROC_Input-HTWOS_Combined” query to creates an INPROC input file from the combined waste streams:

```
SELECT HTWOS_Combined.Source, HTWOS_Combined.Comment,
HTWOS_Combined.Subsite, HTWOS_Combined.Year,
HTWOS_Combined.Stat_Distribution, HTWOS_Combined.Constituent_name,
HTWOS_Combined.SumOfConstituent_value, HTWOS_Combined.Dataset
FROM HTWOS_Combined LEFT JOIN [1-Constituent_Site-Subsite-Year] ON
(HTWOS_Combined.Year = [1-Constituent_Site-Subsite-Year].Year) AND
(HTWOS_Combined.Subsite = [1-Constituent_Site-Subsite-Year].Subsite) AND
(HTWOS_Combined.Comment = [1-Constituent_Site-Subsite-Year].SiteCode)
WHERE (((HTWOS_Combined.Constituent_name) Not Like "CCl*" And
(HTWOS_Combined.Constituent_name) Not Like "Cr*" And
(HTWOS_Combined.Constituent_name) Not Like "NO*" And
(HTWOS_Combined.Constituent_name) Not Like "NH*" And
(HTWOS_Combined.Constituent_name) Not Like "Pu*") AND (([1-
Constituent_Site-Subsite-Year].Year) Is Null))
ORDER BY HTWOS_Combined.Comment, HTWOS_Combined.Subsite,
HTWOS_Combined.Year, HTWOS_Combined.Constituent_name;
```

The results of this query are then exported in a comma-delimited text file format.

21.4.2 Exporting Surrogate Data

The inventory data set does not contain data for every desired site. In some cases, alternate sites with known inventory data can serve as surrogate data sources. Data representing the surrogate sites are stored in the table “Link_Surrogates”. These data are exported and processed into keywords for use in the INPROC code. The data required are: site name of unknown site, volume of unknown site, year for which release occurs to unknown site, site name of the known site, year for basis of known site data, and waste form type (SurrogateSubsite).

22.0 INVPLOT – Inventory Plotting

22.1 Overview

This program generates files of inventory information on an annual time step for combinations of analytes, waste sites, and waste locations. The generated files can be used in a plotting program. The INVPLOT program reads the accumulated, decay-corrected files generated by INGRAB data extractor.

22.1.1 Location in the Processing Sequence

The INVPLOT code reads data files written by the INVENTORY and INGRAB codes, thus cannot be executed until these two codes have completed.

22.1.2 How the Code Is Invoked

INVPLOT can run under either the Windows or the Linux operating system. Under the Windows operating system (Releases 98, NT, 2000, or XP), INVPLOT executes in a DOS box. A run of INVPLOT is initiated by entering the following command line:

```
   INVPLOT “Keyfilename”
```

Under the Linux operating system INVPLOT is executed through any of the following Bourne Shell or C Shell commands:

```
   invplot.exe “Keyfilename”
```

For these commands, INVPLOT.EXE or invplot.exe is the name of the executable program and “Keyfilename” is the name of a control keyword file. Both the name of the executable program and the keyword file may contain path information. If INVPLOT is invoked without entering the name of the keyword file, then the code will prompt the user for the file name. The keyword file, which should be prepared using an editor that can handle ASCII files without leaving embedded control codes, contains text control information describing the run. If INVPLOT cannot open the keyword file, then the code will terminate execution after writing an error message to the standard output device.

22.1.3 Memory Requirements

The INVPLOT code uses dynamic memory allocation, so the memory requirements depend on the problem being analyzed. It is expected that most, if not all, of the runs of the INVPLOT code will require less than 5 MB of memory.

22.2 File Definitions

The INVPLOT program reads three or more input files and writes three or more output files. These files are described in the following sections.

22.2.1 Input Files

The INVPLOT program reads a control keyword file and the “inventory.all” file written by the INVENTORY code. In addition, it reads one or more files written by the INGRAB code. The INVPLOT keyword file contains control information. An example keyword file that extracts data for two cases is provided in Table 22.1. Detailed definitions of the keywords are provided in Section 18.3.

Table 22.1 Example Keyword File for INVPLOT

```
! Keyword file for INVPLOT
REPORT "InvPlot_U238.Rpt"
TITLE "Inventory Plots for SAC. Rev. 1 History 5 Inventory"
USER "Paul W. Eslinger"
! Index file output by INGRAB
FILE INDEX "Inventory.All"
! File for uncertainty data
FILE UNCERT "U238_Uncert.Rpt"
! Type of output
METRIC REALIZATION 1
! Maximum number of curves in any case
!MAXCURVE = 20
! Additional stuff written to the report file
VERBOSE
! Set of curves for all plots
CURVE LABEL="Liquid"    WASTE="liquid"
CURVE LABEL="Glass"     WASTE="glass"
CURVE LABEL="Other"     WASTE="capsul" "store" "TRUSLG"
CURVE LABEL="Reactor"   WASTE="Core" "rxcomp"
CURVE LABEL="Residual"  WASTE="cake"
CURVE LABEL="Air"       WASTE="gas"
CURVE LABEL="Offsite"   WASTE= "HLW"  "NM1" "NM2" "NM4" "NM5" "NM6"
                               "NMd" "NMl" "NMs" "SF" "TRUf" "TRUn"
                               "TRUp" "TRUsf" "WIPP" "offsite"
CURVE LABEL="River"     WASTE="River"
CURVE LABEL="Soil"      WASTE= "soil" "soil1f" "soil1n" "soil3f"
                               "soil3n" "soils" "soilsf" "soilsn"
                               "soilss" "soilst" "soiltu"
CURVE LABEL="Cement"    WASTE= "cement" "cmntcs" "cmntct" "cmntcu"
                               "cmntg3" "cmntH2"
CURVE LABEL="All Types" WASTE="All_Types"
! End of the initial stuff
ENDINIT

CASE
TITLE="U238 Inventory - 300 Site"
ANALYTE "U238"
FILE PLOT "U238_300.csv"
FILE DATA "INGRABA_U238_ALL.CSV"
SITES
 "300_RLWS" "300_RRLWS" "300_VTS" "300-121" "300-123"
  "300-16" "300-2" "300-214" "300-224" "300-24"
  "300-249" "300-25" "300-251" "300-255" "300-262"
  "300-264" "300-265" "300-270" "300-28" "300-33"
  "300-39" "300-4" "300-40" "300-48" "300-80"
  "303-K_CWS" "303-M-SA" "303-M-UOF" "305-B_SF" "307_RB"
  "309-WS-1" "309-WS-2" "313_ESSP" "316-1" "316-2" "316-3"
  "316-5" "325_WTF" "331_LSLDF" "331_LSLT2"
  "333_ESHWSA" "3712_USSA" "600-117" "UPR-300-1"
  "UPR-300-10" "UPR-300-11" "UPR-300-12" "UPR-300-2"
  "UPR-300-32" "UPR-300-34" "UPR-300-36" "UPR-300-37"
  "UPR-300-38" "UPR-300-39" "UPR-300-4" "UPR-300-40"
  "UPR-300-45" "UPR-300-48" "UPR-300-5" "UPR-300-FF-1"
ENDCASE

CASE
TITLE="U238 Inventory - Entire Site"
ANALYTE "U238"
FILE PLOT "All_U238_300.csv"
FILE DATA "N:\CA1_median\inventory\ingrab\IngrabA_U238.csv"
SITES ALL
ENDCASE

END
```

22.2.2 Output Files

The INVPLOT program writes three or more output files. One file is a report file that contains text information about the run of the code. It is not described further here. Another output file is the uncertainty file that contains a summary of all cases performed during the code run. An additional output file is written for every CASE definition. This output file contains the inventory information formatted in comma separated format for ease of input into a plotting program. The file is written in comma separated format. The file starts with nine lines of identification information on file names and the code run. This identification information is followed by a data matrix, one line per year, of the inventory by waste form for each waste form curve.

22.3 Keyword Definitions for the INVPLOT Code

The following restrictions apply on keyword order for the INVPLOT code. The keywords are divided into two sets. The first section, called the control section, is entered once for every run of the code:

- The control section starts with the REPORT keyword and is terminated by an ENDINIT keyword. The other keywords in the control section can be entered in any order.
- The inventory plots are divided into different cases. Each plot definition starts with a CASE keyword and ends with an ENDCASE keyword. All other keywords for a case can be entered in any order.
- The END keyword must be the last keyword in the file.

22.3.1 Control Section Keywords for the INVPLOT Code

22.3.1.1 CURVE Keyword for INVPLOT

The CURVE keyword identifies a set of plot curves that apply to all cases. The following is this keyword's syntax:

```
   CURVE [LABEL="quote 1"] [WASTE= "quote 2" … "quote n"]
```

The plot curve label is entered in the quote string associated with the modifier LABEL. This label is for descriptive purposes and typically is used in the legend of a plot. The waste types to combine for this curve are defined by entering the modifier WASTE followed by a series of waste type IDs. The data for all the waste types will summed before the plot curve is written. The single waste type ID of "All_Types" can be used for a sum across all waste types. Waste type identifiers are case sensitive and must match with waste types contained in the inventory index file.

The following set of keyword entries define 11 curves to be used for all plots:

```
   CURVE LABEL="Liquid"    WASTE="liquid"
   CURVE LABEL="Glass"     WASTE="glass"
   CURVE LABEL="Other"     WASTE="capsul" "store" "TRUSLG"
   CURVE LABEL="Reactor"   WASTE="Core" "rxcomp"
   CURVE LABEL="Residual"  WASTE="cake"
   CURVE LABEL="Air"      WASTE="gas"
   CURVE LABEL="Offsite"  WASTE= "HLW"  "NM1" "NM2" "NM4" "NM5" "NM6" "NMd"
      "NMl" NMs" "SF" "TRUf" "TRUn" "TRUp" "TRUsf" "WIPP" !"offsite"
   CURVE LABEL="River"     WASTE="River"
   CURVE LABEL="Soil"      WASTE= "soil" "soil1f" "soil1n" "soil3f" "soil3n"
      "soils" "soilsf" "soilsn" "soilss" "soilst" "soiltu"
   CURVE LABEL="Cement"    WASTE= "cement" "cmntcs" "cmntct" "cmntcu"
      "cmntg3" "cmntH2"
   CURVE LABEL="All Types" WASTE="All_Types"
```

22.3.1.2 ENDINIT Keyword for INVPLOT

The ENDINIT keyword signifies the end of the control section keyword data. The following is this keyword's syntax:

```
   ENDINIT
```

22.3.1.3 FILE Keyword for INVPLOT

The FILE keyword is used to enter the name of the inventory index file. This file is written by the inventory code and always has the name "inventory.all". The following is this keyword's syntax:

```
   FILE [INDEX="quote1"]
```

The file name is entered in a quote string which must be enclosed in double quotes. Path names up to 200 characters long are supported. The inventory index file name is associated with the modifier INDEX. An example use of this keyword is the following:

```
   FILE INDEX "N:\CA1_median\inventory\inventory.all"
```

22.3.1.4 MAXCURVE Keyword for INVPLOT

The INVPLOT code defaults to allowing 20 inventory curves per case. The optional MAXCURVE keyword can be used to increase the number of curves allowed in each case. The following is this keyword's syntax:

```
MAXCURVE [N1]
```

The single integer entered on this keyword identifies the maximum number of curves to allow for all cases. The following example allows 23 curves to be defined for each case.

```
MAXCURVE 23
```

22.3.1.5 METRIC Keyword for INVPLOT

The METRIC keyword is used to define the output metric for the inventory data. The following is this keyword's syntax:

```
METRIC [ MEAN | REALIZAT=N1 | HORSETAIL ]
```

If the single modifier MEAN is used then the average (mean) of all data realizations is output. If the single modifier REALIZAT is used, followed by an integer, then the data for that specific realization will be output. If the modifier HORSETAIL is used the data for all realizations will be output. The following examples illustrate the use of all definitions for this keyword:

```
METRIC MEAN
METRIC REALIZAT = 3
METRIC HORSETAIL
```

22.3.1.6 REPORT Keyword for INVPLOT

The REPORT keyword is used to define the name of the output report (log) file. It must be the first keyword entered in the keyword file. The following is this keyword's syntax:

```
REPORT [“quote”]
```

The name of the report file is entered in a quote string. File names up to 200 characters long are supported, and path names can be included. The following is an example REPORT keyword record:

```
REPORT “/SAC/C/Test.rpt”
```

22.3.1.7 TITLE Keyword for INVPLOT

The TITLE keyword is used to define a single-line problem title. The problem title will be written to output files. The program will error terminate if the title is not supplied. The following is this keyword's syntax:

```
TITLE [“quote”]
```

The title is entered in a quote string, which must be enclosed in double quotation marks. Titles up to 200 characters long are supported. The following example defines a title for a run of the code:

```
TITLE “Example title line for the INVSUM code.”
```

22.3.1.8 USER Keyword for INVPLOT

The USER keyword is used to identify the user of the program. The user name will be written to output files. The program will error terminate if the user name is not supplied. The following is this keyword’s syntax:

```
USER [“quote”]
```

The user name is entered in a quote string, which must be enclosed in double quotation marks. User names up to 16 characters long are supported. The following example defines John Q. Public as the user running the code:

```
USER “John Q. Public”
```

22.3.1.9 VERBOSE Keyword for INVPLOT

The presence of the optional VERBOSE keyword initiates additional output to the report file. The following is this keyword’s syntax:

```
VERBOSE
```

22.3.2 CASE Section Keywords for the INVPLOT Code

The plots of inventory are divided into different cases. Each plot definition starts with a CASE keyword and ends with an ENDCASE keyword. All other keywords for a case can be entered in any order. The keywords defined for a case are described in the following sections.

22.3.2.1 ANALYTE Keyword for INVPLOT

The ANALYTE keyword is used to define the analyte for which data will be processed. The following is this keyword’s syntax:

```
ANALYTE [“quote 1”]
```

The single quote string on this keyword must an analyte identification string (limit of six characters in length) from the set of the analytes in the inventory results file. An example of this keyword is the following:

```
ANALYTE "H3"
```

22.3.2.2 CASE Keyword for INVPLOT

The CASE keyword is used to signify the start of a plot case definition. The following is this keyword's syntax:

```
CASE
```

22.3.2.3 ENDCASE Keyword for INVPLOT

The ENDCASE keyword is used to signify the end of a plot case definition. The following is this keyword's syntax:

```
ENDCASE
```

22.3.2.4 FILE Keyword for INVPLOT

The FILE keyword is used to enter the name of the INGRAB-produced inventory file and the name of the output file. The following is this keyword's syntax:

```
FILE {PLOT="quote 1"} {DATA="quote 2"}
```

The file names are entered in quote strings which must be enclosed in double quotes. Path names up to 200 characters long are supported. The inventory data file name is associated with the modifier DATA. Typically the data file is an accumulated, decay-corrected output from the INGRAB program. The output data file name is associated with the modifier PLOT. A file name ending in ".csv" is recommended. Both files can be defined using one FILE keyword or they can be entered using separate keywords. An example use of this keyword is the following:

```
FILE PLOT "All_U238_300.csv"
FILE DATA "N:\CA1_median\inventory\ingrab\IngrabA_U238.csv"
```

22.3.2.5 SITES Keyword for INVPLOT

The SITES keyword identifies a set of sites to include in the data sum for a plot. The following is this keyword's syntax:

```
SITES [ ALL | “quote 1” {“quote 2” … “quote n”}]
```

One or more sites are identified using a single SITES keyword. Each site is identified by entering the site ID enclosed in a set of double quotes. All sites can be identified by entering the modifier ALL. The following example identifies 19 sites to include in the data analysis:

```
SITES "300_RLWS" "300_RRLWS" "300_VTS" "300-121"
  "300-123" "300-16" "300-2" "300-214" "300-224"
  "300-24" "300-249" "300-25" "300-251" "300-255"
  "300-262" "300-264" "300-265" "300-270" "300-28"
```

22.3.2.6 TITLE Keyword for INVPLOT

The TITLE keyword is used to define a single-line title used for labeling the output for this case. The following is this keyword's syntax:

```
TITLE [“quote”]
```

The title is entered in a quote string, which must be enclosed in double quotation marks. Titles up to 200 characters long are supported. This title is different for each plot and is different that the title in the control keyword section. The following example defines a title for a particular case:

```
TITLE “U238 – Entire Site”
```

22.3.3 END Keyword for INVPLOT

The END keyword signifies the end of all keyword data. Nominally it will be the last keyword in the keyword file. All data in the keyword file after the END keyword will be ignored. The following is this keyword's syntax:

```
END
```

23.0 INVSUM – Inventory Data Summations

23.1 Overview

The INVSUM (Inventory Data Summation) data extractor allows the user to collect and summarize inventory information. The most commonly used output is a table indicating which sites have a nonzero inventory for each analyte.

23.1.1 Location in the Processing Sequence

The INVSUM code reads data files written by the INVENTORY and INGRAB codes, thus cannot be executed until these two codes have completed.

23.1.2 How the Code Is Invoked

INVSUM can run under either the Windows or the Linux operating system. Under the Windows operating system (Releases 98, NT, 2000, or XP), INVSUM executes in a DOS box. A run of INVSUM is initiated by entering the following command line:

```
INVSUM “Keyfilename”
```

Under the Linux operating system INVSUM is executed through any of the following Bourne Shell or C Shell commands:

```
invsum.exe “Keyfilename”
```

For these commands, INVSUM.EXE or invsum.exe is the name of the executable program and “Keyfilename” is the name of a control keyword file. Both the name of the executable program and the keyword file may contain path information. If INVSUM is invoked without entering the name of the keyword file, then the code will prompt the user for the file name. The keyword file, which should be prepared using an editor that can handle ASCII files without leaving embedded control codes, contains text control information describing the run. If INVSUM cannot open the keyword file, then the code will terminate execution after writing an error message to the standard output device.

23.1.3 Memory Requirements

The INVSUM code uses dynamic memory allocation, so the memory requirements depend on the problem being analyzed. It is expected that most, if not all, of the runs of the INVSUM code will require less than 5 MB of memory.

23.2 File Definitions

The INVSUM code reads three or more input files and writes two output files. The number of input files depends on the number of contaminants being analyzed. These files are described in the following sections.

23.2.1 Input Files

The input files for the INVSUM code are a keyword file, a inventory index file, and one or more inventory data files created by the INGRAB code.

23.2.1.1 INVSUM Keyword File

The INVSUM keyword definition file contains control information for the desired data extraction. Individual keywords are defined in Section 23.3. Table 23.1 contains an example keyword file for the INVSUM code.

Table 23.1 Example Keyword File for INVSUM

```
!  Keyword file for the InvSum program
REPORT "PWE-median.Rpt"
TITLE "Inventory Sums by Site for SAC. Rev. 1 Inventory"
USER "Paul W. Eslinger"
! Index file output by INGRAB
FILE INDEX "O:\PWE\inventory-median\Inventory.All"
FILE SUM   "Prelim5-median.csv"  !Output file
! Analyte specific data
ANALYTE ID="H3"    FILE="O:\PWE\inventory-median\INGRABA_H3_ALL.CSV"
ANALYTE ID="Tc99"  FILE="O:\PWE\inventory-median\INGRABA_Tc99_ALL.CSV"
ANALYTE ID="Sr90"  FILE="O:\PWE\inventory-median\INGRABA_Sr90_ALL.CSV"
ANALYTE ID="U238"  FILE="O:\PWE\inventory-median\INGRABA_U238_ALL.CSV"
ANALYTE ID="I129"  FILE="O:\PWE\inventory-median\INGRABA_I129_ALL.CSV"
! Define the waste stream to be summed
WASTE "All_Types"
! Year for output data
YEAR 2050
! Additional stuff written to the report file
VERBOSE
! End of the keywords
END
```

23.2.1.2 Inventory Index File

The inventory index file used by INGRAB is written by the INVENTORY code. The nominal name of this file is “inventory.all”. The following information is extracted from this file:

- The number of years of inventory data and the suite of calendar years.
- The number of sites included in the analysis and a list of the site IDs.
- The number of analytes in the analysis and a list of the analyte IDs.
- The number of waste types in the analysis and a list of the waste type IDs.

23.2.1.3 Accumulated Inventory Files

A separate accumulated inventory file is required for each analyte in this analysis. These files are all written by the INGRAB code. These files contain inventory values, accumulated to represent the total released for a given site at a given year. The inventory data in these files are decay corrected for radioactive analytes. The accumulated inventory is extracted from these files for the year specified in the INVSUM keyword file.

23.2.2 Output Files

The output files written by the INVSUM code are a report file and a results file.

23.2.2.1 INVSUM Report File

The report file contains summary information from the run of the INVSUM code. This file contains a record of the code version, time and date of run execution, input and output file names, run information, and a summary table of results. This file also contains any error messages generated by the INVSUM code.

23.2.2.2 INVSUM Results File

The INVSUM results file contains the accumulated inventory for every waste site for the specified waste type and year. It contains data for every realization generated by the INVENTORY code. Table 23.2 contains excerpted records from an INVSUM results file for the single analyte tritium (H3) and a single realization. The file is written in comma separated format for ease of importing into a spreadsheet program.

Table 23.2 Excerpted Records from an INVSUM Results File

```
“Program Name:”,”InvSum”
"Program Version:","1.00.A.1"
"Program Date:"," 6 Aug 2003"
"Run ID:","20030925111335"
"Keyword File Name:","PWE-median.key"
"Analyte:","H3"
"Inventory File:","O:\PWE\inventory-median\INGRABA_H3_ALL.CSV"
"Waste Form:","All Types"
"atmosphere", 0.000E+00
"store", 0.000E+00
"216-U-1%2-Fast", 0.000E+00
"100-B-15", 0.000E+00
…
"UPR-200-E-80", 3.087E-06
"UPR-200-E-81", 1.443E-01
…
"UPR-300-FF-1", 0.000E+00
"UPR-600-12", 0.000E+00
"US_Ecology", 3.673E+04
"WRAP", 0.000E+00
"All Sites", 1.957E+05
2050,"year selected for output"
482,"sites had no releases"
497,"sites had releases"
979,"total sites"
```

23.3 Keyword Definitions for INVSUM

In general, the keywords for the INVSUM code can be entered in any order. The following restrictions apply on keyword order:

- The REPORT keyword must be the first keyword in the file.
- The END keyword must be the last keyword in the file.

23.3.1 ANALYTE Keyword for INVSUM

The ANALYTE keyword is used to define the analytes for which inventory data will be extracted. The following is this keyword's syntax:

```
ANALYTE [ID="quote 1"] [FILE="quote 2"]
```

Analytes chosen must be a subset of the analytes used in the fate and transport activities controlled through the ESD keyword file. Multiple analyte keywords may be entered. A separate ANALYTE keyword is required for each desired analyte. The modifiers ID and FILE may be entered in any order. The modifiers associated with the ANALYTE keyword are described in Table 23.3.

Table 23.3 Modifiers Associated with the ANALYTE Keyword in INVSUM

Modifier	Description
ID	The quote string associated with the ID modifier is an analyte ID (up to six characters in length). The analyte ID entered must be one of the set of analytes identified in the environmental settings definition file.
FILE	The quote string associated with the FILE modifier is the name of the data file written by INGRAB for the specified analyte (file name up to 200 characters in length).

The following suite of ANALYTE keywords define three analytes for which inventory information is to be extracted:

```
ANALYTE ID="Tc99"  FILE="D:\SAC\Analyses\Init\Inv\INGRABA_TC_ALL.CSV"
ANALYTE ID="Sr90"  FILE="D:\SAC\Analyses\Init\Inv\INGRABA_SR_ALL.CSV"
ANALYTE ID="U238"  FILE="D:\SAC\Analyses\Init\Inv\INGRABA_U8_ALL.CSV"
```

23.3.2 END Keyword for INVSUM

The END keyword signifies the end of all keyword data. All data in the keyword file after the END keyword will be ignored. The following is this keyword's syntax:

```
END
```

23.3.3 FILE Keyword for INVSUM

The FILE keyword is used to enter the names of some input and output files. Other files names are defined using the ANALYTE and REPORT keywords. The following is this keyword's syntax:

```
   FILE [INDEX=“quote1”] {SUM=“quote2”}
```

The file names are entered in quote strings, which must be enclosed in double quotation marks. Path names up to 200 characters long are supported. The file name associated with a modifier must be entered before the next modifier is entered. At least one FILE keyword is required for every run of the code. Table 23.4 describes the modifiers associated with the FILE keyword.

Table 23.4 Modifiers Associated with the FILE Keyword in INVSUM

Modifier	**Description**
INDEX	The input file identified in the quote string associated with the INDEX modifier contains lists of all of the times, locations, analytes, and waste types that were used in the INVENTORY code. Typically this file is named “inventory.all.”
SUM	The output file identified in the quote string associated with the SUM modifier contains the primary output data for this run of the INVSUM code. A file extension of “.csv” is suggested. The data are comma separated and can be opened directly in a spreadsheet for plotting or other post-processing.

The following two examples show the use of the FILE keyword:

```
   FILE INDEX "O:\History5-stoch\inventory\Inventory.All"
   FILE SUM   "Test_Sum.csv"
```

23.3.4 REPORT Keyword for INVSUM

The REPORT keyword is used to define the name of the output report (log) file. It must be the first keyword entered in the keyword file. The following is this keyword's syntax:

```
   REPORT [“quote”]
```

The name of the report file is entered in a quote string. File names up to 200 characters long are supported, and path names can be included. The following is an example REPORT keyword record:

```
   REPORT “/SAC/C/Test.rpt”
```

23.3.5 TITLE Keyword for INVSUM

The TITLE keyword is used to define a single-line problem title. The problem title will be written to output files. The program will error terminate if the title is not supplied. The following is this keyword's syntax:

```
TITLE [“quote”]
```

The title is entered in a quote string, which must be enclosed in double quotation marks. Titles up to 200 characters long are supported. The following example defines a title for a run of the code:

```
TITLE “Example title line for the INVSUM code.”
```

23.3.6 USER Keyword for INVSUM

The USER keyword is used to identify the user of the program. The user name will be written to output files. The program will error terminate if the user name is not supplied. The following is this keyword's syntax:

```
USER [“quote”]
```

The user name is entered in a quote string, which must be enclosed in double quotation marks. User names up to 16 characters long are supported. The following example defines John Q. Public as the user running the code:

```
USER “John Q. Public”
```

23.3.7 VERBOSE Keyword for INVSUM

The presence of the optional VERBOSE keyword initiates additional output to the screen and the report file. The following is this keyword's syntax:

```
VERBOSE
```

23.3.8 WASTE Keyword for INVSUM

The WASTE keyword is used to identify the single waste stream for which data will be extracted. The program will error terminate if the waste stream ID is not supplied. The following is this keyword's syntax:

```
WASTE [“quote”]
```

The waste stream ID is entered in a quote string, which must be enclosed in double quotation marks. Waste stream ID's up to 24 characters long are supported. The following example defines a run of the code that extracts data for the waste stream with the “cmntcs” ID:

```
WASTE “cmntcs”
```

The typical use of INVSUM is to extract inventory data summed over all waste types. In this case, the ID "All_Types" should be used.

23.3.9 YEAR Keyword for INVSUM

The YEAR keyword is used to identify the single year for which data will be extracted. The program will error terminate if the year is not supplied. The following is this keyword's syntax:

```
YEAR [number]
```

The year is entered as an integer calendar year and should be in the range of years identified in the environmental settings definition keyword file. The following example defines a run of the code that extracts inventory data for the year 2050:

```
YEAR 2050
```

24.0 MAKEU – Inventory Data Uranium Conversion (RES Files)

24.1 Overview

This program reads an inventory results (*.res) file generated by the SAC Rev. 1 inventory code and outputs a revised results file in the same format. The revision is to add a new analyte (uranium as an element) that is computed as a weighted sum of all uranium isotopes.

24.1.1 Location in the Processing Sequence

MAKEU reads a file output by the inventory code. It can be run any time after the inventory code complete.

24.1.2 How the Code Is Invoked

MAKEU can run under either the Windows or the Linux operating system. Under the Windows operating system (Releases 98, NT, 2000, or XP), INGRES executes in a DOS box. A run of INGRES is initiated by entering the following command line:

```
   MAKEU “Keyfilename”
```

Under the Linux operating system MAKEU is executed through any of the following Bourne Shell or C Shell commands:

```
   makeu-1.exe “Keyfilename”
```

For these commands, MAKEU.EXE or makeu-1.exe is the name of the executable program and “Keyfilename” is the name of a control keyword file. Both the name of the executable program and the keyword file may contain path information. If MAKEU is invoked without entering the name of the keyword file, then the code will prompt the user for the file name. The keyword file, which should be prepared using an editor that can handle ASCII files without leaving embedded control codes, contains text control information describing the run. If MAKEU cannot open the keyword file, then the code will terminate execution after writing an error message to the standard output device.

24.1.3 Memory Requirements

The memory used by MAKEU program depends on the number of sites and waste streams. A production run of the code using 1000 waste sites and 40 waste streams is expected to take less that 9 MB of memory.

24.2 File Definitions

The MAKEU program reads two input files and writes two output files. These files are described in the following sections.

24.2.1 Input Files

The MAKEU program reads a control keyword file and a results file written by the INVENTORY code. The MAKEU keyword file contains control information. An example file is provided in Table 24.1. Detailed definitions of the keywords are provided in Section 24.3.

Table 24.1 Example Keyword File for MAKEU

```
REPORT "Test.Rpt"
TITLE "Inventory results - convert Uranium Isotopes to Mass"
USER "Paul W. Eslinger"
FILE RES "inv1.res"
FILE OUTPUT "inv1_U.res"
VERBOSE
OUTPUT ID="U" UNITS="Kg"
SUM ID="U233" FACTOR= 1.0369E-01 UNITS="Ci"
SUM ID="U235" FACTOR= 4.6247E+02 UNITS="Ci"
SUM ID="U236" FACTOR= 1.5448E+01 UNITS="Ci"
SUM ID="U238" FACTOR= 1487.066 UNITS="Ci"
END
```

24.2.2 Output Files

The MAKEU program writes two output files. One file is a report file that contains text information about the run of the code. It is not described further here, however, it should always be examined after a run of the code because error messages are written to the report file. The other output file is a modified results (*.res) file. This file has the uranium mass (kg) entered at the end of the data for every waste site that contains one or more uranium isotopes. Data for one liquid waste stream for the site 100-B-5 is shown in Table 24.2.

Table 24.2 Excerpts from an MAKEU–generated results file

```
"100-B-5",1,"Onsite location, number of years of data that follow"
1954,1,12,"m^3","liquid", 2.41490E+04
"C14","Ci", 1.77331E-10
"Cs137","Ci", 1.56896E-01
"I129","Ci", 4.23598E-08
"Np237","Ci", 3.84186E-07
"Se79","Ci", 6.69845E-07
"Sr90","Ci", 1.30533E-01
"Tc99","Ci", 2.06887E-05
"H3","Ci", 4.46974E-04
"Eu152","Ci", 2.95608E-06
"U235","Ci", 1.28101E-06
"U238","Ci", 2.95680E-05
"U","Kg", 4.45620E-02
```

24.3 Keyword Definitions for MAKEU

In general, the keywords for the MAKEU code can be entered in any order. The following restrictions apply on keyword order:

- The REPORT keyword must be the first keyword in the file.
- The END keyword must be the last keyword in the file.

24.3.1 END Keyword for MAKEU

The END keyword signifies the end of all keyword data. All data in the keyword file after the END keyword will be ignored. The following is this keyword's syntax:

```
END
```

24.3.2 FILE Keyword for MAKEU

The FILE keyword is used to enter the names of all input and output files except for the report file. The following is this keyword's syntax:

```
FILE [modifier1="quote1"] {modifier2="quote2"}
```

The file names are entered in quote strings, which must be enclosed in double quotes. Path names up to 200 characters long are supported. The file name associated with a modifier must be entered before the next modifier is entered. At least one FILE keyword is required for every run of the code.

The name of the results (*.res) file written by the inventory code is associated with the modifier RES. The name of the output file from MAKEU is associated with the modifier OUTPUT. Example file keywords that define these two files are the following:

```
FILE RES "/home/ANALYSIS4/CA1_median/inventory/inv01.res"
FILE OUTPUT "/home/ANALYSIS4/CA1_median/inventory/inv01_U.res"
```

24.3.3 OUTPUT Keyword for MAKEU

The OUTPUT keyword is used to define the output analyte ID and the data units for the output analyte. The following is this keyword's syntax:

```
OUTPUT [ID “quote”] [UNITS=“quote2”]
```

The quote string associated with the ID modifier contains the ID for the output analyte. The quote string associated with the UNITS modifier identifies the units for the output analyte. Nominally, as illustrated in the following keyword, the output analyte ID is “U” and the data units are “Kg.”

```
OUTPUT ID="U" UNITS="Kg"
```

The SAC codes implicitly assume that all mass units for hazardous chemicals are kilograms. Entering an output units string that is not “Kg” may lead to later computational difficulties.

24.3.4 REPORT Keyword for MAKEU

The REPORT keyword is used to define the name of the output report (log) file. It must be the first keyword entered in the keyword file. The following is this keyword’s syntax:

```
REPORT [“quote”]
```

The name of the report file is entered in a quote string. File names up to 200 characters long are supported, and path names can be included. The following is an example REPORT keyword record:

```
REPORT “/SAC/C/Test.rpt”
```

24.3.5 SUM Keyword for MAKEU

The SUM keyword is used to define the suite of analytes that will be converted to mass units and summed. The following is this keyword’s syntax:

```
SUM [ID “quote”] [FACTOR=N1] [UNITS=“quote2”]
```

The quote string associated with the ID modifier contains the ID for the input analyte. The numerical value associated with the FACTOR modifier contains the multiplicative conversion factor (units are Kg/Ci). The conversion factor is the multiplier that converts Curies of the isotope to kilograms of the element. The quote string associated with the UNITS modifier identifies the data units for the input data. The SAC modeling system makes the implicit assumption that radioactive isotopes have units of Ci. Entering any units other than Ci may cause processing difficulties in the modeling system.

Multiple analytes can be used in the sum. Each analyte in the sum must be defined with a separate SUM keyword. A set of five keywords that will compute uranium mass from five uranium isotopes are the following:

```
SUM ID="U233" FACTOR= 1.0369E-01 UNITS="Ci"
SUM ID="U234" FACTOR= 1.6072E-01 UNITS="Ci"
SUM ID="U235" FACTOR= 4.6247E+02 UNITS="Ci"
SUM ID="U236" FACTOR= 1.5448E+01 UNITS="Ci"
SUM ID="U238" FACTOR= 2.9740E+03 UNITS="Ci"
```

24.3.6 TITLE Keyword for MAKEU

The TITLE keyword is used to define a single-line problem title. The problem title will be written to output files. The program will error terminate if the title is not supplied. The following is this keyword’s syntax:

```
TITLE [“quote”]
```

The title is entered in a quote string, which must be enclosed in double quotation marks. Titles up to 200 characters long are supported. The following example defines a title for a run of the code:

```
TITLE “Example title line for the MAKEU code.”
```

24.3.7 USER Keyword for MAKEU

The USER keyword is used to identify the user of the program. The user name will be written to output files. The program will error terminate if the user name is not supplied. The following is this keyword’s syntax:

```
USER [“quote”]
```

The user name is entered in a quote string, which must be enclosed in double quotation marks. User names up to 16 characters long are supported. The following example defines John Q. Public as the user running the code:

```
USER “John Q. Public”
```

24.3.8 VERBOSE Keyword for MAKEU

The presence of the optional VERBOSE keyword initiates additional output to the report file. The following is this keyword’s syntax:

```
VERBOSE
```

25.0 RLGRAB – Extractions from the Release Module

25.1 Overview

RLGRAB is a tool for retrieving release and inventory time-profile data from the VADER runs as written to the vader.table files. RLGRAB can also retrieve release model parameters used in the SAC runs. RLGRAB retrieves data for one analyte per run.

25.1.1 Location in the Processing Sequence

RLGRAB retrieves data from vader.table files built by the Release Module (VADER) code. RLGRAB can be run any time after all the VADER runs in a SAC simulation are completed.

25.1.2 How the Code Is Invoked

RLGRAB only runs under the Linux operating system. RLGRAB is executed through any of the following Bourne Shell or C Shell commands:

```
   rlgrab.exe “RLGRABKeyFileName”
```

For these commands, rlgrab.exe is the name of the executable program and “RLGRABKeyFileName” is the name of the RLGRAB control keyword file. If the RLGRAB control keyword file name is not supplied, it defaults to a local file named “rlgrab.key”. The executable program name and the keyword file name can include path information.

25.1.3 Memory Requirements

The RLGRAB code uses dynamic memory allocation. It is expected that all of the runs of the RLGRAB code will require less than 24 MB of memory.

25.2 File Definitions

The RLGRAB code reads one or more input files and writes one or more output files.

25.2.1 Input Files

RLGRAB reads a control keyword file, an optional site name list, and one or more VADER table files. An example keyword file is provided in Table 25.1. The format of the table files is discussed in Eslinger et al. (2004a), Section 5.5.2 and is not discussed further here.

Table 25.1 Example Keyword File for RLGRAB

```
TITLE "Test Decaying and Filling In"
SITE   "241-A-101"  "241-A-102"
ANALYTE "Sr90"  HALFLIFE=28.78 DECAYYR=1944
WASTETYPE  "CMNT" "SOIL"
YEARS START=1944 END=22050
VARIABLE "RMD_IN" "RMD_OUT" "QTY_IN"
PATHNAME "/home/ANALYSIS4/CA1_median/vadose/"
REALIZATION 1  MAXREAL=1
END
```

RLGRAB attempts to open and read all vader.table files in the subdirectories below the "vadose" subdirectory which match the analyte, site names, and realizations provided in the rlgrab.key file. If a vader.table file cannot be opened or read, RLGRAB notes that occurrence in the RLGRAB.Con file and proceeds. Therefore, the user must verify that all vader.table files have been successfully processed before proceeding to post-retrieval analysis. To verify that all files were actually processed, inspect the "Number failed to open" line near the bottom of the RLGRAB.Rpt file.

25.2.2 Output Files

RLGRAB writes one or two output files. The report file is written for every run of the code and has a default name of RLGRAB.Rpt. This text file documents the processing actions and contains any error messages. The other output file has a default name of RLGRAB.Con and contains yearly consolidated release quantities. The OUTFILE keyword can be used to provide alternative file names for these two files. The RLGRAB.Con file is a text file containing a few header lines and a data matrix. The data matrix is often imported into a plotting package, statistical package, or spreadsheet for further processing and display.

25.3 Keyword Definitions for RLGRAB

In general, the keywords for the RLGRAB code can be entered in any order. The following restrictions apply on keyword order:

- Only one of the PERIOD and YEARS keywords should be used.
- The END keyword must be the last keyword in the file.

25.3.1 ANALYTE Keyword for RLGRAB

The ANALYTE keyword is used to define the analytes for which release data will be extracted. The following is this keyword's syntax:

```
ANALYTE [“quote 1”] {HALFLIFE=n1} {DECAYYR=N2}
```

Analytes chosen must be a subset of the analytes used in the fate and transport activities controlled through the ESD keyword file. Only a single analyte keyword may be entered. The optional numerical data value associated with the optional modifier HALFLIFE defines the halflife (years) of the analyte. A value of 1.0E+34 is used if this modifier is not used. The optional numerical value associated with the optional DECAYYR is a calendar year to which all radioactive data will be decay corrected. If this value is not defined the data are not decay corrected (and thus represent the year of disposal). An example keyword is the following:

```
ANALYTE "Sr90"  HALFLIFE=28.78 DECAYYR=1944
```

25.3.2 END Keyword for RLGRAB

The END keyword signifies the end of all keyword data. All data in the keyword file after the END keyword will be ignored. The following is this keyword's syntax:

```
END
```

25.3.3 OUTFILE Keyword for RLGRAB

The optional OUTFILE keyword is used to modify the default output file names. The following is this keyword's syntax:

```
OUTFILE [RAW="quote 1"]
```

The quote string associated with the RAW modifier contains the path name and base file name for output. The report file will have the location and name derived from appending ".Rpt" to the quote string. The output results file will have the location and name derived from appending ".Con" to the quote string. An example use of this keyword is the following:

```
OUTFILE "/home/ANALYSIS4/CA1_median/Case2"
```

This keyword would result in naming the two output files as follows:

```
"/home/ANALYSIS4/CA1_median/Case2.Rpt"
"/home/ANALYSIS4/CA1_median/Case2.Con"
```

25.3.4 PATHNAME Keyword for RLGRAB

The optional PATHNAME keyword is used to identify path name to the vadose directory of the analysis being examined. The following is this keyword's syntax:

```
PATHNAME ["quote"]
```

The quote string containing the pathname must end with the "/" character. The path associated with the RLGRAB.key file will be used if this keyword is not entered. An example use of this keyword is the following:

```
PATHNAME "/home/ANALYSIS4/CA1_median/vadose/"
```

25.3.5 PERIOD Keyword for RLGRAB

The PERIOD keyword identifies the start and stop years for the analysis being examined. The following is this keyword's syntax:

```
PERIOD [START=N1] {END=N2} [STOP=N3]
```

The PERIOD and YEARS keywords identify the same data. Only one of these two keywords should be used in a single run of the code. The modifier START and the associated value N1 identify the year to start extracting data. The modifier STOP and the associated value N3 identify the year to stop extracting data. The optional modifier END and the associated value N2 also identify the year to stop extracting data. The STOP modifier takes precedence if both the END and STOP modifiers are used. Start and stop years should be entered as whole numbers with the stop year no smaller than the start year. The following is an example PERIOD keyword that extracts data from 1944 through 3050:

```
PERIOD START=1944 STOP=3050
```

25.3.6 REALIZATION Keyword for RLGRAB

The REALIZAT keyword identifies the realization numbers to process and the total number of realizations that were simulated. The following is this keyword's syntax:

```
REALIZATION {MAXREAL=N1} [ N2-N3 | N4 {N5} … {N6} | ALL]
```

The total number of realizations processed in the analysis is identified by the modifier MAXREAL. A default value of 100 realizations will be used if the modifier MAXREAL is not present. The specific realizations to process can be specified on three different ways. First, a range of realizations (small to large) can be entered by providing two numbers separated by a hyphen. Second, a list of specific realizations can be specified as individual realization numbers. Finally, all realizations can be specified by entering the modifier ALL. The following example REALIZAT keywords all request realizations 1 through 6 of a data set containing exactly six realizations.

```
REALIZAT ALL MAXREAL=6
REALIZAT 1 2 3 6 5 4 MAXREAL=6
REALIZAT 1-6 MAXREAL=6
```

25.3.7 SITE Keyword for RLGRAB

The optional SITE keyword is used to identify a list of site ID's to process. The following is this keyword's syntax:

```
SITE [“quote1] (“quote2”} … {“quoteN”)
```

The quote strings entered on this keyword must each be a valid site ID for the analysis being examined. More than one SITE keyword can be entered. The effect of any SITE keywords and the SITEFILE keyword is cumulative. An example use of this keyword is the following:

```
SITE  “200_ETF” “200-E-100” “200-E-102” “200-E-103”
```

25.3.8 SITEFILE Keyword for RLGRAB

The optional SITEFILE keyword is used to identify an input file that contains a list of site ID’s to process. The following is this keyword’s syntax:

```
SITEFILE [“quote”]
```

The effect of any SITE keywords and the SITEFILE keyword is cumulative. An example use of this keyword is the following:

```
SITEFILE “SiteID_Case10.dat”
```

25.3.9 TITLE Keyword for RLGRAB

The optional TITLE keyword is used to define a single-line problem title that will be written to output files for labeling purposes. The following is this keyword’s syntax:

```
TITLE [“quote”]
```

An example use of this keyword is the following:

```
TITLE “SAC Special Run – Retrieval 17”
```

25.3.10 VARIABLE Keyword for RLGRAB

The VARIABLE keyword is used to identify the type of output to process. The following is this keyword’s syntax:

```
VARIABLE [“quote1”] (“quote2”} … {“quoteN”)
```

This keyword specifies the variables in the VADER.table files to extract and sum. Entries in the quote strings are not case sensitive for this keyword. More than one VARIABLE keyword can be entered. The effect of multiple VARIABLE keywords is cumulative. The user can retrieve any variable in the VADER.table file heading by name. Example entries for this keyword are the following:

```
VARIABLE “Qty_In”   ! Extract the annual inventory deposit
VARIABLE “RMD_IN”   ! Extract the annual remediation import
VARIABLE “RMD_OUT”  ! Extract the annual remediation export
VARIABLE “Release”  ! Extract the annual quantity released
VARIABLE “CumQty”   ! Extract the net inventory remaining
VARIABLE “CumRel”   ! Extract the cumulative quantity released
VARIABLE “Qw(cm)”   ! Extract the annual infiltration rate
```

Normally RLGRAB should be used to retrieve only one time-dependent variable per run, since it makes no sense for most of the variables to be added together. However, retrievals that combine inventory deposits and remediation transfers, in order to gain a picture of total inventory at a site not accounting for release (decayed to deposit or transfer year) can be performed. This is accomplished by setting the VARIABLE keyword arguments to more than one variable, as in:

```
   VARIABLE “QTY_IN” “RMD_IN” “RMD_OUT” ! Retrieve all deposits and transfers
   VARIABLE “RMD_IN” “RMD_OUT”          ! Focus on remedial actions
   VARIABLE “QTY_IN” “RMD_IN”           ! Focus on deposits and imports
   VARIABLE “QTY_IN” “RMD_OUT”          ! Focus on deposits and exports
```

25.3.11 WASTETYP Keyword for RLGRAB

The WASTETYP keyword is used to identify a list of waste types to process. The following is this keyword’s syntax:

```
   WASTETYP [“quote1] (“quote2”} … {“quoteN”)
```

This keyword specifies the waste types in the VADER.table files to extract and sum. Entries in the quote strings are not case sensitive for this keyword. The text in the quote string defining a waste type needs to actually appear in the VADER.table file “Waste” heading group in order for any data to be extracted. Matching is declared whenever the quote string on this keywords contains the specified waste type as an embedded string (for example, “soilsn” is retrieved as “SOIL”). More than one WASTETYP keyword can be entered. The effect of multiple WASTETYP keywords is cumulative.

To obtain results organized by waste type it is necessary to perform a separate retrieval for each waste type. One approach is to an initial RLGRAB retrieval to build a RLGRAB.Rpt file. The report file contains information on all waste forms output to the vader.table file for all sites, whether or not they were actually added to the totals in the RLGRAB.Con file. These appear as the “Specs” records in the report file.

Specify the string “AGGREGATE” for all releases from solid waste types in the vadose zone. If “AGGREGATE” is specified, do not specify any other solid waste types. Specify the string “LIQUID” for all releases from liquid waste streams to the vadose zone. A run extracting all liquid releases to both the vadose zone and the river should specify “LIQUID” and “RIVER”. An example use of this keyword to extract releases from soil waste types is the following:

```
   WASTETYPE "SOIL"
```

This example keyword will extract and sum data for the following waste types: “soil”, “soil1f”, “soil1n”, “soil3f”, “soil3n”, “soils”, “soilsf”, “soilsn”, “soilss”, “soilst”, and “soiltu”.

25.3.12 YEARS Keyword for RLGRAB

TheYEARS keyword identifies the start and stop years for the analysis being examined. The following is this keyword's syntax:

```
   YEARS [START=N1] [STOP=N2]
```

The PERIOD and YEARS keywords identify the same data. Only one of these two keywords should be used in a single run of the code. The modifier START and the associated value N1 identify the year to start extracting data. The modifier STOP and the associated value N2 identify the year to stop extracting data. Start and stop years should be entered as whole numbers with the stop year no smaller than the start year. The following is an example YEARS keyword that extracts data from 1944 through 3050:

```
   YEARS START=1944 STOP=3050
```

26.0 SIMS – SIM Inventory Data Processing

26.1 Overview

The SIM (Soil Inventory Model) information is received in large Excel files that have multiple worksheets. These worksheets contain stochastic volume and release amounts for many liquid release sites at Hanford. The SIMS utility code is used to reformat and decay correct the data so they can be imported into the SAC inventory database. SIM data, as received, are decay corrected to a single common year. The SIMS processor back-decays data, as necessary, to correct activity amounts (Ci) for radioactive analytes to the year of disposal.

26.1.1 Location in the Processing Sequence

The SIMS utility code is a preprocessor to the inventory database. It is used before any other SAC code.

26.1.2 Memory Requirements

The SIMS code uses dynamic memory allocation. It is expected that all of the runs of the SIMS code will require less than 2 MB of memory.

26.1.3 How the Code Is Invoked

SIMS runs under the Windows operating system in a DOS box. A run of SIMS is initiated by entering the following command line:

```
SIMS FilePrefix
```

For this command, SIMS.EXE is the name of the executable program and "FilePrefix" is the prefix for an input file name. The SIMS code attempts to open an input file named FilePrefix.csv. If SIMS cannot find the specified file, then the code will terminate execution after writing an error message to the standard output device.

26.2 File Definitions

The SIMS code reads a two input files and writes nine output files.

26.2.1 Input Files

The first input file for the SIMS code contains the data for a single release area (see the processing description in Section 26.3). This file name always ends in ".csv". This data file has the following structure:

Line 1: This line contains heading information including the SIM data set name, the processing date and time, and labels for volume and concentration distributions. This line is skipped during processing.

Line 2: This line contains column headings including the site, year, analyte, units, and statistical labels. This line is also skipped during processing.

Lines 3 on: Each following line contains 78 data values. These values are a data index, an analyte index, the site ID, the year of processing, the analyte ID, the data units, and statistical quantities for three data sets (total quantity, total volume, and concentration). The total quantity information is skipped and the volume and concentration distributions are used instead.

The first 4 lines from a SIMS input data file are shown in Table 26.1. The lines are wrapped in the table so a blank line is used to indicate the beginning of the next data line. A production data set may contain 10,000 or more lines of data.

The second input file for SIMS is a library of radioactive chain decay information that is always named RMDLIB.DAT. The format of this file is not described further because the user typically does not modify this file.

26.2.2 Output Files

The SIMS code writes a text log file for every run of the code. This file is named "SIMS.OUT". The code also writes a text error message file named "L.ERR". These files should always be examined to determine whether an unexpected errors were encountered during the run of the code.

The SIMS code writes a primary output file and a number of secondary files that are not used further in the data processing. If the input file is named CASE.csv then the primary output file is named CASE.out. Thus, the input "*.csv" file and the output file share the same file name prefix. The primary output file is then imported into the inventory database.

Excerpts from the primary output file from SIMS is provided in Table 26.2. The first line shows as wrapped to the second line in this example. This line contains variable names for use in the inventory database. These data represent the volume and quantities of each analyte in each waste stream as a statistical distribution made up of 23 points defined as a cumulative distribution function.

A number of additional files are written by the SIMS code. The data file named "STREAMS.OUT" should be ignored. If the input file is named CASE.csv, then the following additional files are written:

- CASE.qaf – A file that contains data for quality assurance checks on mean values
- CASE.qaM – A file that contains data for quality assurance checks on median values
- CASE.005 – A file that contains data for quality assurance checks on 0.5% values
- CASE.05 – A file that contains data for quality assurance checks on 5% values
- CASE.95 – A file that contains data for quality assurance checks on 95% values
- CASE.995 – A file that contains data for quality assurance checks on 99.5 values
-

These files should also be ignored.

Table 26.1 Excerpted Lines from a SIMS Input Data File

```
sim_7_Jun_04,,,,,,5/28/2004 8:17:47 PM,,,,,,,,,,,,,,,,,,,,,,,,Total Volume Distribution,,,,,,,,,,,,,,,,,,,,,,,,
Concentration Distribution,,,,,,,,,,,,,,,,,,,,,,

s,a,site,year,analyte,units,Mean,StdDev,Median,0.5%,5.0%,10.0%,15.0%,20.0%,25.0%,30.0%,35.0%,40.0%,
45.0%,50.0%,55.0%,60.0%,65.0%,70.0%,75.0%,80.0%,85.0%,90.0%,95.0%,99.5%,,Mean,StdDev,0.5%,5.0%,
10.0%,15.0%,20.0%,25.0%,30.0%,35.0%,40.0%,45.0%,50.0%,55.0%,60.0%,65.0%,70.0%,75.0%,80.0%,85.0%,
90.0%,95.0%,99.5%,,Mean,StdDev,0.5%,5.0%,10.0%,15.0%,20.0%,25.0%,30.0%,35.0%,40.0%,45.0%,50.0%,
55.0%,60.0%,65.0%,70.0%,75.0%,80.0%,85.0%,90.0%,95.0%,99.5%

55,1,216-A-1,1955,Na,kg,1660.72343,555.214747,1574.655405,689.7818562,927.8632123, 1039.256268,
1121.665843,1196.567701,1267.544231,1330.332286,1387.830612,1449.155344,1510.725258,1574.655405,
1640.881031,1708.794834,1785.184304,1865.167991,1957.382479,2070.635388,2208.22735,2391.58399,
2689.68635,3663.329121,,0.098400047,0.004017238,0.089542637,0.091670689,0.092960444,
0.093948977,0.094782662,0.095517653,0.096182095,0.096793069,0.097361283,0.097895271,0.09839999,
0.098904961,0.099437765,0.10000882,0.100618056,0.101282104,0.102017587,0.102851514,0.103839671,
0.105133454,0.107264129,,16878.60694,5595.852738,6982.181335,9422.820029,10617.28797,11448.66166,
12195.06805,12886.20981,13506.2747,14149.62904,14746.05075,15366.07745,16014.47359,16677.43812,
17380.96961,18132.67244,18970.74674,19911.01066,21024.71886,22399.11583,24242.25285,27281.37256,
36943.52395

55,2,216-A-1,1955,Al,kg,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,,0.098400047,0.004017238,
 0.089542637,0.091670689,0.092960444,0.093948977,0.094782662,0.095517653,0.096182095,
0.096793069,0.097361283,0.097895271,0.09839999,0.098904961,0.099437765,0.10000882,
0.100618056,0.101282104,0.102017587,0.102851514,0.103839671,0.105133454,0.107264129,,
0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
```

Table 26.2 Excerpts from an Output File from SIMS

```
Dataset,SiteCode, Attribute2, Year, Stat_distribution, Constituent_name,
Constituent_value
sim_7_Jun_04, 216-A-1, liquid,1955, Mean  , volume,  9.8400E+01
sim_7_Jun_04, 216-A-1, liquid,1955, SD    , volume,  4.0172E+00
sim_7_Jun_04, 216-A-1, liquid,1955, V0005 , volume,  8.9543E+01
sim_7_Jun_04, 216-A-1, liquid,1955, V0050 , volume,  9.1671E+01
sim_7_Jun_04, 216-A-1, liquid,1955, V0100 , volume,  9.2960E+01
sim_7_Jun_04, 216-A-1, liquid,1955, V0150 , volume,  9.3949E+01
```

```
sim_7_Jun_04, 216-A-1, liquid,1955, V0200 , volume,  9.4783E+01
sim_7_Jun_04, 216-A-1, liquid,1955, V0250 , volume,  9.5518E+01
sim_7_Jun_04, 216-A-1, liquid,1955, V0300 , volume,  9.6182E+01
sim_7_Jun_04, 216-A-1, liquid,1955, V0350 , volume,  9.6793E+01
sim_7_Jun_04, 216-A-1, liquid,1955, V0400 , volume,  9.7361E+01
sim_7_Jun_04, 216-A-1, liquid,1955, V0450 , volume,  9.7895E+01
sim_7_Jun_04, 216-A-1, liquid,1955, V0500 , volume,  9.8400E+01
sim_7_Jun_04, 216-A-1, liquid,1955, V0550 , volume,  9.8905E+01
sim_7_Jun_04, 216-A-1, liquid,1955, V0600 , volume,  9.9438E+01
sim_7_Jun_04, 216-A-1, liquid,1955, V0650 , volume,  1.0001E+02
sim_7_Jun_04, 216-A-1, liquid,1955, V0700 , volume,  1.0062E+02
sim_7_Jun_04, 216-A-1, liquid,1955, V0750 , volume,  1.0128E+02
sim_7_Jun_04, 216-A-1, liquid,1955, V0800 , volume,  1.0202E+02
sim_7_Jun_04, 216-A-1, liquid,1955, V0850 , volume,  1.0285E+02
sim_7_Jun_04, 216-A-1, liquid,1955, V0900 , volume,  1.0384E+02
sim_7_Jun_04, 216-A-1, liquid,1955, V0950 , volume,  1.0513E+02
sim_7_Jun_04, 216-A-1, liquid,1955, V0995 , volume,  1.0726E+02
sim_7_Jun_04, 216-A-1, liquid,1955, Mean  , U234,  4.2170E-04
sim_7_Jun_04, 216-A-1, liquid,1955, SD    , U234,  3.7018E-04
sim_7_Jun_04, 216-A-1, liquid,1955, V0005 , U234,  2.5605E-05
sim_7_Jun_04, 216-A-1, liquid,1955, V0050 , U234,  6.8501E-05
sim_7_Jun_04, 216-A-1, liquid,1955, V0100 , U234,  9.7731E-05
sim_7_Jun_04, 216-A-1, liquid,1955, V0150 , U234,  1.2356E-04
sim_7_Jun_04, 216-A-1, liquid,1955, V0200 , U234,  1.4664E-04
sim_7_Jun_04, 216-A-1, liquid,1955, V0250 , U234,  1.7029E-04
sim_7_Jun_04, 216-A-1, liquid,1955, V0300 , U234,  1.9533E-04
sim_7_Jun_04, 216-A-1, liquid,1955, V0350 , U234,  2.2036E-04
sim_7_Jun_04, 216-A-1, liquid,1955, V0400 , U234,  2.4744E-04
sim_7_Jun_04, 216-A-1, liquid,1955, V0450 , U234,  2.7727E-04
sim_7_Jun_04, 216-A-1, liquid,1955, V0500 , U234,  3.1079E-04
sim_7_Jun_04, 216-A-1, liquid,1955, V0550 , U234,  3.4545E-04
sim_7_Jun_04, 216-A-1, liquid,1955, V0600 , U234,  3.8541E-04
sim_7_Jun_04, 216-A-1, liquid,1955, V0650 , U234,  4.3020E-04
sim_7_Jun_04, 216-A-1, liquid,1955, V0700 , U234,  4.8320E-04
sim_7_Jun_04, 216-A-1, liquid,1955, V0750 , U234,  5.4677E-04
sim_7_Jun_04, 216-A-1, liquid,1955, V0800 , U234,  6.2969E-04
sim_7_Jun_04, 216-A-1, liquid,1955, V0850 , U234,  7.3950E-04
sim_7_Jun_04, 216-A-1, liquid,1955, V0900 , U234,  8.9096E-04
sim_7_Jun_04, 216-A-1, liquid,1955, V0950 , U234,  1.1597E-03
sim_7_Jun_04, 216-A-1, liquid,1955, V0995 , U234,  2.1192E-03
sim_7_Jun_04, 216-A-1, liquid,1955, Mean  , U235,  1.8645E-05
```

26.3 Procedure for Processing SIM files for Input to Inventory Database

The SIM data are received in large Excel files that have multiple sheets. Each file has a sheet titled “Legend” and additional sheets with release data for specific areas. Each of these sheets are saved to a comma separated variable (.csv) file and processed with the SIMS program. The following steps are required in preparing the “.csv” files:

1. Open a SIM Excel file.
2. Open a sheet in the file for a specific waste area.
3. In cell A1, change the contents (initially Total Inventory”) to the set name to be assigned to this SIM case. Typically the set name is SUM_date (SIM_07_Jun_2004 for example).
4. Save the sheet as a “.csv” file with the area name and extension “.csv”. For example, the Excel sheet named “Solid Waste Zone” is given the file name of “SOLIDWASTE.csv”.
5. Repeat this process until data for all waste areas have been save to disk as “.csv” files.

The next step is to run the SIMS program to convert each of the “.csv” files into data files that can be imported in the inventory database. If there were four input files named BPLANT.CSV, BFARM.CSV, 200WPONDS.CSV, and 200EPONDS.CSV, then the following four entries would be made at a DOS prompt (SIMS only runs under the Windows operating system):

```
SIMS BPLANT
SIMS BFARM
SIMS 200WPONDS
SIMS 200EPONDS
```

This sequence of four runs results in the four output files named BPLANT.OUT, BFARM.OUT, 200WPONDS.OUT, and 200EPONDS.OUT. The “.OUT” files are then input to the Inventory Database. The database import would be easier if the extension on each of these files were “.csv” rather than “.out”. Copy the “.out” files to another directory before changing the extension to “.csv” so the input data for SIMS are not destroyed.

27.0 STOCHASTIC – Stochastic Values Generation

27.1 Overview

This utility is a stand-alone code that generates values defined by statistical distributions. Output options include individual generated values and summary statistics. It can be used to check the effects of stochastic variables before they are included in a SAC simulation.

27.1.1 Location in the Processing Sequence

The STOCHASTIC code can be executed independent of other codes in the SAC processing sequence. However, is can also be coupled with other utility codes (such as Inv_Med, Sto_Med, and Imp_Med) to generate keyword files populated with the median values for each of the stochastic variables.

27.1.2 How the Code Is Invoked

STOCHASTIC can run under either the Windows or the Linux operating systems. Under the Windows operating system (2000, or XP), STOCHASTIC executes in a DOS box. A run of STOCHASTIC is initiated by entering the following command line:

```
STOCHASTIC “Keyfilename”
```

Under the Linux operating system CSCALE is executed through any of the following Bourne Shell or C Shell commands:

```
stochastic.exe “Keyfilename”
```

For these commands, STOCHASTIC.EXE or stochastic.exe is the name of the executable program and “Keyfilename” is the name of an existing control keyword file. Both the name of the executable program and the keyword file may contain path information. If STOCHASTIC is invoked without entering the name of the keyword file, then the code will prompt the user for the file name. If STOCHASTIC cannot find or open the keyword file, then the code will terminate execution after writing an error message to the standard output device.

27.1.3 Memory Requirements

The STOCHASTIC code uses dynamic memory allocation, so the memory requirements depend on the problem being analyzed. A reasonably large example run where the STOCHASTIC code required 16 MB of memory (on a Windows XP machine) used six stochastic variables and one million realizations.

27.2 File Definitions

The STOCHASTIC code reads a single input file and writes up to three output files.

27.2.1 Input Files

The single input file for the STOCHASTIC code is a control keyword file. An example keyword file is provided in Table 27.1. Detailed descriptions for the individual keywords are provided in Section 27.3.

Table 27.1 Example Keyword File for STOCHASTIC

```
REPORT "Step1.Rpt"
USER "John Q. Doe"
TITLE "Generate Median Values for Vadose Zone Stochastic Data"
SEED 12456564.0
REALIZAT 1000001
SINGLEKEY PERCENT=0.50 FILE="Step1.Val"
!
STOCHASTIC "30" "X" 1   1.000
STOCHASTIC "31" "X" 10  20
     0.00000E+00  1.000000E+00
     5.00000E-01  1.000000E+00
     5.54012E-01  1.044167E+00
     6.04938E-01  1.088333E+00
     6.52778E-01  1.132500E+00
     6.97531E-01  1.176667E+00
     7.39198E-01  1.220833E+00
     7.77778E-01  1.265000E+00
     8.13272E-01  1.309167E+00
     8.45679E-01  1.353333E+00
     8.75000E-01  1.397500E+00
     9.01235E-01  1.441667E+00
     9.24383E-01  1.485833E+00
     9.44444E-01  1.530000E+00
     9.61420E-01  1.574167E+00
     9.75309E-01  1.618333E+00
     9.86111E-01  1.662500E+00
     9.93827E-01  1.706667E+00
     9.98457E-01  1.750833E+00
     1.00000E+00  1.795000E+00
STOCHASTIC "52" "X" 6   1.000  2.590  16.900
!
END
```

27.2.2 Output Files

The STOCHASTIC code writes from one to three output files:

1. **Report File.** The report file is an ASCII file containing information about the run of the STOCHASTIC program. It is written for every run of the code. All error messages (except for those indicating the report file could not be opened) are written to this file.
2. **SINGLEKEY File.** A revised STOCHASTIC keyword can optionally be written to an output file for every input STOCHASTIC keyword. Output to this file is controlled by the SINGLEKEY keyword. The file contains the new keywords in text format. A typical use of this

keyword is to convert the input STOCHASTIC keywords, each defined for a statistical distribution, into new keywords defining a constant set to the median of the input distribution.

3. **Values File.** The generated stochastic values can optionally be written to an output file. Output to this file is controlled by the DEBUG keyword. The file contains the generated values in text format.

27.3 Keyword Definitions for the STOCHASTIC Code

In general, the keywords can be entered in any order. The following restrictions apply on keyword order:

- The REPORT keyword must be the first keyword in the file.
- The END keyword must be the last keyword in the file.

27.3.1 DEBUG Keyword for STOCHASTIC

The DEBUG keyword is used to activate dumping of intermediate calculations to the report file and to output generated values. It should be used sparingly because large output files can result. The following is this keyword's syntax:

```
DEBUG {DEFINITION}{STATISTICS}{STOCHASTIC=“Quote 1” {COLUMN}}
```

Multiple DEBUG cards can be entered with combinations of modifiers, or a single card can be entered containing all of the modifiers. The modifiers can be entered in any order. Table 27.2 describes the modifiers associated with the DEBUG keyword.

Table 27.2 Modifiers Associated with the DEBUG Keyword in STOCHASTIC

Modifier	Description
DEFINITION	The presence of the DEFINITION modifier on the DEBUG keyword will cause the definition of all stochastic variables to be written to the report file.
STATISTICS	The presence of the STATISTICS modifier on the DEBUG keyword will cause summary statistics for all generated variables to be computed and written to the report file.
STOCHASTIC	The presence of the STOCHASTIC modifier on the DEBUG keyword will cause all generated values to be written to the file identified in the associated quote string. If the modifier COLUMN is also present, the values will be written in a single column, otherwise, the values are written in comma separated format with all values for a single variable on one line.

The following sequence of keyword records illustrate the use of the DEBUG keyword:

```
DEBUG STOCHAST "Test_Median.Csv" COLUMN
DEBUG DEFINITI
DEBUG STATISTI
```

27.3.2 END Keyword for STOCHASTIC

The END keyword signifies the end of all keyword data. All data in the keyword file after the END keyword will be ignored. The following is this keyword's syntax:

```
END
```

27.3.3 REALIZATION Keyword for STOCHASTIC

The REALIZATION keyword defines the number of realizations to generate. The following is this keyword's syntax:

```
REALIZATION value1
```

The integer value1 has a minimum value of 1. No specific maximum number of realizations is enforced, but large values may result in large output files. Values up to 5,000,000 have been verified using a small number of variables.

The following keyword record sets the number of realizations to 1000:

```
REALIZAT 1000
```

27.3.4 REPORT Keyword for STOCHASTIC

The REPORT keyword is used to define the name of the output report (log) file. It must be the first keyword entered in the keyword file. The following is this keyword's syntax:

```
REPORT [“quote”]
```

The name of the report file is entered in a quote string. File names up to 200 characters long are supported, and path names can be optionally included. The following is an example REPORT keyword record:

```
REPORT “G:\SAC\Sys\EC\Test.rpt”
```

27.3.5 SEED Keyword for STOCHASTIC

The SEED keyword sets the value for the seed for the random number generator. The following is this keyword's syntax:

```
SEED Value1
```

The value for Value1 must be an integer or real number in the range 1 to 999999. The following is an example keyword record:

```
SEED 344443
```

27.3.6 SINGLEKEY Keyword for STOCHASTIC

The SINGLEKEY keyword is used to output an optional file of data for a user-defined sample percentile for all stochastic variables. This option supports generation of median-value data sets for the inventory, vadose zone, and impact codes. The following is this keyword's syntax:

```
   SINGLEKEY [PERCENT=Value1|MEAN] [FILE="quote1"]
```

Either the PERCENT modifier or the MEAN modifier is required. The numerical value associated with the modifier PERCENT is used to select the sample percentile for output. A value in the range 0 to 1 is allowed. A value of 0.5 will select the median generated value. The arithmetic mean of the generated values will be used for output if the MEAN modifier is used.

The quote string associated with the FILE modifier identifies the output file where the data will be written. The quote string must be enclosed in double quotation marks and file names up to 200 characters long are supported. The following example indicates that keywords using median values for two stochastic variables are to be written to the file named "Tmp_Key.Key":

```
   SINGLEKEY PERCENT=0.5 File="Tmp_Key.key"
   STOCHASTIC KDSOIL "KdI129"  6 0.0 4.0 15.0 "Soil-water Kd for I129"
   STOCHASTIC KDSOIL "KdCs137" 9 6.908 0.692 TRUNCATE 0.01 0.99 "Kd Cs137"
```

This example results in the following two output keywords:

```
   STOCHASTIC "KdI129"  1  5.9170E+00 "Soil-water Kd for I129"
   STOCHASTIC "KdCs137" 1  1.0002E+03 "Kd Cs137"
```

The following example indicates that keywords using the mean values for two stochastic variables are to be written to the file named "Tmp_Key.Key":

```
   SINGLEKEY MEAN File="Tmp_Key.key"
   STOCHASTIC KDSOIL "KdI129"  6 0.0 4.0 15.0 "Soil-water Kd for I129"
   STOCHASTIC KDSOIL "KdCs137" 9 6.908 0.692 TRUNCATE 0.01 0.99 "Kd Cs137"
```

This example results in the following two output keywords:

```
   STOCHASTIC "KdI129" 1  6.3333E+00 "Soil-water Kd for I129"
   STOCHASTIC "KdCs137" 1  1.2289E+03 "Kd Cs137"
```

27.3.7 STOCHASTIC Keyword for STOCHASTIC

The STOCHASTIC keyword is used to enter the definition of a statistical distribution for stochastic variables. The following is this keyword's syntax:

```
   STOCHASTIC [“Quote1”] [Dist_Index Parameters] {TRUNCATE U1 U2} {“Quote2”}
```

The entry for Quote1 must be a unique character string of up to 20 characters that will be used to identify this stochastic variable in subsequent uses. It is case sensitive and embedded spaces are significant. The entry for Quote2 is a description for the stochastic variable that can be up to 64 characters long. An entry for Quote2 is not required.

The entry for Dist_Index must be an integer in the range 1 to 13 that identifies the index of a statistical distribution. The available statistical distributions are defined in Table 27.3. The word Parameters in the general syntax statement indicates the numerical values of parameters required for defining the statistical distribution. The additional modifier TRUNCATE can be used for all distribution types except 1, 3, and 10 (constant, discrete uniform, and user defined). If TRUNCATE is entered, it must be followed by two values in the interval 0 to 1, inclusive. The lower value must be less than the upper value. These two values specify the tail probabilities at which to impose range truncation for the distribution. Truncation data must be entered after all of the other parameters that define the distribution. Further information on generation of stochastic variables is provided in Section 1.0.

Table 27.3 Statistical Distributions Available in STOCHASTIC

Index	Distribution	Truncate	Parameters Required
1	Constant	No	Single value.
2	Uniform	Yes	Lower limit, upper limit.
3	Discrete Uniform	No	Smallest integer, largest integer.
4	Loguniform (base 10)	Yes	Lower limit, upper limit.
5	Loguniform (base e)	Yes	Lower limit, upper limit.
6	Triangular	Yes	Lower limit, mode, upper limit.
7	Normal	Yes	Mean, standard deviation.
8	Lognormal (base 10)	Yes	Mean, standard deviation of logarithms.
9	Lognormal (base e)	Yes	Mean, standard deviation of logarithms.
10	User Defined	Yes	Number of pairs, data for pairs of values (Prob(X_i),X_i).
11	Beta	Yes	Alpha, beta, lower limit, upper limit. The mean of the distribution would be alpha/(alpha+beta) if the limits were 0 and 1.
12	Log ratio	Yes	Mean, standard deviation (of derived normal), lower limit, upper limit.
13	Hyperbolic arcsine	Yes	Mean, standard deviation (of derived normal).

The following is an example stochastic keyword for a variable assigned a constant of 234.432:

```
STOCHASTIC “Unique1” 1 234.432 “Define a constant distribution”
```

The following is an example stochastic card for a bioconcentration factor that is normally distributed with a mean of 125 and a standard deviation of 5 for a frog exposed to ^{14}C:

```
STOCHASTIC “BCFC14Frog” 7  125.0  5.0  “Example Distribution”
```

27.3.8 TITLE Keyword for STOCHASTIC

The TITLE keyword is used to define a single-line problem title. The problem title will be written to output files. The program will error terminate if the title is not supplied. The following is this keyword’s syntax:

```
TITLE [“quote”]
```

The title is entered in a quote string, which must be enclosed in double quotation marks. Titles up to 200 characters long are supported. The following example defines a title for a run of the code:

```
TITLE “Example title line for the INVSUM code.”
```

27.3.9 USER Keyword for STOCHASTIC

The USER keyword is used to identify the user of the program. The user name will be written to output files. The program will error terminate if the user name is not supplied. The following is this keyword’s syntax:

```
USER [“quote”]
```

The user name is entered in a quote string, which must be enclosed in double quotation marks. User names up to 16 characters long are supported. The following example defines John Q. Public as the user running the code:

```
USER “John Q. Public”
```

28.0 Sto_Med – Median Values for Stomp (Vadose Zone)

28.1 Overview

A run of the vadose zone flow and transport code STOMP with stochastic parameters set to the median value (50th percentile) of their range requires an input keyword file for the ESP code designed for a single realization. The input keyword file developed for the ESP code typically contains full stochastic distributions for the input variables. A sequence of computer codes is available that reads a full stochastic keyword file and writes the associated median-values keyword file.

28.1.1 Location in the Processing Sequence

Conversion of the stochastic keyword file into a median-values keyword file must occur before the vadose zone setup pass is performed by the ESP code.

28.1.2 How the Code Is Invoked

A sequence of three codes is used to convert the stochastic keyword file into a median-values keyword file. For purposes of discussion, let the stochastic keyword file be named “stochastic.key” and let the median-values keyword file be named “median.key”. The sequence of three codes is invoked as follows:

```
inv_step1.exe “stochastic.key”
stochastic.exe “Step1.Key”
inv_step2.exe “stochastic.key” “median.key”
```

The user does not need to prepare any input files or enter other commands to control this sequence of calculations.

28.1.3 Memory Requirements

Sto_step1 and sto_step2 have minimal memory requirements.

28.2 Computational Sequence

A sequence of three codes is used to convert the stochastic keyword file into a median-values keyword file. Invocation of the three codes are described in the following paragraphs.

28.2.1 Sto_Step1 Execution

The first step in the processing sequence is to run the STO_STEP1 code. This code reads the stochastic keyword file and writes a separate keyword file for the STOCHASTIC code. The user does not need to modify the output keyword file.

Under the Windows operating system (Releases 98, NT, 2000, or XP), STO_STEP1 executes in a DOS box. A run of STO_STEP1 is initiated by entering the following command line:

```
sto_step1 “stochastic.key”
```

Under the Linux operating system STO_STEP1 is executed through any of the following Bourne Shell or C Shell commands:

```
   sto_step1.exe “stochastic.key”
```

For these commands, STO_STEP1 or sto_step1.exe is the name of the executable program and “stochastic.key” is the name of the stochastic keyword file.

28.2.2 Stochastic Execution

The second step in the processing sequence is to run the STOCHASTIC code. This code reads the file named “Step1.Key” written by the STO_STEP1 code and writes a file named “Step1.Val” for use in the STO_STEP2 code.

Under the Windows operating system (Releases 98, NT, 2000, or XP), STOCHASTIC executes in a DOS box. A run of STOCHASTIC is initiated by entering the following command line:

```
   STOCHASTIC “Step1.Key”
```

Under the Linux operating system STOCHASTIC is executed through any of the following Bourne Shell or C Shell commands:

```
   stochastic.exe “Step1.Key”
```

For these commands, STOCHASTIC or stochastic.exe is the name of the executable program, “Step1.Key” is the name of the input keyword file and “Step1.Val” is the name of the output file of median values.

28.2.3 Sto_Step2 Execution

The final step in the processing sequence is to run the STO_STEP2 code. This code reads the stochastic keyword file, a file named “Step1.Val” written by the STOCHASTIC code, and writes the median-values keyword file.

Under the Windows operating system (Releases 98, NT, 2000, or XP), STO_STEP2 executes in a DOS box. A run of STO_STEP2 is initiated by entering the following command line:

```
   sto_step2 “stochastic.key” “median.key”
```

Under the Linux operating system STO_STEP2 is executed through any of the following Bourne Shell or C Shell commands:

```
   sto_step2.exe “stochastic.key” “median.key”
```

For these commands, STO_STEP2 or sto_step2.exe is the name of the executable program, "stochastic.key" is the name of the stochastic keyword file and "median.daf" is the name of the median-values keyword file.

29.0 VZGRAB – Groundwater/Vadose Zone Integration

29.1 Overview

The VZGRAB utility code extracts data associated with the vadose zone flow and transport model. Options include extracting control data from the ESD and VZGRAB keyword files, extracting data from vadose zone flow and transport output files, and formatting and exporting the extracted data for use in a spreadsheet or plotting program.

29.1.1 How the Code Is Invoked

VZGRAB only runs under the Linux operating system. VZGRAB is executed through the following Bourne Shell or C Shell command:

```
   gwgrab-1.exe “ESDKeyfilename” “VZGRABKeyFileName”
```

For the command, gwgrab-1.exe is the name of the executable program, and “ESDKeyfilename” is the name of the ESD keyword file providing overall control to the SAC simulation and “VZGRABKeyFileName” is the name of the VZGRAB control keyword file. If VZGRAB is invoked without two filenames the code will terminate execution after writing an error message to the standard output device. VZGRAB must be invoked from the root directory of a SAC analysis set and the ESD keyword file name must be the local file name (no path information). The keyword file name can include path information.

29.1.2 Memory Requirements

VZGRAB uses dynamic memory allocation. Code runs using large data sets (1000 waste sites, 11 analytes, 100 realizations) may cause memory demands to exceed 2.1 gigabytes, thereby causing the run to error terminate. In this case, multiple runs using a smaller set of extractions are required to extract all of the data.

29.2 File Definitions

The VZGRAB code reads two or more input files and writes one or more output files.

29.2.1 Input Files

The VZGRAB code reads the ESD keyword file, a VZGRAB-specific keyword file and optionally many other files in the problem directory structure. Only the VZGRAB keyword file is discussed in this section. The example VZGRAB keyword file provided in Table 29.1 will extract release data for a single site, a single analyte, and five realizations.

Table 29.1 VZGRAB Example Keywords for a Single Release Site

```
PATH “\file name\”            ! Tell VZGRAB where to put the output.
EXTRACT RELEASE         ! Extract release to groundwater data.
TIME ANNUAL TOTAL    ! Sum to calendar years and entire simulation period
SITE “216-B-11A#B”         ! Extract data for this site only.
ANALYTE ID=“H3”         ! Extract data for this analyte only.
REALIZATION 1 2 3 4 5      ! Extract data for these five realizations.
OUTPUT RAW           ! Report raw data for each selected realization.
END
```

The example VZGRAB keyword file provided in Table 29.2 will extract release data summed over all sites, a single analyte, and four realizations.

Table 29.2 VZGRAB Example Keywords for Summing Over All Release Sites

```
PATH “\file name\”   ! Tell VZGRAB where to put the output.
EXTRACT RELEASE      ! Extract release to groundwater data.
TIME ANNUAL TOTAL    ! Sum to calendar years & entire simulation period
SITE ALL SUM         ! Extract data for all sites and sum over all sites
ANALYTE “Tc99”       ! Extract data for this analyte only.
REALIZATION 1 2 3 4  ! Extract data for these four realizations.
OUTPUT RAW           ! Report raw data for each selected realization.
END
```

The example VZGRAB keyword file provided in Table 29.3 will extract release data summed over a list of user-specified sites, a single analyte, and three realizations.

Table 29.3 VZGRAB Example Keywords for Summing Over Selected Release Sites

```
PATH "\file name\"   ! Tell VZGRAB where to put the output
EXTRACT RELEASE      ! Extract release to groundwater data
TIME ANNUAL TOTAL    ! Sum to calendar years & entire simulation period
LIST   "216-B-3-2"   ! Extract data only for the sites in this list.
LIST   "216-B-3-3"
LIST   "216-B-3A"
LIST   "216-B-3B"
LIST   "216-B-3C"
ANALYTE  ID=“U238”   ! Extract data for this analyte only.
REALIZATION 1 2 3    !- Extract data for these three realizations only.
OUTPUT RAW           !- Report raw data for each selected realization.
END
```

29.2.2 Output Files

VZGRAB writes one or more output files. A text log file is created for each run that contains brief information about the job. For some combination of inputs the log file is the only output file written.

If annual release values are requested a separate output file will be created for every analyte requested. These files reside in the directory defined using the PATH keyword and have names with the following structure: vz_release_gw_ann_XXXXXX.csv. The string XXXXXX is replaced by the analyte ID for each analyte. If cumulative releases are requested a separate output file will be created for every analyte requested. These files reside in the directory defined using the PATH keyword and have names with the following structure: vz_release_gw_cum_XXXXXX.csv. The string XXXXXX is replaced by the analyte ID for each analyte. The annual and cumulative release files are written in comma-separated variable format and are intended to be imported into a spreadsheet or plotting package for further analysis.

29.3 Keyword Descriptions for VZGRAB

All user instructions regarding data extraction are supplied via the VZGRAB keyword control file and communicated by means of keywords. Keywords may be provided in any order. However, all information after the END keyword will be ignored.

29.3.1 ANALYTE Keyword for VZGRAB

The ANALYTE keyword is used to define the analyte (or analytes) for which data will be processed. The following is this keyword's syntax:

```
ANALYTE [[“quote 1”] {“qoute2”} … {“quoteN”} | ALL]
```

Several ANALYTE keywords may be entered or a single ANALYTE keyword with more than one ID can be used. The effect of multiple analyte keywords is cumulative. No quote strings are required if the modifier ALL is used because all analytes defined in the ESD keyword file for the analysis will be used. One or more quote strings containing the ID of analytes defined in the ESD keyword file must be entered if the ALL modifier is not used. Two examples of this keyword are the following:

```
ANALYTE “C14” “U238”
ANALYTE ALL
```

29.3.2 END Keyword for VZGRAB

The END keyword signifies the end of all keyword data. All data in the keyword file after the END keyword will be ignored. The following is this keyword's syntax:

```
END
```

29.3.3 EXTRACT Keyword for VZGRAB

The EXTRACT keyword defines what type of data is to be extracted. Several EXTRACT keywords with different modifiers may be included in one keyword file but a single EXTRACT keyword with more than one modifier is also acceptable. The following is this keyword's syntax:

```
EXTRACT [INQUIRE | INFLUX | BALANCE | RELEASE | REMEDIATE | PARAMETERS|
   PROFILE | LIMIT | RESTART]
```

The modifiers for the EXTRACT keyword are defined in Table 29.4.

Table 29.4 Modifiers Associated with the EXTRACT keyword in the VZGRAB Control File

Modifier	Description
INQUIRE	Returns information about the SAC run such as which analytes are defined and how many realizations were run.
INFLUX	Amount of analyte (kg) or amount of radiation (Ci) for radioactive analytes influx and amount of fluid (m^3) influx to the Vadose Zone.
BALANCE	Amount of analyte (kg) or amount of radiation (Ci) for radioactive analytes balance at SAC-wide times (use EXTRACT INQUIRE to determine balance times).
RELEASE	Amount of Analyte (kg) or amount of radiation (Ci) for radioactive analytes released to the groundwater zone.
REMEDIATE	Amount of Analyte (or amount of radiation) remediated from the Vadose Zone.
PARAMETERS	Values of stochastic input parameters.
PROFILE	Nodal STOMP concentrations reported by depth.
LIMIT	Checks all STOMP results to detect time-step-limited terminations.
RESTART	Checks the STOMP restart header files for correctness.

Example uses of this keyword are the following:

```
EXTRACT RELEASE
EXTRACT BALANCE
EXTRACT RELEASE BALANCE RESTART
```

29.3.4 LIST keyword for VZGRAB

The LIST keyword defines a list of user-selected sites to group for summation. The following is this keyword's syntax:

```
LIST “quote1” … “quoteN”
```

The SUM keyword should not be used if the LIST keyword is used. Several LIST keywords may be entered or a single LIST keyword with more than one ID can be used, however, data from several LIST keywords define a single composite list of sites. Each quote string must contain a valid site ID from the ESD keyword file. The following example identifies five sites to sum as a group:

```
LIST "100-B-15" "100-B-3"
LIST "100-B-5" "100-B-8" "100-C-3"
```

29.3.5 PATH keyword for VZGRAB

The PATH keyword defines the path to the directory where the extracted results are stored. The following is this keyword's syntax:

```
PATH  “path name”
```

The path name must be entered in a quote string and should end with the delimiter “/”. An example use of this keyword is the following:

```
PATH “/home/ANALYSIS4/CA1_median/vzgrab/”
```

29.3.6 OUTPUT keyword for VZGRAB

The OUTPUT keyword defines how to format the output data. The following is this keyword's syntax:

```
OUTPUT [RAW]
```

The modifier RAW must be entered on this keyword. It indicates that all data output will be raw data. An example use of this keyword is the following:

```
OUTPUT RAW
```

29.3.7 REALIZATION keyword for VZGRAB

The REALIZATION keyword defines which realization or realizations to report. The following is this keyword's syntax:

```
REALIZATION [N1 {N2} … {Nk}|ALL]
```

Several REALIZATION keywords may be entered or a single REALIZATION keyword with more than one number can be entered. If the modifier ALL is used all realizations defined in the ESD keyword file will be used. Otherwise, one or more realization numbers must be entered. Two example uses of this keyword are the following:

```
REALIZATION ALL ! Use all realizations
REALIZATION 5 2 7 100 ! Define 4 realizations explicitly
```

29.3.8 SITE Keyword for VZGRAB

The SITE keyword is used to define one or more sites to report on. The following is this keyword's syntax:

```
SITE [“quote1” {“quote2”} … {“quoteN”} | ALL SUM]
```

The LIST keyword should not be used if the SITE keyword is used. Data for each designated site will be reported separately if the modifier SUM is not used. If the modifiers ALL and SUM are used then the data will be reported as the sum over all sites in the ESD keyword file. The LIST keyword must be used to sum data for a subset of sites. The first example below reports data individually for five waste sites. The second example below reports data as the sum over all sites:

```
SITE "100-B-15" "100-B-3" "100-B-5" "100-B-8" "100-C-3"
SITE ALL SUM
```

29.3.9 TIME keyword for VZGRAB

The TIME keyword defines how to present the data with respect to time. The following is this keyword's syntax:

```
TIME [{ANNUAL} {TOTAL} {FROM year1 year2}]
```

Data will be summed on a calendar year basis if the modifier ANNUAL is present. Data will be summed for the entire simulation period if the modifier TOTAL is present. Data will also be summed over a range of years from year1 to year2 if the FROM modifier is present. Three example uses of this keyword are the following:

```
TIME ANNUAL
TIME ANNUAL TOTAL
TIME ANNUAL FROM 1944 TO 3000
```

Although both TOTAL and FROM modifiers can be used in a single run of the code the data for both options are written to the same file in an interlaced fashion. Using both the TOTAL and FROM options in the same run of the code is not recommended.

30.0 VZSWAP – Inventory Data Extraction

30.1 Overview

The VZSWAP utility code provides a convenient means to swap third-party vadose zone release results produced outside the SAC framework (for example, from site-specific performance assessment analyses) into or out of a SAC assessment.

30.1.1 Location in the Processing Sequence

The VZSWAP code can be executed after the SAC vadose zone model has been run.

30.1.2 How the Code Is Invoked

Prior to invoking the VZSWAP code, the user must place the third-party release to be swapped for SAC results in the appropriate /vadose subdirectories in files named ‘release.3RD' and prepare a swapfile that lists the /vadose subdirectories to operate upon.

The VZSWAP code is then invoked from this command and two arguments:

```
   vzswap-1.exe swapfile [3RD|SAC|SUM]
```

the executable name is vzswap-1.exe. The *swapfile* argument is the name of a text file containing a list of vadose zone directories to perform the swap operation. The second argument may be either 3RD, SAC, or SUM. The modifier 3RD is used to direct VZSWAP to place third-party release results into the ‘release’ file. The SAC modifier directs VZSWAP to restore original ‘release’ file contents generated by SAC. The SUM modifier directs VZSWAP to sum both the third-party release and the SAC release into a new summed release record.

To help prompt the user, help is provided if the second command argument is ‘help’, e.g.,

```
   vzswap-1.exe help
```

30.1.3 Memory Requirements

VZSWAP requires a trivial amount of memory.

30.2 File Definitions

The following files are utilized by VZSWAP:

- ***swapfile*** – an ASCII text file produced by the user that specifies the directories under the /vadose directory of a SAC assessment that will be operated on by VZSWAP
- **release** – the original release file produced by SAC using the STOMP code in a given /vadose subdirectory

- **release.SAC** –VZSWAP will store a copy of the original ‘release’ file found in a given /vadose subdirectory as ‘release.SAC’
- **release.3RD** – The file under which the 3rd-party release results are provided, in the same format as a STOMP ‘release’ file (except the #SAC header information is not required; VZSWAP will insert these lines)

31.0 Keyword Language Syntax

Each line of a keyword data file is parsed into numeric and character data. These are interpreted to set up control information and define input parameters. An input line can contain up to 256 characters of information. Lines longer than 256 characters are truncated to 180 characters.

Every line of the input data file is considered a keyword record, continuation record, or a comment record. Keyword records contain a keyword beginning in column 1. The keyword is used to determine the purpose of the subsequent data. Continuation records are used when a keyword record requires too much data to be placed on one line. Comment lines are ignored by the reading software but are useful for annotating the input file.

The information from each keyword record and subsequent continuation lines is moved into storage arrays. Data that can be deciphered as numeric values are placed into a numeric array. Other data are classified either as "secondary keywords" (called "modifiers") or "quote strings." Secondary keywords are stored as character images in an array. All such keywords or modifiers read from the input file are changed to uppercase before being stored. Quote strings are text strings that are enclosed in double quotation marks. These are stored exactly as they are read from the input file.

31.1 Keyword Records

Keyword records start in column 1 with any letter from A to Z, in either upper- or lowercase. The first eight letters of a keyword are stored in a variable and are used by the modeling software to determine the actions desired by the program user. All subsequent lines of text that do not have an alphabetic character or comment character in column 1 are treated as continuation lines. The following is an example keyword record (where SAMPLEKEY starts in column 1):

```
SAMPLEKEY 2 0 500 1 100
```

The word SAMPLEKEY is the keyword. The numbers 2, 0, 500, 1, and 100 are numeric data.

31.2 Continuation Records

Continuation records start with any valid separator character (except a double quotation mark). These are treated as additional data to the previous keyword record. Section 5.5 identifies valid separator characters. The combined data on a keyword line and on the subsequent continuation line(s) are treated as a single block of information. All numeric values and character strings on those lines are used as input data relevant to the keyword of the keyword line. The two following keyword entries contain the same information:

```
SAMPLEKEY 2 0 500 1 100

SAMPLEKEY 2 0
  500 1 100
```

31.3 Comment Records

Any line with the characters $, !, /, or * in column 1 will be treated as a comment record. These lines are ignored by the input data record reader. Both the $ and the ! can also be used to signify in-line comments (not in column 1). Any information that follows a $ or a ! will be ignored. The * and / characters indicate a comment only if one is the first character on the input line. The * character is also used as a repetition factor in the input. The following are some examples of comment lines:

```
$This is a comment line
/This is a comment line
!This is a comment line
*This is a comment line
```

The following are some examples of in-line comments:

```
SAMPLEKEY 3 4.0 5.0 !Trailing information is ignored after the !
SAMPLEKEY 3 4.0 5.0 $Trailing information is ignored after the $
```

31.4 Input Data Handling

Each line of the input is read and parsed into numeric and character values. All numeric values are converted to real numbers (as opposed to FORTRAN integer). All data that cannot be interpreted as numeric information are stored as character values.

Numeric data can include a leading sign (+ or -), integer characters 0 through 9, a decimal point, and an exponent indication (“E” or “e”). The FORTRAN “double precision” exponent indicator “D” is not valid. A maximum of 10 digits is allowed when entering numeric data.

Secondary keywords, or modifiers, are character strings that could not be interpreted as numeric values. These are converted to uppercase, where necessary, and stored in an array. The number of secondary keywords that are moved into the array is stored for internal use.

Only the first eight characters of any keyword are significant. Keywords fewer than eight characters long are left justified and blank-filled.

31.4.1 Quote Strings

Quote strings are strings of literal text that must be used exactly as given in the input line. They are enclosed by double quotation marks and are typically used for passing file names into a program. Only the first 200 characters of a quote string are saved. Each quote string must begin and end on a single line of the input file. When an unclosed quote is encountered, an implied quote is created at the end of that line. The following is an example of quote string usage:

```
FILE “c:\apps\human\test.dat”
```

31.4.2 Data Separators

Keywords, numeric data, and secondary keywords must be separated by any one of the following data "separators": space character, comma, equal sign, colon, semicolon, left parenthesis, right parenthesis, single quotation mark, double quotation mark, and tab character.

Also, any character with a ASCII character storage code of less than 10 is treated as a separator character. This is used mainly to identify the ASCII tab character as a data separator. Double quotation marks are used differently than the other separators. They indicate text strings that are stored without conversion by the program. As an illustration of the use of separator characters, the following keyword records all contain the same information:

```
SAMPLEKEY 3 4.5 5.6 6.7
SAMPLEKEY 3 (4.5,5.6,6.7)
SAMPLEKEY 3 ( 4.5=5.6'6.7 )
SAMPLEKEY 3:4.5  5.6:6.7
```

32.0 Stochastic Variable Generation

Many of the codes in SAC, Rev. 1, generate values for stochastic variables. All of the codes use the same suite of statistical routines to do this generation. The following are some major considerations for this process:

- Each distribution is generated using the Probability Integral Transformation method (Mood et al. 1974, p. 202).
- The uniform number generator uses a linear congruential method (Lewis, Goodman, and Miller 1969).
- Stratified sampling is used when the number of values to be generated is greater than 1.
- Most distributions may be truncated between two limits that are specified as limits in the uniform domain on the interval 0 to 1.
- The user may specify a cumulative distribution function in the form of a table of values.
- Information about a stochastic variable is linked to a unique character ID. Access to all information about the variable is available through use of the variable ID.

The following are the available statistical distributions:

- Constant value
- Uniform distribution between two limits
- Discrete uniform distribution on a set of contiguous integers
- Loguniform (base 10) distribution between two limits
- Loguniform (base e) distribution between two limits
- Triangular distribution defined using a lower limit, mode, and an upper limit
- Normal distribution with a mean and standard deviation
- Lognormal (base 10) distribution specified by the mean and standard deviation of the logarithms of the data
- Lognormal (base e) distribution specified by the mean and standard deviation of the logarithms of the data
- User-specified cumulative distribution function input as a table of probabilities and exceedance values
- Beta distribution that can be shifted and scaled from the standard (0,1) interval
- Log-ratio from a normal distribution
- Hyperbolic arcsine from a normal distribution

32.1 Keywords Defining Stochastic Variables

Depending on the utility code, stochastic variables can be defined for STOCHAST, KDSOIL, DILUTE, POROSITY, SATURATI, HYDRAULI, SORPTION, and RETARDAT keywords. The following general description is presented for STOCHAST keyword although it applies to each of the other keywords. The following is this keyword's syntax:

```
STOCHASTIC [{ID=}“Quote1”] [Dist_Index Parameters] {TRUNCATE U1 U2}
   {{LABEL=}“Quote2”} {UNITS=“quote3”}
```

The entry for Quote1 must be a unique character string of up to 20 characters that will be used to identify this stochastic variable in subsequent uses. It is case sensitive and embedded spaces are significant. It is sometimes useful to make the character string some combination of a variable name and other data so that it can be recreated easily when stochastic data is needed. The entry for Quote2 is an optional description for the stochastic variable that can be up to 64 characters long and is used for output labeling purposes.

The entry for Dist_Index must be an integer in the range 1 to 13 that identifies the index of a statistical distribution. The statistical distributions are defined in Table 32.1. The word Parameters in the general syntax statement indicates the numerical values of parameters required for defining the statistical distribution. The additional modifier TRUNCATE can be used for all distribution types except 1, 3, and 10. If TRUNCATE is entered, it must be followed by two values in the interval 0 to 1, inclusive. The lower value must be less than the upper value. These two values specify the tail probabilities at which to impose range truncation for the distribution. Truncation data must be entered after all of the other parameters that define the distribution.

Table 32.1 Statistical Distributions Available in All Impacts Codes

Index	Distribution	Truncate	Parameters Required
1	Constant	No	Single value.
2	Uniform	Yes	Lower limit, upper limit.
3	Discrete Uniform	No	Smallest integer, largest integer.
4	Loguniform (base 10)	Yes	Lower limit, upper limit.
5	Loguniform (base e)	Yes	Lower limit, upper limit.
6	Triangular	Yes	Lower limit, mode, upper limit.
7	Normal	Yes	Mean, standard deviation.
8	Lognormal (base 10)	Yes	Mean, standard deviation of logarithms.
9	Lognormal (base e)	Yes	Mean, standard deviation of logarithms.
10	User Defined	Yes	Number of pairs, data for pairs of values (Prob(X_i),X_i).
11	Beta	Yes	Alpha, beta, lower limit, upper limit. The mean of the distribution would be alpha/(alpha+beta) if the limits were 0 and 1.
12	Log ratio	Yes	Mean, standard deviation (of normal), lower limit, upper limit.
13	Hyperbolic arcsine	Yes	Mean, standard deviation (of normal).

The following is an example stochastic keyword for a variable assigned a constant of 234.432:

```
   STOCHASTIC “Unique1” 1 234.432 “Define a constant distribution”
```

The constant can take any value.

The following is an example stochastic keyword for a variable assigned a uniform distribution on –2 to 7:

```
   STOCHASTIC ID="Unique2" 2 –2.0 7 "Uniform distribution on –2 to 7"
```

The two limits can take any values as long as the second value is strictly greater than the first value. The following is an example stochastic keyword for a variable assigned a discrete uniform distribution on the integers 6 to 70:

```
   STOCHASTIC "Unique3" 3 6 70 "Discrete uniform distribution on 6 to 70"
```

The two limits must be integers where the second integer is strictly greater than the first integer.

The following is an example stochastic keyword for a variable assigned a loguniform (base 10) distribution on the interval 10^{-7} to 10^{-3}:

```
   STOCHASTIC ID="Unique4" 4 1.0E-7 1.0E-3
      LABEL="Define a loguniform (base 10) variable on 0.0000001 to 0.001"
```

The two limits must both be greater than zero and the second limit must be greater than the first limit.

The following is an example stochastic keyword for a variable assigned a loguniform (base e) distribution on the interval 10^3 to 10^6:

```
   STOCHASTIC "Unique5" 5  1.0E3  1E+6
      "Define a loguniform (base e) distribution on 1000 to 1000000"
```

The two limits must both be greater than zero and the second limit must be greater than the first limit.

The following is an example stochastic keyword for a variable assigned a triangular distribution with a minimum of 2, a mode of 3, and a maximum of 7:

```
   STOCHASTIC "Unique6" 6  2 3 7  "Triangular distribution on (2,3,6)"
```

The three values that define the triangular must all be different, and they must be entered in increasing order.

The following is an example stochastic keyword for a bioconcentration factor that is normally distributed with a mean of 125 and a standard deviation of 5 for a frog exposed to ^{14}C:

```
   STOCHASTIC "BCFC14Frog" 7  125.0  5.0  "Example normally distributed frog"
```

The mean value can be any number, but the standard deviation must be greater than zero.

The following keyword would define a different stochastic variable from the one just entered because the identification string (Quote1) is case sensitive:

```
   STOCHASTIC "BCFC14FROG" 7  125.0  5.0  "Example normally distributed frog"
```

The following keyword entry would define a lognormal (base 10) distribution where the mean and standard deviation (of the logarithms) are –2.0 and 0.5:

```
   STOCHASTIC “Unique8”  8  –2  0.5 “Lognormal (base 10) variable”
```

The mean value can be any number, but the standard deviation must be greater than zero.

The following keyword entry would define a lognormal (base e) distribution where the mean and standard deviation (of the logarithms) are –2.0 and 0.5. In addition, the lognormal distribution will be truncated between the lower 0.025 and upper 0.99 probabilities.

```
   STOCHASTIC “Unique9” 9 –2 .5 TRUNCATE 0.025 0.99
      “Example for a truncated lognormal variable”
```

The mean value can be any number, but the standard deviation must be greater than zero.

The following keyword entry illustrates the use of the user-defined distribution (distribution type 10). This example entry uses seven pairs of values. The first pair of numbers uses a probability of 0 to define the lower limit of the distribution at 8.4 E-7. The last pair of numbers uses a probability of 1 to define the upper limit of the distribution at 1.73E-6. The other values are associated with the probability levels of .025, .167, .5, .833, and .975. The probability data and distribution percentiles must be entered in strictly increasing order.

```
   STOCHASTIC "Sr90Con" 10  7
      0  8.40E-7
      2.50E-02  9.20E-7
      1.67E-01  1.06E-6
      5.00E-01  1.21E-6
      8.33E-01  1.37E-6
      9.75E-01  1.58E-6
      1  1.73E-6
```

The first pair of numbers uses a probability of 0 to define the lower limit of the distribution. The last pair of numbers uses a probability of 1 to define the upper limit of the distribution. The intervening pairs define probability levels and the associated data values. The probabilities and data values must be entered in strictly increasing order.

The following keyword entry would define a beta distribution with parameters 1.1 and 2.1 on the interval (0,1):

```
   STOCHASTIC "Unique11-1"  11  1.1  2.1  0.0  1.0
      "Beta (1.1,2.1) on the interval 0,1"
```

Let the first parameter be denoted by α and the second parameter be denoted by β. The mean of the beta distribution would be $\alpha /(\alpha +\beta)$ if the limits were 0 and 1. Both α and β must be greater than zero. The lower limit must be less than the upper limit.

The following keyword entry would define a beta distribution with parameters 1.1 and 2.1 but on the interval –2 to 4:

```
   STOCHASTIC "Unique11-2"  11  1.1  2.1  -2.0  4.0
      "Beta (1.1,2.1) on the interval (-2,4)"
```

The following keyword entry would define a log ratio distribution from a normal (-1.459,1.523) distribution on the interval -5.756 to 4.33:

```
   STOCHASTIC "Test1203" 12 -1.459  1.523 -5.756 4.330
      "Log ratio from Normal(-1.4,1.5) on (-5.756,4.330)"
```

The entry for the normal standard deviation (a value of 1.523 in this example) must be greater than zero. The last two numerical values define the interval for the generated values, so the lower limit must be smaller than the upper limit.

The following keyword entry would define a hyperbolic arcsine distribution from a normal (0.189,0.146) distribution:

```
   STOCHASTIC "Test1302" 13 0.189  0.146
      "Hyperbolic Arcsine from Normal(0.189,0.146)"
```

The entry for the normal standard deviation (a value of 0.189 in this example) must be greater than zero.

32.2 Probability Concepts

The distribution of a continuous random variable X (the term *continuous* indicates that the random variable is defined over a continuum of values) is completely described by its probability density function, f(x). The interpretation of the probability density function is that the area under f(x), for an interval a<x<b, equals the probability that the random variable, X, will fall in the interval (a,b), denoted P[a<X<b]. One cannot make the statement P[X=t], because there is no area under the probability density function at any given point. Two axioms of probability theory (Mood et al. 1974, p. 22) are that the probability of any event is between zero and one, and the integral of the probability density function over the entire support (the interval [L,U]) of X equals 1. The integral of the probability density function from the lower bound L to some value x (suppose it is less than the upper bound U) represents the probability that X will be observed in the interval (L,x). This integral operation defines the cumulative distribution function for the random variable X. The cumulative distribution function is denoted by F(x) (the capital F for the cumulative distribution function corresponds to the lowercase f for the probability density function) and mathematically is represented by:

$$F(x) = \int_{L}^{U} f(s)ds$$

The inverse cumulative distribution function, $[F^{-1}(\bullet)]$, is single-valued if x is in the interval (L,U). Hence if p'= F(x') is known, in theory x'= F^{-1}(p') can be obtained.

32.3 Probability Integral Transform Method

Generation of a random variable from a given distribution typically involves the use of information either about f or F. There are two philosophical approaches to generating random numbers: exact methods and approximate methods. The algorithms embedded in SAC employ exact methods. Exact methods can be further categorized into probability integral transform methods and functional methods. The probability integral transform method is employed in SAC.

In the probability integral transform method, the random variable of interest is expressed as a function of a U(0,1) random variable, where U(0,1) denotes the continuous random variable ranging uniformly over the interval (0,1). The probability density function of the uniform random variable is g(u)=1 if u0(0,1) and is zero elsewhere. The cumulative distribution function for this random variable takes the exceedingly simple form G(u)=u. It can be shown that any cumulative distribution function evaluated at a random value X (instead of being evaluated at a known value x as in the previous discussion) is distributed uniformly over the interval (0,1) (Mood et al. 1974, p. 202). Therefore, given a realization u of the U(0,1) random variable and a selected statistical distribution (known cumulative distribution function), one can set u=F(x) and solve to obtain $x=F^{-1}(u)$. The value x thus obtained is a random realization from the selected statistical distribution.

In principle, one can obtain an exact solution for x given any specific cumulative distribution function and value u. In reality, there exist some distributions, such as the normal and beta distributions, for which no closed-form analytical expression for F^{-1} exists, and hence approximation methods must be applied.

The probability integral transform method allows efficient sampling from a subregion of the interval (L,U), such as (c,d), where L<c<d<U. In this case one would find the corresponding interval in the uniform domain, say (c',d'), and sample uniformly over that interval, by sampling from the rescaled uniform distribution [for example, u'=(d-c)u+c] and then obtaining x as usual using $x=F^{-1}(u')$. The rescaled uniform distribution takes the form g(u)=1/(d'-c') for u0(c',d') and is zero elsewhere. For any distribution with probability density function f(x) and cumulative distribution function F(x), the probability density function, under truncation to the interval (c,d), is $f_T(x) = f(x)/[F(d)-F(c)]$. The divisor ensures that $f_T(x)$ integrates to unity.

32.4 Stratified Sampling

Stratified sampling can easily be implemented when generating random deviates using the probability integral transform method. This is accomplished by dividing the uniform interval (0,1) into subintervals, or strata, and sampling a specified number of times within each stratum, each time obtaining the corresponding value of x. Within SAC, the strata intervals are assigned equal probability, and exactly one value is sampled within each stratum. The method generates samples from each stratum, and then randomly shuffles the entire set of realizations using a variation of the Quicksort algorithm (Hoare 1961). The primary purpose of stratified sampling is to achieve more evenly spaced (in a probability sense) samples from the distribution of a random variable than would result from randomly sampling over the whole range of the distribution. Iman and Conover (1982) have shown that stratified sampling can result in more efficient estimation of simulation results for a variety of estimators than when using simple random sampling.

32.5 Generation Algorithms

Table 27.3 summarizes the statistical distributions available in the SAC codes. The following paragraphs describe the generation algorithms.

32.5.1 Algorithm for the Uniform Distribution

Algorithms that generate truly random uniform numbers do not exist, although many algorithms generate pseudo-random deviates (hereafter loosely referred to as random numbers). The selection of a random-number generator is based on four considerations: 1) computer implementability, 2) degree of independence within a sequence of deviates, 3) periodicity or cyclic length of a sequence, and 4) uniform coverage of sequences (occurrence) over the interval (0,1), the square (0,1) X (0,1), etc., up to the hypercube (0,1) in k dimensions.

Commonly used random-number generation techniques are linear congruential methods (Park and Miller 1988). The SAC codes use a linear congruential random number generator. The linear congruential generator generates random integers from an algorithm of the form $S_i = (A \,.\, S_{i-1} + C) \bmod M$. where S_i is the ith generated random integer, A and C are constants, M is the modulus of the generated integers, and mod denotes the remainder function. These integers are converted to approximate uniform (0,1) numbers by the division $U_i = S_i / M$.

The period of a sequence $\{U_i\}$ of generated deviates is the minimal value k such that $U_i=U_{i+k}$ (this occurs independent of i for linear congruential generators). It can be shown that the period of any congruential generator does not exceed M. Therefore, if one is generating many uniform deviates, it is desirable that M be large. The performance of each congruential generator (each choice of A, C, and M) can thus be examined with respect to criteria proceeding from the four considerations given above. The SAC implementation uses A=16807, C=0, and M=2147483647. These choices yield a sequence $\{U_i\}$ that 1) is implementable on a 32-bit computer without machine language coding, 2) is sufficiently independent on an element-by-element basis, 3) possesses a long cycle (period), and 4) has a reasonable degree of coverage over all hypercubes of dimension less than k. These conclusions proceed from results from tests described in Fishman and Moore (1986).

Any value, x, generated from the uniform (a,b) distribution in the SAC codes makes use of a value, y, from the U(0,1) distribution. The value y is first generated, and then x is evaluated as x=a+(b-a)y. The uniform (a,b) distribution will be denoted by U(a,b).

32.5.2 Algorithm for the Discrete Uniform Distribution

The probability density function for the discrete uniform distribution is f(x)=1/N for each of the N integers ranging in the interval L to U. The generation algorithm for the discrete uniform distribution is:

$$F^{-1}(u) = L + \mathrm{int}[u(U - L + 1)]$$

where the int(·) function returns the integer portion of its argument.

32.5.3 Algorithm for the Loguniform Distribution

The probability density function for the loguniform random variable of base b is:

$$f(x) = \frac{I(b^c < x < b^d)}{x(d-c) \bullet \ln(b)}$$

for -4<c<d<4, where I is an indicator function (0 if false, 1 if true), b is the logarithm base (either 10 or the natural constant e), and ln(b) denotes the natural logarithm of b.

The inverse cumulative distribution function algorithm used to generate a value, x, from the loguniform distribution first generates a value, y, from the U(c,d) distribution and then evaluates the expression $x=b^u$.

32.5.4 Algorithms for the Triangular Distribution

The triangular distribution has probability density function:

$$f(x) = \begin{cases} 2(x-a)/[(b-a)(c-a)] & \text{for} \quad a < x \le b \\ 2(c-x)/[(c-b)(c-a)] & \text{for} \quad b \le x < c \end{cases}$$

and takes the value 0 elsewhere.

The following equation provides the generation algorithm for the triangular distribution:

$$F^{-1}(u) = \begin{cases} a + \sqrt{u(c-a)(b-a)} & \text{for} \quad 0 \le u \le (b - a)/(c - a) \\ c - \sqrt{(1-u)(c-a)(c-b)} & \text{for} \quad (b - a)/(c - a) \le u \le 1 \end{cases}$$

32.5.5 Algorithms for the Normal Distribution

A normal (μ,**F**^{2}) random deviate, y, is obtained by generating a N(0,1) deviate, x, and then transforming that value using the equation y=μ+**F**x. The normally distributed random variable with mean μ and variance &^{2}, denoted as N(μ,**F**), has the probability density function:

$$f(x) = \frac{e^{-(x-\mu)^2/(2\sigma^2)}}{\sigma\sqrt{2\pi}}$$

for -4<x<4, -4<μ<4 and **F**>0.

The inverse cumulative distribution function for the N(0,1) random variable does not have a closed form expression. It is approximated by:

$$F^{-1}(p) = \begin{cases} q\,A(q^2)/B(q^2) & \text{for } |q| < 0.42 \\ \operatorname{sgn}(q)\,C(r)/D(r) & \text{otherwise} \end{cases}$$

where q=p-0.5, and r=√ln(0.5-|q|). The quantity (0.5-|q|) is formed as p or, to avoid cancellation if p is small, as (1-p). The letters A, B, C, and D represent polynomials of order 3, 4, 3, and 2, respectively, whose coefficients are given in Beasley and Springer (1985), and sgn(q)=1 if q>0 and -1 if q<0.

32.5.6 Algorithms for the Lognormal Distribution

The logarithm of a random variable that is lognormally distributed is distributed as a normal N(µ,**F**^{2}) random variable (thus the name lognormal). The probability density function of the lognormal distribution is:

$$f(x) = \frac{A}{x\sigma\sqrt{2\pi}} e^{-[\log(x)-\mu]^2/2\sigma^2}$$

for x>0 and &>0. Because this distribution is available in both base 10 and natural logarithm base form, the constant A is 1/log$_e$10 for base 10 and 1 for the natural logarithm base. The logarithm log(x) is also evaluated in terms of the chosen base.

A lognormal random variable, x, is generated using a two-step process. First, a value, y, is generated from the N(µ,**F**^{2}) distribution. This value is then used in the expression x=b^y, where the base b is either 10 or the natural constant e as desired.

32.5.7 Algorithms for the User-Defined Distribution

In addition to selecting from parametric families of distributions, the user may implement any other distribution by supplying a table of data pairs corresponding to the pairs [F(x),x]. Thus, the user provides the SAC code with discrete evaluations of the cumulative distribution function. The algorithm linearly interpolates between these points to solve for F^{-1} when generating a random deviate.

32.5.8 Algorithms for the Beta Distribution

The beta random variable, x, is described by the probability density function:

$$f(x) = \frac{x^{p-1}(1-x)^{q-1}}{B(p,q)}$$

for p>0, q>0, and 0<x<1. This variable can be transformed to the interval (a,b) and the resulting probability density function for the random variable, y, takes the form:

$$f(x) = \frac{(b-a)^{-(p+q+1)}(y-a)^{p-1}(b-y)^{q-1}}{B(p,q)}$$

for p>0, q>0, and a<y<b. The second expression for the probability density function can be obtained from the first by the change of variable y=(b-a)x+a.

A closed form expression for the beta inverse cumulative distribution function does not exist. The algorithm implemented is provided in Algorithm AS 64/AS 109 (Griffiths and Hill 1985, p. 121). Algorithm AS 64/AS 109 requires the logarithm of B(p,q). Using the relationship between the beta and gamma functions (Mood et al. 1974, p.535), B(p,q)=G(p+q)/{G(p)G(q)}, where G(·) denotes the gamma function, the logarithm of B(p,q) is computed using Algorithm ACM 291 (Pike and Hill 1966). Algorithm AS 64/AS 109 uses approximate starting values and a Newton-Rhapson iterative method to achieve a final solution.

32.5.9 Algorithms for the Log Ratio Distribution

Let y denote a random variable from the log ratio distribution on the interval (a,b). The probability density function for y is:

$$f(y) = \frac{(b-a)e^{-(\ln\{(y-a)/(b-y)\}-\mu)^2/(2\sigma^2)}}{\sigma(y-a)(b-y)\sqrt{2\pi}}$$

where a<b and a<y<b.

A random variable, y, from the log ratio distribution is generated using a two-step process. First, a value, x, is generated from the N(μ,**F**^{2}) distribution. This value is then used in the expression:

$$F^{-1}(x) = \frac{a+be^x}{1+e^x}$$

32.5.10 Algorithms for the Hyperbolic Arcsine Distribution

Let the random variable x be a normally distributed random variable with mean μ and variance &^{2}. Then, let y be a random variable defined as y = sinh^{-1}(x). The probability density function for y is:

$$f(y) = \frac{0.5(e^u + e^{-u})e^{-(\sinh(y)-\mu)^2/(2\sigma^2)}}{\sigma\sqrt{2\pi}}$$

for -4<y<4, -4<μ<4 and **F**^{2}>0.

A random variable, y, from the hyperbolic arcsine distribution is generated using a two-step process. First, a value, x, is generated from the $N(\mu,\mathbf{F}^2)$ distribution. This value is used in the expression:

$$F^{-1}(x) = \sinh(x)$$

33.0 References

Beasley JD and SG Springer. 1985. "Algorithm AS 111, The Percentage Points of the Normal Distribution" in *Applied Statistics Algorithms*, P Griffiths and ID Hill (eds.). Ellis Horwood Limited, Chichester, England.

DOE. 1998. *Screening Assessment and Requirements for a Comprehensive Assessment, Columbia River Comprehensive Assessment*. DOE/RL-96-16, U.S. Department of Energy, Richland, Washington.

DOE. 1999a. *Radioactive Waste Management Manual*. DOE M 435.1-1, U.S. Department of Energy, Washington, D.C.

DOE. 1999b. *Implementation Guide for Use with DOE M 435.1-1*. DOE G 435.1-1, U.S.Department of Energy, Washington, D.C.

DOE Order 435.1. 2001. *Radioactive Waste Management*. U. S. Department of Energy Headquarters, Washington, D.C. Available on the internet at http://www.directives.doe.gov/pdfs/doe/doetext/neword/435/o4351c1.html

EPA. 1998. *Guidelines for Ecological Risk Assessment*. EPA/630/R-95/002F, Risk Assessment Forum, U.S. Environmental Protection Agency, Washington, D.C. (also published in the Federal Register 63(93):26846-26924).

Eslinger, PW, DW Engel, LH Gerhardstein, CA Lo Presti, TB Miley, WE Nichols, DL Strenge and SK Wurstner. 2004a. *User Instructions for the Systems Assessment Capability, Rev. 1, Computer Codes. Volume 1: Inventory, Release, and Transport Modules*. PNNL-14852,Volume 1, Pacific Northwest National Laboratory, Richland, Washington.

Eslinger PW, TB Miley, C Arimescu, and BA Kanyid. 2004b. *User Instructions for the Systems Assessment Capability, Rev. 1, Computer Codes. Volume 2: Impact Modules*. PNNL-14852, Volume 2, Pacific Northwest National Laboratory, Richland, Washington.

Firestone M, P Fenner-Crisp, T Barry, D Bennet, S Chang, M Callahan, A Burke, J Michaud, M Olsen, P Cirone, D Barnes, WP Wood, and SM Knott. 1997. *Guiding Principles for Monte Carlo Analysis*. EPA/630/R-97/001, Risk Assessment Forum, U.S. Environmental Protection Agency, Washington, D.C.

Fishman GS and LR Moore. 1986. "An Exhaustive Analysis of Multiplicative Congruential Random Number Generators with Modulus 2^{31}-1" in *Siam Journal of Scientific and Statistical Computing* 7(1):24-44.

Griffiths P and ID Hill (eds.). 1985. *Applied Statistics Algorithms*, Ellis Horwood Limited, Chichester, England .

Hoare CAR. 1961. "Partition: Algorithm 63," "Quicksort: Algorithm 64," and "Find: Algorithm 65." in *Comm. ACM* **4**, 321-322.

Iman RL and WJ Conover. 1982. “A Distribution-free Approach to Inducing Rank Correlations Among Input Variables" in *Communications in Statistics,* Vol. B11, No. 3, pp. 311-334.

Lewis PA, AS Goodman, and JM Miller. 1969. “A Pseudo-random Number Generator for the System/360” in *IBM Systems Journal*, Vol. 8, No. 2, pp. 136-145.

Mood AM, FA Graybill, and DC Boes. 1974. *Introduction to the Theory of Statistics*, Third Edition. McGraw-Hill Book Co., New York.

Park SK and KW Miller. 1988. "Random Number Generators: Good Ones are Hard to Find" in *Comm. of the ACM*, Vol. 31. No. 10, pp 1192-1201.

Pike MC and ID Hill. 1966. "ACM Algorithm 291: Logarithm of Gamma Function" in *Communications of the ACM*, Vol. 9, No. 9, p. 684.

Simpson BC, RA Corbin, and SF Agnew. 2001. *Hanford Soil Inventory Model.* BHI-01496 Rev 0., Bechtel Hanford Inc., Richland, Washington.

PNNL-14852, Volume 3

Distribution

U.S. Department of Energy, RL

B. L. Charboneau (CD)	A6-33
R. D. Hildebrand (CD)	A5-13
J. G. Morse (CD)	A6-38
DOE Public Reading Room (2P)	H2-53

U.S. Department of Energy, ORP

R. W. Lober (CD)	H6-60

CH2M HILL

F. J. Anderson (CD)	E6-35
T. J. Knepp (CD)	H6-03
F. Mann (CD)	E6-35

Fluor Federal Services

R. Puigh (CD)	E6-17

Fluor Hanford, Inc.

T. W. Fogwell (CD)	E6-35

Pacific Northwest National Laboratory

C. Arimescu (1P)	K6-52
R. L. Aaberg (CD)	K3-54
M. P. Bergeron (CD)	K9-36
R. W. Bryce (3CD/P)	K6-75
D. W. Engel (CD/P)	K5-12
P. W. Eslinger (5 CD/5P)	K6-80
B. A. Kanyid (CD)	K7-22
C. T. Kincaid (CD/P)	K9-33
C. A. Lo Presti (CD)	K5-12
T. B. Miley (3CD/3P)	K6-80
B. A. Napier (CD)	K3-54
W. E. Nichols (CD/P)	K9-33
M. C. Richmond (CD)	K9-33
R. G. Riley (P)	K6-96
D. L. Strenge (CD)	K3-54
S. K. Wurstner (CD/P)	K9-36
Hanford Technical Library (2)	P8-55