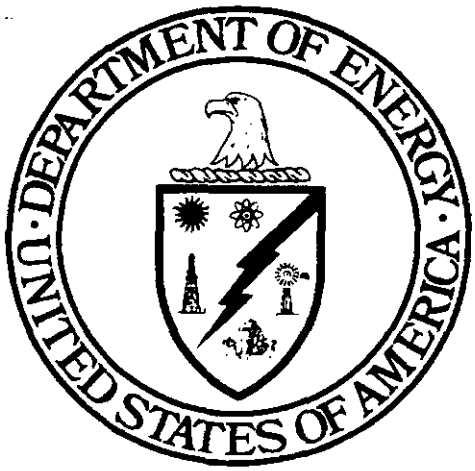
Title 40 CFR Part 191
Compliance Certification
Application
for the
Waste Isolation Pilot Plant

Appendix CUTTINGS



United States Department of Energy
Waste Isolation Pilot Plant

Carlsbad Area Office
Carlsbad, New Mexico

WIPP PA User's Manual for CUTTINGS_S

WIPP PA

User's Manual

for

CUTTINGS_S, Version 5.03

DOCUMENT VERSION 1.00

WPO #37765

22 MAY 1996

CONTENTS

1.0 INTRODUCTION

This document is the User's Manual for the WIPP code called CUTTINGS_S* (version number 5.03). Using repository-height data from SANTOS via BRAGFLO, CUTTINGS_S estimates the volume of wastes brought to the surface as a result of an inadvertent borehole drilled directly over the WIPP repository so as to penetrate a waste panel. The user's manual discusses the code, its execution, and its performance in the context of the 1996 Waste Isolation Pilot Project (WIPP) Compliance Certification Application (CCA) Performance Assessment (PA), and in that context only. The manual identifies the code, its authors and expert consultants (Section 1). It describes the code's WIPP-PA purposes and functions (Section 2), provides recommended user training (Section 3), outlines the code's theoretical basis and numerical methods (Section 4), its inherent capabilities and limitations (Section 5), describes user interactions (Section 6), input files (Section 7), error messages (Section 8), output files (Section 9), and useful reference materials (Section 10). It provides examples of relevant input, output, and debug files in its Appendices as well as a set of sample test problems to familiarize the user with the run sequence.

It is useful to remember that the present version of CUTTINGS_S's evolved in time through a sequence of WIPP PAs. Design criteria for each PA differed slightly from the set before, and careful readers will detect, especially in the appendices, terminology, test cases, and parameter values such as radioisotope decay data, that serve no apparent purpose in the *present* version of the code. Outdated but working subroutines were normally retained within the code so as to provide versatility. Similarly, some portions of the code (for example, the "stuck-pipe" mode) are working and tested, but they do not come into play because of the parameter values in effect for the CCA calculation. Portions of the code that are not actually called and used in the CCA PA calculations are not to be regarded as part of this software quality-assurance procedure, even though references to them may appear on files included in the appendices.

1.1 Software Identifier:

Code Name: CUTTINGS_S, a borehole waste-removal code for cuttings, cavings, and spallings.

WIPP Prefix: CUSP

Version Number: 5.03

Date: 23/ May/ 96

Platform: FORTRAN 77 for Open VMS AXP, ver. 6.1, on a DEC Alpha.

* The final "_S" in the name signifies "spallings", whose effects the code includes.

1.2 Points of Contact:

Sponsor: R. A. Cole
New Mexico Engineering Research Institute (NMERI)
851 University Blvd. SE, Suite 101
Albuquerque NM 87106 - 4339
Voice: 505 272 7243 (NMERI) Fax: 505 272 7203

Consultants: J. W. Berglund
New Mexico Engineering Research Institute (NMERI)
851 University Blvd. SE, Suite 101
Albuquerque NM 87106 - 4339
Voice: 505 272 7228 (NMERI) Fax: 505 272 7203

and

J. S. Rath
New Mexico Engineering Research Institute (NMERI)
851 University Blvd. SE, Suite 101
Albuquerque NM 87106 - 4339
Voice: 505 272 7226 (NMERI) Fax: 505 272 7203
Voice: 505 766 9629 (Geo-Centers) Fax: 505 766 9125,

2.0 FUNCTIONAL REQUIREMENTS

The functional requirements stated below are identical to those listed in CUSP's Requirements Document (WPO#: 24320).

2.1 Functional Requirements

R.1 CUSP calculates the amount of repository material brought to the surface due to erosion of the borehole resulting from laminar flow in the drilling fluid.

R.2 CUSP calculates the amount of repository material brought to the surface due to erosion of the borehole resulting from turbulent flow in the drilling fluid.

R.3 CUSP calculates the amount of repository material brought to the surface due to blowout of the borehole.

R.4 CUSP calculates radionuclide chains as a result of initial inventory and radioactive decay, based on the time of intrusion.

R.5 CUSP calculates model specific parameter values based on experimental data.

2.2 Performance Requirements

There are no performance requirements for CUSP.

2.3 Attribute Requirements

There are no attribute requirements for CUSP.

2.4 External Interface Requirements

R.6 CUSP utilizes routines from CAMDAT_LIB, CAMCON_LIB, SDBREAD_LIB, and CAMSUPES_LIB. Consequently it must be linked with these libraries.

R.7 CUSP requires one CDB input file from the BRAGFLO code, CUSP_INP$BRAGCDB

R.8 CUSP requires one input file containing preliminary data base information, CUSP_INP$CDB.

R.9 CUSP requires one input file containing model and site dependent parameters and radionuclide properties, inventories, drilling procedures, and characteristics of the drilling fluid, CUSP_INP$TXT0.

R.10 CUSP requires one input file identifying input sample vector values that will be used in the analysis, CUSP_INP$TXT1.

R.11 CUSP generates one output file CUSP_OUT$DBG, which contains information that is used for comparing with acceptance criteria.

R.12 CUSP generates one output file CUSP_OUT$NVERIFY, which contains information that is used in the functional testing for hand calculations, and is not used in production runs.

R.13 CUSP generates one output CDB (binary) file, CUSP_OUT$CDB, containing output generated by the code.

2.5 Other Requirements

There are no other requirements for CUSP.

3.0 REQUIRED USER TRAINING AND/OR BACKGROUND

To exercise CUTTINGS_S, users should have (1) basic knowledge of open VMS, (2) basic facility with Digital Command Language, and (3) an overall understanding of Sandia's CAMDAT database, which is used in virtually every WIPP code (Rechard, 1992; and Rechard et al., 1993). User's should also have (4) access to the WIPP cluster of DEC Alpha microcomputers with an open VMS AXP (Ver. 6.1) operating system or their functional equivalents.

To manipulate and/or interpret the results of CUTTINGS_S as it is exercised in WIPP PAs, user's should have (1) a basic understanding of the physics of radioactive decay, the mechanics of laminar and turbulent flow of non-Newtonian fluid suspensions in annular conduits, erosion scouring of matrixed materials, flow through permeable media, and of the large-scale deformation and failure of elastic, plastic, and viscoelastic materials (2) a basic understanding of partial-differential equations and integral calculus, as they apply in the mathematical formulation of physical principles and especially of conservation of momentum, energy, and mass*, (3) a generalist's conceptual understanding of numerical methods as they are applied to the numerical solution of the boundary-value problems of mathematical physics and chemistry, and (4) a basic overview understanding of the WIPP PA process, including conceptual models, scenarios, inventories, uncertainty sampling, input-data vectors, and a general familiarity with the files and functions of the WIPP codes that interface with CUTTINGS_S and especially GENMESH, MATSET, POSTLHS, and BRAGFLO (WIPP PA Dept [5 Volumes], 1992).

* Strictly speaking, the decay of radioisotopes is not mass conserving. It requires loss of mass, although the amounts are insignificant for present purposes. Fluid and gas flows associated with CUTTINGS_S are mass conserving.

4.0 DESCRIPTION OF THE MODEL AND METHODS

The discussion of the model and methods given below is but a brief overview of the theory and methods employed by CUTTINGS_S. It is intended for users who have only a cursory interest in the mathematical and fluid dynamical details. A thorough, detailed treatment of the theory and numerics applied in CUTTINGS_S is given in Appendix A of this manual.

4.1 Description of the Model

CUTTINGS_S is a multi-faceted computational procedure to assess the effects of *direct removal* of wastes buried in a WIPP-like nuclear-waste repository. Direct removal is hypothesized to occur as the result of inadvertent penetration of the repository by a borehole drilled in connection with oil or gas exploration at some unknown time in the future (Berglund, 1993). The word "direct" refers to the fact that CUTTINGS_S removals from the repository to the surface occur *at the time* of drilling. Since drilling operations are normally completed on the time scale of weeks, removal of drilled wastes from the repository to the surface (i.e., the accessible environment) takes place entirely within a single timestep of the much-longer-term groundwater transport codes. Hence, the word "direct" should be taken to mean "from the repository to the surface within a single timestep of a typical WIPP groundwater code."

Removal is subdivided according to process, which results in three principal categories, herein labeled A, B, and C. First, the borehole is assumed (A) to penetrate directly through waste containers stored within the repository. The drill bit cuts directly through the waste containers, allowing those repository wastes located on a collision course with the drill bit (called cuttings) to be transported directly and immediately to the surface via the cooling fluid (muddy brine) that is circulated through the borehole during drilling. The volume of repository material removed as cuttings is the portion of the deformed repository cut by the borehole.

Wastes that originally bordered the borehole on its exterior are permitted to erode into the cooling fluid as a result of the fluid's frictional effects on the outer cylindrical surface of the borehole. If shear stresses in the cooling fluid at the wall exceed the shear strength of the matrixed materials located at the wall, (B) erosion will occur and wall material (cavings) will be removed*. As a result, the borehole diameter may increase locally at the level of the repository. Because of that action, the volume of wastes removed from the repository can actually be larger than the volume of the borehole originally cut through the wastes. As the diameter of the borehole increases, the fluid shear stresses at the outer boundary decrease. If they decrease enough that the failure stress of the wall material equals or exceeds the fluid shear stress at the outer boundary of the flow region, then the caving process will terminate. CUTTINGS_S allows for both laminar and turbulent shear flow in the circulating fluids, which, in turn, leads to two different models to assess the effects of material removal due to erosion. If the cooling-fluid flow is laminar, CUTTINGS_S employs an analytical solution of the modeling equations to

* In tightly matrixed or crystaline media such as the WIPP-site halite formation, exploratory boreholes are usually unlined. Rather, the drilling fluid is changed to saturated salt solution to minimize dissolution at the walls, and the borehole is permitted to support itself until the drill bit reaches granular strata below the halite horizon.

evaluate wall stresses and other flow parameters. The analytical solution takes the form of non-linear integral equations, which CUTTINGS_S evaluates via an iterative numerical procedure. If the flow is turbulent, CUTTINGS_S turns to a simpler empirical model to evaluate the turbulent shear stresses. Transition to turbulence is assumed to occur at a Reynolds number greater than 2100. In nature, the transition to turbulence actually occurs over a range of Reynolds numbers and is not an abrupt process. Normally there is a transitional regime during which the flow is partly laminar and partly turbulent with active and chaotic transitions between the two states. CUTTINGS_S does not attempt to model the transition regime. Rather, it assumes the flow becomes fully and abruptly turbulent when the Reynolds exceeds 2100. CUTTINGS_S employs a corrective procedure to compensate for the effects of that assumption (see Appendix A for details). To date, all WIPP-like runs have required turbulent cooling-fluid flow.

The cuttings/cavings model described thus far is functionally similar to the CUTTINGS model used in previous WIPP PAs (WIPP PA Dept, 1992, Vol 1). However, an important new element has been added to the newer CUTTINGS_S code, namely, the inclusion of a third material-removal mechanism, removal in the form of spallings, which is discussed below.

If the repository is pressurized at the time of penetration, and if the pressure within the repository is high enough (i.e., above the hydrostatic pressure in the cooling-fluid column), repository materials originally surrounding the borehole, but interior to the repository-fluid interface, can be (C) forced to deform inward toward the borehole so as to replace the materials already removed by action of (A) the cutting bit and (B) the erosive brine. The solid fraction of these new materials are called spallings*. Depending on the parameters of the problem, spalling deformations can be rapid, or they can be a sluggish, more like a large-amplitude, viscoelastic flow. If repository materials are relatively impermeable, the spalled materials can deform slowly inward so as to press against the rotating drill string. If repository pressures are high enough, frictional forces against the drill string may be high enough to stop the drill from turning. In that event, drillers normally withdraw the drill (vertically) and redrill downward through the spalled materials that clog the borehole. On occasion, that procedure must be repeated more than once. If repository materials are more permeable, the pressurized gas may simply seep through the matrix carrying loose particles with it. Alternatively, if repository pressures are too low, the spallings part of CUTTINGS_S may do nothing at all, in which case CUTTINGS_S will operate as its parent code, CUTTINGS, operated. The possibilities are varied, and CUTTINGS_S is designed to be applicable in any of the various hypothesized WIPP-drilling modes. Thus, CUTTINGS_S is perhaps best thought of as several related models, applicable to several sets of drilling circumstances, rather than as one all-encompassing, single-mode model. Moreover, CUTTINGS_S may have to be applied more than once during an assessment run, at more than one intrusion time, since multiple borehole intrusions are permissible in WIPP PAs.

* If the repository is wet with brine at time of penetration, which possibility is allowed in WIPP PAs, three phases of material could, in principle, flow toward the borehole, namely: pressurized gases, pressurized brine, and solid wastes. CUTTINGS_S estimates only the amount of solids that are moved into the borehole. Pressurized brine that may be released into the borehole is treated by an indepenednt WIPP code named DBR_BRAGFLO. The gas flow, although crucial to the overall deformational process, transports no radioisotopes and is therfore not assessed.

The total volume of repository material removed by CUTTINGS_S is the sum over all intrusions of (A) the material directly in line with the borehole that is directly cut by the drill bit (cuttings), (B) the material directly surrounding the borehole that is eroded into the borehole through the shearing action of the flowing cooling fluid (cavings), and (C) the material that was originally remote from but in the neighborhood of the borehole and was deformed toward and eventually into the borehole by the action of repository gas pressures (spallings). Options (A) and (B) are assumed to operate continuously, whereas option (C) operates only when repository pressure conditions permit it (the cut off being 8MPa.). The amount of material removed by each process, during each hypothesized drilling intrusion is reported in CUTTINGS_S's output in terms of an equivalent area. Downstream WIPP codes, namely CCDFGF, (1) use BRAGFLO's repository-height data (from SANTOS) to convert the equivalent areas to volumes of repository material, and then (2) assign various sampled aged repository contents to convert these material volumes to releases. Thus, (3) CCDFGF evaluates the release effects of striking contact-handled versus remote-handled wastes. The reader is referred to Appendix A for details.

The fluid-flow portion of CUTTINGS_S is based on a non-linear analytical integral solution for laminar helical flow of a non-Newtonian (Oldroyd-type) fluid in an annular pipe, and on empirical equations for turbulent flow in the same geometrical configuration. CUTTINGS_S acquires its parameter data from (i) the input CDB files provided by the WIPP codes MATSET and BRAGFLO, (ii) a user-supplied input text file, and (iii) the carefully controlled and QAed WIPP property database. Discrete values of sampled data are provided by the WIPP sampling code POSTLHS in cases where parameter uncertainty is judged to impact assessments of the repository's performance. CUTTINGS_S input includes the rotational speed of the drill string, cutting-bit diameter, the shear-rate-dependent viscosity and density parameters for the drilling mud, borehole roughness, the compacted repository thickness and porosity, and the effective failure shear strength of the compacted repository material. A few required parameters, namely drilling-mud flow rate and drill-collar diameter, are calculated internally by the code, based on present-day oil-drilling industry guidelines.

4.2 Description of the Modeling Methods

CUTTINGS_S does not endeavor to integrate conservation equations characterizing the flow on a point-for-point, time-for-time basis in search of a detailed space-time history of the removal process. Rather, removal is treated as instantaneous, that is, within a single timestep of the longer-term groundwater codes. Thus, numerical methods are used primarily to determine the limiting values of the physical parameters that represent the final configurations of the various (sampled) drilled systems, which is all that is required to assess radioactive waste removal due to the cuttings, cavings, and spallings that arise during drilling operations. These methods are fairly straightforward, iterative, recursive approximation techniques that involve systems of algebraic equations. They are discussed thoroughly in Appendix A (Sections 2.2.1.1, 2.2.2.1, and 2.4.1.2.1), which is highly recommended to readers who aim to understand this code.

5.0 INHERENT CAPABILITIES AND LIMITATIONS OF THE SOFTWARE

CUTTINGS_S is not presently put forward as a general borehole-mechanics code. Rather, it was developed with WIPP-like conditions and applications firmly in mind. To function properly, it requires inputs from other WIPP PA codes, specifically GENMESH, MATSET, POSTLHS, and BRAGFLO. Thus, although CUTTINGS_S is quite likely to be fully applicable to other kinds of boreholes, other geologies, and other parameter ranges, it is not presently recommended for applications outside the presently-conceived boundaries that define the WIPP, primarily because such applications have not yet been tested. Consequently, no claims are made that CUTTINGS_S should give reasonable results for parameter ranges that are atypical of the WIPP and its immediate environs.

All aspects of CUTTINGS_S are based on the present-day drilling methods and technologies that would apply at a WIPP-like site. CUTTINGS_S has been tested under those conditions and those conditions only. Hence, applications beyond the limits of those assumptions are not recommended at this time.

6.0 USER INTERACTIONS WITH THE SOFTWARE

Before launching into the exercising of CUTTINGS_S, the reader is reminded that CUTTINGS_S is normally exercised as one of a sequence of codes that includes GENMESH, MATSET, and POSTLHS, and requires input data files that originate with the code BRAGFLO. To assist the reader, the code sequence is depicted in figure 1, and examples of the input control files required by GENMESH and MATSET in preparation for a CUTTINGS_S run are given in Appendix B. POSTLHS requires an input control file, but it is empty.

6.1 Exercising CUTTINGS_S Interactively

CUTTINGS_S has been modified in comparison with its parent code, CUTTINGS, so as to remove the possibility of user-interactive runs. This change is in keeping with preparations for the 1996 CCA PA in which no interactive runs are foreseen.

To execute CUTTINGS_S from a command line, it is necessary to define a logical symbol that invokes its executable, by typing either of the following two lines at the VMS $ prompt:

$CUSP :== "$WP$PRODROOT:[CUSP.EXE]CUSPPA96.EXE" <CR>*

or

$CUSP :== "$WP$TESTROOT:[CUSP.EXE]CUSP_TEST_NODBG.EXE"

The two executables are equivalent. Only their locations on the system differ. The first executable resides in the production-run (the CCA PA) portion of the system, and the second resides in the Quality-Assurance (QA) test-problem Configuration-Management-System (CMS) portion of the system. No matter which definition is used, typing "$CUSP" and striking the carriage-return key will result in invoking CUTTINGS_S's executable.

Like all WIPP codes, CUTTINGS_S must be made aware of the names of and paths to the suite of input files it requires in order to exercise. It must also be given user-selected names for the files to which it will write its output. In command-line execution mode, CUTTINGS_S will search for files by reviewing its internal list of logical symbols associated with input/output files. Thus, the user must inform CUTTINGS_S which of its list of internal logical symbols corresponds to which input or output file for that specific run. That information is normally transferred by typing (at the VMS $ prompt that will appear after defining the executable logical) a line such as the following:

$DEFINE CUSP_INP$CDB XY:[ABC.DEF.GHI]JKL.MNO

* In this user's manual, VMS commands will always be entered by striking the carriage-return key. That instruction will not be repeated hereafter.

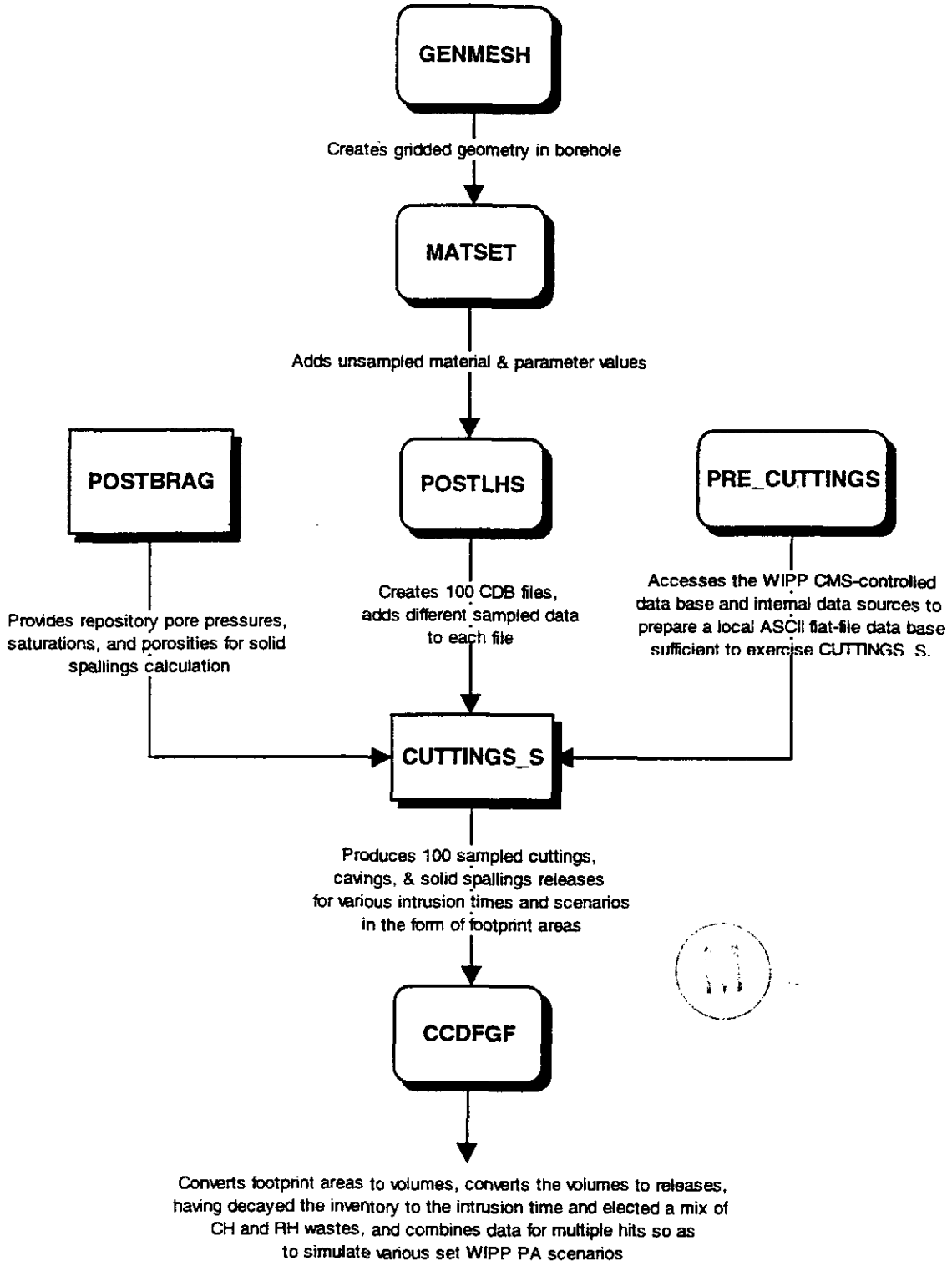


Figure 1: The code sequence for CUTTINGS_S in the 1996 CCA PA.

where XY:[ABC.DEF.GHI]JKL.MNO represents the specific input file the user wishes to associate with the input logical CUSP_INP$CDB . The path need be included only if the file resides other then in the local working-level directory.

6.2 CUTTINGS_S's Input/Output File Structure

Up to 11 logical symbols may be associated with CUTTINGS_S's various input/output files. Of them, 6 are required and 5 are optional in regulatory runs. Before discussing the nature and function of the 11 possible input/output files, we first list them one-by-one, for ready reference, together with their corresponding logical symbols. They are:

File ID No.	Input/Output File Names	Associated Logical Symbol	Is the file Required or Not?
1.	Binary input CAMDAT file (from MATSET)	CUSP_INP$CDB	Yes
2.	Input control file (text) specifying repository/model parameters, initial inventories, generic radioisotope database.	CUSP_INP$TXT0	Yes
3.	Input control file (text) specifying drilling and intrusion parameters	CUSP_INP$TXT1	Yes
4.	BRAGFLO binary output .CDB file.	CUSP_INP$BRAGCDB	
	Regulatory runs		Yes
	Test runs		No
5.	CUTTINGS_S's binary output .CDB file	CUSP_OUT$CDB	Yes
6.	CUTTINGS_S's output debug text file	CUSP_OUT$DBG	Yes
7.	Text file for verification testing	CUSP_OUT$NVERIFY	No
8.	Radioisotope inventory history (in CURIES) output text file	CUSP_OUT$IHISTO	No
9.	Normalized-release-history output text file (EPA 40CFR191, Subpt B) file in ASCII format	CUSP_OUT$NHISTO	No
10.	Time-history plotting text file for:		
	Long half-life radioisotope decay	CUSP_OUT$PLT1	No
	Medium half-life radioisotope decay	CUSP_OUT$PLT2	No
	Short half-life radioisotope decay	CUSP_OUT$PLT3	No
11.	Output text file for transfer to the WIPP code CCDFPERM	CUSP_OUT$ICTRN	No

The first four files of the above list are CUTTINGS_S's input data and control files. They are described briefly in Section 7.0 of this user's manual. The text-format input files include specific examples printed out in full in the appendices of this user's manual. Appendix C is included to assist the user in understanding the format used in CUTTINGS_S's input control files (file numbers 1 and 2).

The last seven files of the above list are CUTTINGS_S's output files. They are described briefly in Section 9.0 of this user's manual. Of the required output files, only the debug file is created in text format. One example of a CUTTINGS_S output file is included in Appendix D. It is a debug file. It is annotated with italicized remarks that identify and explain the various groups of data that are reported therein.

6.3 Exercising CUTTINGS_S

Once the logical symbol CUSP has been defined (see Section 6.1), and the necessary input and output files have been properly assigned (see Sections 6.1 and 6.2), CUTTINGS_S may be exercised simply by typing the logical symbol:

$CUSP

followed by a carriage return. If all goes well, CUTTINGS_S will run and issue a "normal completion" message. If errors are encountered, CUTTINGS_S will abort and issue an error message(s). See Section 8.0 for a listing of the possible error messages CUTTINGS_S can produce.

6.4 Exercising CUTTINGS_S's Test Problems in Batch Mode

To run all five of CUTTINGS_S's QA test problems in a single run, type the following command sequence:

$LIBCUSP

$CFE CUSP_TEST_ALL.COM

$@CUSP_TEST_ALL "Generation Number"

At this point, a command file that defines all of the input and output files associated with the five QA test problems will be invoked, and the five problems will be executed. As a result, all of the various input files and output files generated in the process will be available for examination. It is recommended that new users carry out the above exercise and familiarize themselves with the results before going on to regulatory-type calculations. The command @CUSP_TEST_ALL may have a "generation number" appended as a parameter. In the absence of a generation number, CUTTINGS_S chooses the current test-problem set. If earlier test-problem sets are of interest, they may be specified by appending a generation number to the command.

To run any single test problem, type the following command sequence:

$LIBCUSP

$CFE CUSP_TEST_BAT.COM

$@CUSP_TEST_BAT "Problem Number"

The specified test run will now exercise and produce for the user's perusal the input and output files associated with the specified "Problem Number."

To run the calibration problem; the following command sequence is typed:

$CFE CUSP_TEST_CALIBRATE.COM

$@CUSP_TEST_CALIBRATION

The calibration problem exercises in roughly 10 to 15 minutes, and produces two output files. The first output file is named CUSP_TEST_CAL_EXP.VER and is an echo of the input file. The second output file is named CUSP_TEST_CAL_EXP.PLT and provides a history of the search to find minimized values of FSE, FGE, and FCE (see Appendix A for definitions of nomenclature).

6.5 Using CUTTINGS_S's PRE_CUTTINGS Mode

Because interactions with the controlled WIPP database require large and unnecessary expenditures of input/output time during production runs, CUTTINGS_S includes a special operational mode designed to expedite data acquisitions during production runs. It is called PRE_CUTTINGS, and in it, the WIPP QAed property database is accessed but once during an entire production run. PRE_CUTTINGS reads a template file that interfaces the WIPP property database, extracting the required values and writing them to an ASCII file that can be read time and again by CUTTINGS_S without subsequent reference to the property database. The ASCII data file thus produced is under strict CMS control.

PRE_CUTTINGS's template file, which is shown in Appendix E, treats three data types. The first type is exemplified by CUTTINGS_S's code parameters, which are discussed in the theoretical Manual (see Appendix A). These values are essential to exercising CUTTINGS_S, but were not entered as part of the WIPP's CCA controlled database. They are discussed in detail in Appendix A. The second type are common property values as taken from the WIPP's CCA controlled database. For example, the datum BOREHOLE:L1 is called. In this example, the colon at mid word identifies the form as a WIPP controlled database property. BOREHOLE is the material identifier and L1 is the parameter identifier. Using these two keywords, the WIPP property database is scanned until the numerical value associated with that specific parameter is

located. CUTTINGS_S then writes to the new template the parameter identifier L1 and the corresponding numerical value that was found in the database.

The third data type in the input template file includes the 10 radioisotope decay chains and the 29 specific radioisotopes that have initial inventories, which are denoted by the word "SAVE." CUTTINGS_S tracks selected data for all of these radioisotopes. When CUTTINGS_S encounters the key word "GENERATE_RADIO", it sorts the list alphabetically and for each unique radioisotope, it queries the property database for atomic weight, half life, EPA release limit, initial contact-handled inventories, and initial remote-handled inventories. The code converts the half life in seconds to half life in years and calculates the specific activation energy, which is a function of atomic weight and half life. The code then writes all the necessary radioisotopic properties in tabular form. Appendix F is an example of the output ASCII text file from PRE_CUTTINGS. Appendix G is a sampled "COM" file to test this portion of the code. Because this functionality of the code is basically data in / data out, it was not assigned a specific test problem.

6.6 Using CUTTINGS_S's CHECK_OUT Mode

CUTTINGS_S has a special mode, called "CUSP$CHECK_OUT," designed to check the agreement between Model 2's blowout calculation and corresponding experimental results. CHECK_OUT was added so the blowout model could be used for confirmation of experiential results. The CHECK_OUT mode is set by specifying "CUSP$CHECK_OUT" as TRUE in DEC's Digital Command Language (DCL). Appendix H provides a listing of the DCL nomenclature required to exercise this operational mode. The CUTTINGS_S input file CUSP_INP$TXT0 provides data for the experimental input, the geometry of the test fixture, and the properties of air. It is also shown in Appendix H. In CHECK_OUT mode, three functions can be performed, as follows:

1. The "EXPERIMENT" function is set by including the keyword EXPERIMENT in the CUSP_INP$TXT1 text file. It allows CUTTING_S to be run as though data is being gathered from an experimental run.

2. The "VALIDATE" function is set by including the keyword VALIDATE in the CUSP_INP$TXT1 text file. It allows CUTTINGS_S to be run over the total range of possible inputs and thereby checks the code for numerical instabilities.

3. The "CALIBRATE" function is set by including this keyword CALIBRATE in the CUSP_INP$TXT1 text file. This function searches for the optimum values of FSE & FGE (see Appendix A) to minimize the error between the experiential results and the computer model. The CALIBRATE function is illustrated as a test problem #6 in the Validation Document.

The input files for these three functions are also shown in Appendix H.

7.0 DESCRIPTION OF CUTTINGS_S'S INPUT FILES

CUTTINGS_S's input files are listed in Section 6.2 together with their corresponding CUTTINGS_S logical symbols. They are file numbers 1 through 4 in that listing.

File 1. CUSP_INP$CDB is the principal input CDB file and is absolutely required by CUTTINGS_S. It is the file to which CUTTINGS_S's results will be transferred and it is normally provided by an upstream code such as MATSET. In production runs, sampled data from POSTLHS would be fed to this input file. In that case, there would be N input CDB files, one for each of the sampled values where N is the number of samples employed. In test-case mode or debug mode, it is inconvenient to open the editor and hand build an input CDB file for specific input values. However, even in test or debug mode, CUTTINGS_S expects to receive an input CDB file. As a convenience to users, an all-purpose, input CDB file for use with test problems and in debug runs is stored in the configuration-management-system (CMS) portion of the WIPP Alpha cluster. It is named: CUSP_TEST_1.CDB and can be called by typing:

$LIBCUSP

$CFE CUSP_TEST_1.CDB

File 2. CUSP_INP$TXT0 is an ASCII input control file that contains various repository, drill geometry, and material input parameters required by CUTTINGS_S. File 2 also contains the required radioisotope inventory data for the scenario currently under study. In regulatory calculations, these data are taken directly from the controlled WIPP property database. File 2 also provides radioisotope data essential for (1) the Bateman (1910) decay calculation, for (2) calculation of radioactive release values in Curies, and for (3) normalizing calculated releases in terms of the Environmental Protection Agency's (EPA's) 40CFR191 release limits (WIPP PA Dept [Vol 1], 1992). A sample CUSP_INP$TXT0 file usable for QA test and/or debug problems is available in the CMS portion of the WIPP Alpha cluster. It is named: CUSP_TEST_REPOSITORY.DAT , and it is printed in full in Appendix I of this user's manual. Most of its parameters are specific to the CUTTINGS_S model and to individual radioisotopic properties.

File 3. CUSP_INP$TXT1 is an ASCII input control file that contains various drilling and intrusion input parameters required by CUTTINGS_S. A sample CUSP_INP$TXT1 file usable for test and/or debug runs is available in the CMS portion of the WIPP Alpha cluster. It is named CUSP_TEST_#.INP where # is the QA test problem number and varies from 1 through 5. In a normal production run, this file would contain no specific input data other than an intrusion time. Instead, only character variables would appear, identifying the element property name on the input CDB file. Values would then be assigned from the secondary database (see item 4, below). Appendix J is sample printout of a typical production input file. For specific testing or

debugging, this file would have fixed specific numerical values. Appendix K is a listing of CUSP_TEST_2.INP.

File 4. CUSP_INP$BRAGCDB is the output CDB file from BRAGFLO as it is exercised in the undisturbed or disturbed scenarios. CUTTINGS_S extracts from it (1) the repository porosity at time of decommissioning, and (2) the repository porosity and (3) repository pressure at the specified intrusion time. In regulatory calculations, these data would be provided by a regulatory BRAGFLO run. For QA testing and debug purposes, a fixed BRAGFLO output CDB file has been stored in the CMS portion of the WIPP Alpha cluster. It is named CUSP_TEST_INP_BF.CDB , and it is the BRAGFLO output file used for test problem 1.

8.0 ERROR MESSAGES

Numerous procedural errors can cause CUTTINGS_S to abort. In general, these errors relate to the improper input of data. By noting where the code was when the abort occurred, and what the code was trying to read at the time, these problems can generally be resolved. However, there is not specific advice that can be offered to users in resolving errors beyond the general statement: locate the error and repair it. However, Appendix L, which was extracted primarily from the source code, contains useful operating notes and tips to the user. They are included purely because the code sponsor felt they might be useful to novice users.

To familiarize readers with CUTTINGS_S's error-message patterns, CUTTINGS_S's various error messages are listed below, section for section within the code.

SUBROUTINE ADDMEM(MORMEMR, MORMEMC)

```
      CALL QAABORT(
     $  '(ADDMEM) - REAL array size errors detected ***')
      CALL QAABORT(
     $  '(ADDMEM) - REAL array memory errors detected ***')
      CALL QAABORT(
     $  '(ADDMEM) - CHARACTER array size errors detected ***')
      CALL QAABORT(
     $  '(ADDMEM) - CHARACTER array memory errors detected ***')
      CALL QAABORT('(ADDMEM1) - REAL array size errors detected ***')
      CALL QAABORT('(ADDMEM1) - REAL memory errors detected ***')
      CALL QAABORT(
     $  '(ADDMEM1) - CHARACTER array size errors detected ***')
      CALL QAABORT(
     $  '(ADDMEM1) - CHARACTER array memory errors detected ***')
      CALL QAABORT('(ADDMEM2) - REAL array size errors detected ***')
      CALL QAABORT('(ADDMEM2) - REAL memory errors detected ***')
      CALL QAABORT(
     $  '(ADDMEM2) - CHARACTER array size errors detected ***')
      CALL QAABORT(
     $  '(ADDMEM2) - CHARACTER array memory errors detected ***')
      CALL QAABORT('(ADDMEM3) - REAL array size errors detected ***')
      CALL QAABORT('(ADDMEM3) - REAL memory errors detected ***')
      CALL QAABORT(
     $  '(ADDMEM3) - CHARACTER array size errors detected ***')
      CALL QAABORT(
     $  '(ADDMEM3) - CHARACTER array memory errors detected ***')
```

```
      SUBROUTINE BRGCDB( MAXC, MAXQ, IQ, LQ, Q, C, IERRDB )

      IF(IERRDB.NE.0)CALL QAABORT('(BRGCDB) - reading input CAMDAT '//
     $                    'sizing parameters ***')
      IF(IERRDB.NE.0)CALL QAABORT('(BRGCDB) - reading input CAMDAT '//
     $                    'TITLE ***')
      IF(IERRDB.NE.0)CALL QAABORT('(BRGCDB) - reading number of input'//
     $                    ' CAMDAT analysis variables ***')

------------------------------------------------------------------------
      SUBROUTINE DRILL( IWASFLG, IHIT, XTINT, DBDIAM, TUNITS, XNEWAREA)

          CALL QAABORT('(DRILL) - Negative ROUTER %%%')

------------------------------------------------------------------------
      SUBROUTINE GETVAL( NOUTFL, NUQPRP, NELBLK, VARNAM, IASPRP,
     $                XMATPR, NAMELB, NMATPR, XVAL    )

        CALL QAABORT('(GETVAL) - ILLEGAL I.C. interpolation '//
     +          'variable value range name ***')
------------------------------------------------------------------------
      INTEGER FUNCTION GINDX( RADNAM, IDMTRL )

        CALL QAABORT('(GINDX) - cannot locate nuclide index in '//
     $              'generic radioisotope data base (GRDB) ***')

------------------------------------------------------------------------
      INTEGER FUNCTION GJNDX( RADNAM, NUCNAM )

        CALL QAABORT('(GJNDX) - cannot locate nuclide index in '//
     $              'unique radioisotope names array (NCUNAM(*)) ***')

------------------------------------------------------------------------
      SUBROUTINE GSDBQA( FSDB, NQAREC, QAINFO )

        CALL QAABORT('(GSDBQA) - Error reading SDB version number '//
     $              'using SDBREAD_LIB routine: SDBQERRY ***')
        CALL QAABORT('(GSDBQA) - Error reading SDB geology name/t'//
     $              'ype using SDBREAD_LIB routine: SDBQERRY ***')

------------------------------------------------------------------------
```

PROGRAM CUSP_MAIN

```
 IF(IERR.NE.0)CALL QAABORT('(CUSP_MAIN) - Setting up DYNAMIC'//
$                  ' base arrays ***')
 IF(IERR.NE.0)CALL QAABORT('(CUSP_MAIN) - Opening input '//
$                  ' CAMDAT ***')
 IF(IERR.NE.0)CALL QAABORT('(CUSP_MAIN) - Setting up DYNAMIC'//
$                  ' base arrays ***')
 IF(IERR.NE.0)CALL QAABORT('(CUSP_MAIN) - Opening input '//
$                  ' CAMDAT ***')
 IF(IERR.NE.0)CALL QAABORT('(CUSP_MAIN) - opening CUTTINGS'//
$                  ' input text file 1 ***')
 IF(IERR.NE.0)CALL QAABORT('(CUSP_MAIN) - opening CUTTINGS'//
$                  ' input parameter text file  ***')
 IF(IERR.NE.0)CALL QAABORT('(CUSP_MAIN) - opening CUTTINGS'//
$                  ' input parameter binary file  ***')
 IF(IERRDB.NE.0) CALL QAABORT ('(BRGCDB) - Check BRAG CDB ***')
 IF(IERR.NE.0)CALL QAABORT('(CUSP_MAIN) - Opening output'//
$                  ' CAMDAT and SCRATCH files ***')
 IF(IERR.NE.0)CALL QAABORT('(CUSP_MAIN) - Closing input '//
$                  'CAMDAT ***')
 IF(IERR.NE.0)CALL QAABORT('(CUSP_MAIN) - Trouble closing'//
$                  ' output cdb-scratch & CAMDAT files, etc... ***')
```

SUBROUTINE MEMADJ(MORMEM, MNUSED, Q)

```
   CALL QAABORT('(MEMADJ) - Invalid REAL dynamic array '//
+          'indices detected ***')
   CALL QAABORT('(MEMADJ) - REAL DYNAMIC MEMORY '//
+          'ALLOCATION ERRORS ***')
   CALL QAABORT('(MEMADJ2) - Invalid REAL dynamic array '//
+          'indices detected ***')
   CALL QAABORT('(MEMADJ2) - REAL '//
$           'DYNAMIC MEMORY ALLOCATION ERRORS ***')
   CALL QAABORT('(MEMADJ3) - REAL '//
$           'DYNAMIC MEMORY ALLOCATION ERRORS ***')
```

SUBROUTINE MEMADJC(MORMEMC, MNUSED, C)

```
      CALL QAABORT('(MEMADJ) - Invalid CHARACTER dynamic array '//
     +          'indices detected ***')
      CALL QAABORT('(MEMADJC) - CHARACTER DYNAMIC MEMORY '//
     +          'ALLOCATION ERRORS ***')
      CALL QAABORT('(MEMADJC2) - Invalid CHARACTER dynamic array '//
     +          'indices detected ***')
      CALL QAABORT('(MEMADJC2) - CHARACTER '//
     $          'DYNAMIC MEMORY ALLOCATION ERRORS ***')
      CALL QAABORT('(MEMADJC3) - ### CHARACTER ### '//
     $          'DYNAMIC MEMORY ALLOCATION ERRORS ***')
```

SUBROUTINE PARBIN0

```
      CALL QAABORT(' (PARBIN0) - PARBIN0 error input parameters ***')
```

SUBROUTINE PARTXT0

```
      IF (IERR .GT. 0) CALL QAABORT(
     $                    ' (PARTXT0) - PARTXT0 error input parameters ***')
```

SUBROUTINE PREPRO(FIDB, FTXT0, FBIN0, FTXT1, FODB, FBRG)

```
      IF(IERR.NE.0)CALL QAABORT('(CUSP_MAIN) - opening PRE_CUTTINGS'//
     $                    ' input template file ***')
      IF(IERR.NE.0)CALL QAABORT('(CUSP_MAIN) - opening PRE_CUTTINGS'//
     $                    ' input desc_0500 file ***')
      IF(IERR.NE.0)CALL QAABORT('(CUSP_MAIN) - opening PRE_CUTTINGS'//
     $                    ' input text file 1 ***')
      IF(IERR.NE.0)CALL QAABORT('(CUSP_MAIN) - opening CUTTINGS'//
     $                    ' output ascii SDB file ***')
      IF(IERR.NE.0)CALL QAABORT('(CUSP_MAIN) - opening CUTTINGS'//
     $                    ' output binary SDB file ***')
      IF(IERR.NE.0)CALL QAABORT('(CUSP_MAIN) - opening CUTTINGS'//
     $                    ' input text file 0 ***')
      IF(IERR.NE.0)CALL QAABORT('(CUSP_MAIN) - opening CUTTINGS'//
     $                    ' input text file 1 ***')
      IF(IERR.NE.0)CALL QAABORT('(CUSP_MAIN) - opening CUTTINGS'//
     $                    ' output verify file ***')
      IF(IERR.NE.0)CALL QAABORT('(CUSP_MAIN) - opening CUTTINGS'//
     $                    ' output plot file ***')
```

```
------------------------------------------------------------------
      SUBROUTINE PROCDB( MAXC, MAXQ, IQ, LQ, Q, C )

        IF(IERR.NE.0)CALL QAABORT('(PROCDB) - reading input CAMDAT '//
     $                    'sizing parameters ***')
        IF(IERR.NE.0)CALL QAABORT('(PROCDB) - reading input CAMDAT '//
     $                    'TITLE ***')
        IF(IERR.NE.0)CALL QAABORT('(PROCDB) - reading number of input'//
     $                    ' CAMDAT analysis variables ***')

------------------------------------------------------------------
      SUBROUTINE PROSDB( MAXC, MAXQ, IQ, LQ, Q, C , IWASFLG, QAINFO )

        IF (IABORT .NE. 0)
     &  CALL QAABORT('(CUSP_PROSDB) - finding data in the INGRES SDB'//
     &               ' database ***')

------------------------------------------------------------------
      REAL*4 FUNCTION QUANKG( DT,    LPOINT, NRILC,  NLEFT,  RADCHN,
     $                  NGRAD,  IDMTRL, HALFY,  AWT,    XMOLES )

        CALL QAABORT('(QUANKG) - cannot find all radioisotopes in '//
     $               'local chain in GRDB rad. names array ***')

------------------------------------------------------------------
      SUBROUTINE REABIN0( NNIC,   RADCHN, IDMTRL, NUCNAM,
     $                MAPSAV, SAVNAM, HALFS,  HALFY,  AWT,
     $                ACTCNV, EPAREL,CIPMSQ, MAPINV, IWASFLG )

      CALL QAABORT(' (PARBIN0) - PARBIN0 error input parameters ***')

------------------------------------------------------------------
      SUBROUTINE REASDB (DBNAME, CALC_NAME, IDMTRL, IDPRAM, MEDIAN,
     &                IABORT)

      CALL QAABORT('(CUSP_PROSDB) - extracting '//
     $           ' idmtrl, idpram ***')

------------------------------------------------------------------
      SUBROUTINE REATXT0( NNIC,   RADCHN, IDMTRL, NUCNAM,
     $                MAPSAV, SAVNAM, HALFS,  HALFY,  AWT,
     $                ACTCNV, EPAREL, CIPMSQ, MAPINV, IWASFLG )

        IF (ICHECK .GT. 11111)
     &    CALL QAABORT('(REATXT0) - read errors radionuclide input***')
```

```
      IF(IERR.GT.0)CALL QAABORT('(REATXT0) - read errors ***')
            CALL QAABORT('(REATXT0) - Repeating/Duplicate radio'//
     $                  'isotope names in chain ***')
        CALL QAABORT('(REATXT0) - Illegal names in "SAVE" '//
     $              'radionuclides list %%%')

------------------------------------------------------------------------
      SUBROUTINE REATXT1( IASPRP,  NAMELB,  NMATPR, XMATPR, TINT,      1
     &                    DIAMMOD, TUNITS,  PORO,   TCLOUT, PRESSR,
     &                    PARTDIA, SATGASX, CEMSIG, FGE,    FSE,
     &                    FCE )

          CALL QAABORT('(REATXT1) - ILLEGAL drilling bit diamet'//
     &                 'er specified (<=0) ***')
      IF(IERR.GT.0)CALL QAABORT('(REATXT1) - read errors ***')

------------------------------------------------------------------------
      SUBROUTINE RECHEA( Q,     C,     IDEBLK, NUMPRP, IASPRP,       1
     $                   XMATPR, QAINFO, NAMELB, NMATPR )

       IF(IERR.NE.0)
     $  CALL QAABORT('(RECHEA) - reading material ID and names array ***')
       IF(IERR.NE.0)
     $  CALL QAABORT('(RECHEA) - reading PROPERTY NAMES array ***')
       IF(IERR.NE.0)
     $  CALL QAABORT('(RECHEA) - reading PROPERTY values ***')
       IF(IERR.NE.0)CALL QAABORT('(RECHEA) - reading QA array ***')
       IF(IERR.NE.0)CALL QAABORT('(RECHEA) - obtaining input CAMDAT '//
     $                 'no. of time steps ***')

------------------------------------------------------------------------
      SUBROUTINE RGRDB( IDMTRL, HALFS, HALFY, AWT, ACTCNV, EPANORM )  1

        CALL QAABORT('(RGRDB) - read errors ***')
        CALL QAABORT('(RGDB) - counting entries in GRDB ***')

------------------------------------------------------------------------
      SUBROUTINE SCASDB( FSDB,  NENTRY, ID1,   ID2,   ID3,           1
     $                   ID4,   ID5,    ID6,   ID7,   IDENT )

        CALL QAABORT('(SCASDB) - Bad SDB initialization *****')
        CALL QAABORT('(SCASDB) - Error calling SDBREAD_LIB routine'//
     $              ': SDBCASES ***')
        CALL QAABORT('(SCASDB) - Error calling SDBREAD_LIB routine'//
     $              ': SDBINFO ***')
```

```
      CALL QAABORT('(SCASDB) - Error calling SDBREAD_LIB routine'//
     $           ': SDBNINFO ***')
      CALL QAABORT('(SCASDB) - keyword IDMTRL missing on SDB '//
     $           'header description ***')
      CALL QAABORT('(SCASDB) - keyword IDPRAM missing in SDB '//
     $           'header description ***')
      CALL QAABORT('(SCASDB) - keyword IDSCAL missing in SDB '//
     $           'header description ***')
      CALL QAABORT('(SCASDB) - keyword MEDIAN missing in SDB '//
     $           'header description ***')
      CALL QAABORT('(SCASDB) - keyword LOWRNG missing in SDB '//
     $           'header description ***')
      CALL QAABORT('(SCASDB) - keyword HIRNGE missing in SDB '//
     $           'header description ***')
      CALL QAABORT('(SCASDB) - keyword UNITS missing in SDB '//
     $           'header description ***')

------------------------------------------------------------------
      SUBROUTINE SCATXT0( IWASFLG, NAMELB, KOMAT,  OUTMAT, IDMTRL,   1
     $                    QAINFO,  ISETUP )

      IF(N.NE.NGRAD)CALL QAABORT(
     $' (SCATXT0) - SCATXT0 input text for generic sizing ***')
      IF(IERR.GT.0)CALL QAABORT(
     $' (SCATXT0) - SCATXT0 input text for sizing parameters ***')

------------------------------------------------------------------
      SUBROUTINE SCATXT1( IWASFLG, IASPRP, XMATPR, NAMELB, NMATPR,   1
     &                    IDMTRL,  OUTMAT )

        IF (MULTP .NE. JULTP) CALL QAABORT(
     & ' (SCATXT1) - SCATXT1 MULTP gt NULTP ***')
        IF (MULTP .GT. NULTP) CALL QAABORT(
     & ' (SCATXT1) - SCATXT1 order of multiple hits wrong ***')
        IF (INUM .GT. NRMS) CALL QAABORT(
     &    ' (SCATXT1) - SCATXT1 INUM gt NRMS ***')
999   IF(IERR.GT.0)CALL QAABORT(
     &' (SCATXT1) - SCATXT1 input text for sizing parameters ***')

------------------------------------------------------------------
      SUBROUTINE SGRDB                                          1

          CALL QAABORT('(SGRDB) - Illegal GRDB format ***')
666   IF(IERR.GT.0)CALL QAABORT('(SGRDB) - read errors ***')
------------------------------------------------------------------
```

```
      SUBROUTINE WRCHEA( IWASFLG, Q,    CH,    IASPRP, NMATPR,    1
     $           QAINFO,  LDUMP,  RDUMP,  KOMAT,  NUCNAM,
     $           SAVNAM,  MAPSAV )

        IF(IERR.NE.0)CALL QAABORT(
     $  '(WRCHEA) - ADDING "DENSITY" as a CAMDAT property ***')
        IF(IERR.NE.0)CALL QAABORT(
     $  '(WRCHEA) - ADDING "DOMEGA" as a CAMDAT property ***')
        IF(IERR.NE.0)CALL QAABORT(
     $  '(WRCHEA) - ADDING "ABSRO" as a CAMDAT property ***')
        IF(IERR.NE.0)CALL QAABORT(
     $  '(WRCHEA) - ADDING "TAUFAIL" as a CAMDAT property ***')
        IF(IERR.NE.0)CALL QAABORT(
     $  '(WRCHEA) - ADDING "VISCO" as a CAMDAT property ***')
        IF(IERR.NE.0)CALL QAABORT(
     $  '(WRCHEA) - ADDING "YLDSTRSS" as a CAMDAT property ***')
        IF(IERR.NE.0)CALL QAABORT('(WRCHEA) - Trouble writing '//
     $'CUTTINGS SDB QA record to CDB-scratch file ***')
        IF(IERR.NE.0)CALL QAABORT('(WRCHEA) - Trouble writing '//
     $'CUTTINGS QA record to CDB-scratch file ***')
        IF(IERR.NE.0)CALL QAABORT('(WRCHEA) - Writing "HEADER" '//
     $'section to the final output CAMDAT file ***')
        IF(IERR.NE.0)CALL QAABORT('(WRCHEA) - Error writing'//
     $  ' NEW GLOBAL variable names (AREA_C) to CDB ***')
        IF(IERR.NE.0)CALL QAABORT('(WRCHEA) - Error writing'//
     $  ' NEW GLOBAL variable names (AREA_S) to CDB ***')
        IF(IERR.NE.0)CALL QAABORT('(WRCHEA) - Error writing'//
     $  ' NEW GLOBAL variable names (AREA_T) to CDB ***')
        IF(IERR.NE.0)CALL QAABORT('(WRCHEA) - Error writing'//
     $  ' NEW GLOBAL variable names (VOL_C) to CDB ***')
        IF(IERR.NE.0)CALL QAABORT('(WRCHEA) - Error writing'//
     $  ' NEW GLOBAL variable names (VOL_S) to CDB ***')
        IF(IERR.NE.0)CALL QAABORT('(WRCHEA) - Error writing'//
     $  ' NEW GLOBAL variable names (VOL_T) to CDB ***')
        IF(IERR.NE.0)CALL QAABORT('(WRCHEA) - Error writing'//
     $  ' NEW GLOBAL variable names (DRILDIAM) to CDB ***')
        IF(IERR.NE.0)CALL QAABORT('(WRCHEA) - Error writing'//
     $    ' NEW GLOBAL variable names (NUM_INTR) to CDB ***')
        IF(IERR.NE.0)CALL QAABORT('(WRCHEA) - Error writing'//
     $    ' NEW GLOBAL variable names (POROSITY) to CDB ***')
        IF(IERR.NE.0)CALL QAABORT('(WRCHEA) - Error writing'//
     $    ' NEW GLOBAL variable names (HFINAL) to CDB ***')
        IF(IERR.NE.0)CALL QAABORT('(WRCHEA) - Error writing'//
     $    ' NEW GLOBAL variable names (PRESGAS) to CDB ***')
        IF(IERR.NE.0)CALL QAABORT('(WRCHEA) - Error writing'//
```

```
$    ' NEW GLOBAL variable names (PRESBRIN) to CDB ***')
     IF(IERR.NE.0)CALL QAABORT('(WRCHEA) - Error writing'//
$    ' NEW GLOBAL variable names (SATGAS) to CDB ***')
     IF(IERR.NE.0)CALL QAABORT('(WRCHEA) - Error writing'//
$    ' NEW GLOBAL variable names (SATBRIN) to CDB ***')
     IF(IERR.NE.0)CALL QAABORT('(WRCHEA) - Error writing'//
$    ' NEW GLOBAL variable names (POROSITY) to CDB ***')
     IF(IERR.NE.0)CALL QAABORT('(WRCHEA) - Error writing'//
$    ' NEW GLOBAL variable names (HFINAL) to CDB ***')
     IF(IERR.NE.0)CALL QAABORT('(WRCHEA) - Error writing'//
$    ' NEW GLOBAL variable names (PRESGAS) to CDB ***')
     IF(IERR.NE.0)CALL QAABORT('(WRCHEA) - Error writing'//
$    ' NEW GLOBAL variable names (SATBRIN) to CDB ***')
     IF(IERR.NE.0)CALL QAABORT('(WRCHEA) - Error writing'//
$' NEW GLOBAL variable names (NUCNAM C_) to CDB ***')
     IF(IERR.NE.0)CALL QAABORT('(WRCHEA) - Error writing'//
$' NEW GLOBAL variable names (NUCNAM S_) to CDB ***')
     IF(IERR.NE.0)CALL QAABORT('(WRCHEA) - Error writing'//
$' NEW GLOBAL variable names (NUCNAM T_) to CDB ***')

------------------------------------------------------------------------
      SUBROUTINE WRPLFIL( IWASFLG, MAPSAV,  HALFY,  NUCNAM, RADHIS, 1
     $                    TINT,   XRAD1,   XRAD2,  XRAD3,  TMPNM1,
     $                    TMPNM2,  TMPNM3 )

      CALL QAABORT('(WRPLFIL) - Exceeded maximum history plott'//
     $           'ing file "width" ***')

------------------------------------------------------------------------
      SUBROUTINE WRTANA( IWASFLG, Q,      C,     TIME,   HIFLAG,    1
     $                   AINTC,  DRILDIA, NUCNAM,  MAPSAV,  SUMCI,
     $                   XPORO, XPRGS, XPRBR, XSATG, KULTP, KH )

     IF(IERR.NE.0)CALL QAABORT('(WRTANA) - Error writing'//
$    ' NEW GLOBAL variable values (AINTC(1,KH)) to CDB ***')
     IF(IERR.NE.0)CALL QAABORT('(WRTANA) - Error writing'//
$    ' NEW GLOBAL variable values (AINTC(2,KH)) to CDB ***')
     IF(IERR.NE.0)CALL QAABORT('(WRTANA) - Error writing'//
$    ' NEW GLOBAL variable values (AINTC(3,KH)) to CDB ***')
     IF(IERR.NE.0)CALL QAABORT('(WRTANA) - Error writing'//
$    ' NEW GLOBAL variable values (VOL_C) to CDB ***')
     IF(IERR.NE.0)CALL QAABORT('(WRTANA) - Error writing'//
$    ' NEW GLOBAL variable values (VOL_S) to CDB ***')
     IF(IERR.NE.0)CALL QAABORT('(WRTANA) - Error writing'//
$    ' NEW GLOBAL variable values (VOL_T) to CDB ***')
```

```
      IF(IERR.NE.0)CALL QAABORT('(WRTANA) - Error writing'//
     $  ' NEW GLOBAL variable values (DRILDIA) to CDB ***')
      IF(IERR.NE.0)CALL QAABORT('(WRTANA) - Error writing'//
     $    ' NEW GLOBAL variable values (NUM_INTR) to CDB ***')
      IF(IERR.NE.0)CALL QAABORT('(WRTANA) - Error writing'//
     $    ' NEW GLOBAL variable values (POROF) to CDB ***')
      IF(IERR.NE.0)CALL QAABORT('(WRTANA) - Error writing'//
     $    ' NEW GLOBAL variable values (HFINAL) to CDB ***')
      IF(IERR.NE.0)CALL QAABORT('(WRTANA) - Error writing'//
     $    ' NEW GLOBAL variable values (PRESGAS) to CDB ***')
      IF(IERR.NE.0)CALL QAABORT('(WRTANA) - Error writing'//
     $    ' NEW GLOBAL variable values (PRESBRIN) to CDB ***')
      IF(IERR.NE.0)CALL QAABORT('(WRTANA) - Error writing'//
     $    ' NEW GLOBAL variable values (SATGAS) to CDB ***')
      IF(IERR.NE.0)CALL QAABORT('(WRTANA) - Error writing'//
     $    ' NEW GLOBAL variable values (SATBRIN) to CDB ***')
      IF(IERR.NE.0)CALL QAABORT('(WRTANA) - Error writing'//
     $    ' NEW GLOBAL variable values (POROF) to CDB ***')
      IF(IERR.NE.0)CALL QAABORT('(WRTANA) - Error writing'//
     $    ' NEW GLOBAL variable values (HFINAL) to CDB ***')
      IF(IERR.NE.0)CALL QAABORT('(WRTANA) - Error writing'//
     $    ' NEW GLOBAL variable values (PRESGAS) to CDB ***')
      IF(IERR.NE.0)CALL QAABORT('(WRTANA) - Error writing'//
     $    ' NEW GLOBAL variable values (SATBRIN) to CDB ***')
      IF(IERR.NE.0)CALL QAABORT('(WRTANA) - Error writing'//
     $  ' NEW GLOBAL variable values (SUMCI) to CDB ***')
      IF(IERR.NE.0)CALL QAABORT('(WRTANA) - Error writing a completed'//
     $' time-step and advancing to next step /DBOSTEP ***')

------------------------------------------------------------------------

      INTEGER FUNCTION INEXT(INDEX)

      WRITE (6,'(" INCORRECT WORD TYPE WAS FOUND, INEXT")')
      WRITE (6,'(" WORD CMD ",A,
     &          " INTEGER  ",I20,
     &          " RNUMBER  ",E16.8)') C18, NUM1, RNUM1
      STOP ' ERROR INEXT'



------------------------------------------------------------------------

      REAL*4 FUNCTION RNEXT(INDEX)
```

```
      WRITE (6,'(" INCORRECT WORD TYPE WAS FOUND, RNEXT")')
      WRITE (6,'(" WORD CMD ",A,
     &          " INTEGER  ",I20,
     &          " RNUMBER  ",E16.8)') C18, NUM1, RNUM1
      STOP ' ERROR RNEXT'



------------------------------------------------------------------------

      SUBROUTINE ANEXT (INDEX, AVALUE, NP)

        WRITE (6,100)
100     FORMAT(" NONE REAL NUMBER WAS FOUND, ANEXT")
        STOP ' ERROR ANEXT'

      WRITE (6,130)
130   FORMAT(" NO END OF ARRAY, ANEXT")
      STOP ' ERROR ANEXT'


------------------------------------------------------------------------

      SUBROUTINE SOLID_OUT (DIA, SATBRIN, HF, PG, PARTDIA, PORO,

C
      IF (PG .LT. PSUF) THEN
        WRITE(NOUTFL,
     &     '(/" **** REPOSITORY PRESSURE LESS THAN ATMOSPHERIC")')
        STOP
      ENDIF
C
```

9.0 DESCRIPTION OF CUTTINGS_S'S OUTPUT FILES

CUTTINGS_S's output files are listed above in Section 6.2, together with their identification numbers and the logical symbols CUTTINGS_S associates with them. The output files are file numbers 7 through 13. Each of them is briefly described below. Of the required files, the only user-readable output appears in the debug file, an annotated example of which is given in Appendix D.

File 7. CUSP_OUT$CDB is the principal CUTTINGS_S output CDB file. It contains the calculated total volume (in terms of an areal footprint) of all radioactive materials, isotope by isotope, brought to the surface by the combined actions of cuttings, cavings, and spallings. Being a CDB file, it is binary and therefore not directly human readable.

File 8. CUSP_OUT$DBG is an ASCII output file that echoes the various input files, the input parameter values, and includes a summary of the output calculations given in file 7.

NOTE: There are three run-time debug flags that cause changes in the amount of output written to the DBG output file. They are defined as follows:

DEBUG = <LOGICAL> for debugging dynamic memory allocation.

DBRAG = <LOGICAL> for debugging the BRAGFLO CDB file

NOECHO = <LOGICAL> to cancel echoing of input data to the DBG file

These run-time flags may be invoked and set by typing the variable, for example: DBSPALL on the command line. Alternatively, the user may type the line: $DEFINE CUSP$DPSPALL TRUE prior to the execution of CUTTINGS_S.

File 9. CUSP_OUT$NVERIFY is an optional ASCII output file of intermediate results that is used during developmental phases to verify CUTTINGS_S's calculations against other independent calculations, normally performed by hand.

Files 10 through 11 are optional output files that are normally forwarded to system plotting packages for diagnostic and developmental purposes. See the summary table above for a listing of their logical symbols. Their use in regulatory calculations is not contemplated. Hence, these logicals will either remain undefined or will be defined as NULL or CANCEL, in which case the code will not write to them.

10.0 REFERENCES

Bateman, H. (1910) The Solution of a System of Differential Equations Occurring in the Theory of Radio-active Transformations, *Proc. Cambridge Phil. Soc.* **16**, 423.

Berglund, J. W. (1993) Mechanisms Governing the Direct Removal of Wastes from the WIPP repository Caused by Exploratory Drilling, SAND 92-7295. Sandia National Laboratories, Albuquerque, NM.

Rechard, R. P., ed. (1992). User's Reference Manual for CAMCON: Compliance Assessment Methodology Controller; Version 3.0. SAND90 - 1983. Sandia National Laboratories, Albuquerque NM.

Rechard, R. P. , A. P. Gilkey, H. J. Iuzzolino, D. K. Rudeen, and K. A. Byle. (1993). *Programmer's Manual for CAMCON: Compliance Assessment Methodology Controller.* SAND90 - 1984. Sandia National Laboratories, Albuquerque NM.

WIPP PA Department (1992). Preliminary Performance Assessment for the Waste Isolation Pilot Plant, December 1992. Volume 1: Third Comparison with 40 CFR 191, Subpart B. SAND92 - 0700/1. Sandia National Laboratories, Albuquerque NM.

WIPP PA Department (1992). Preliminary Performance Assessment for the Waste Isolation Pilot Plant, December 1992. Volume 2: Technical Basis. SAND92 - 0700/2. Sandia National Laboratories, Albuquerque NM

WIPP PA Department (1992). Preliminary Performance Assessment for the Waste Isolation Pilot Plant, December 1992. Volume 3: Model Parameters. SAND92 - 0700/3. Sandia National Laboratories, Albuquerque NM.

WIPP PA Department (1992). Preliminary Performance Assessment for the Waste Isolation Pilot Plant, December 1992. Volume 4: Uncertainty and Sensitivity Analyses for 40 CFR 191, Subpart B. SAND92 - 0700/4. Sandia National Laboratories, Albuquerque NM.

WIPP PA Department (1992). Preliminary Performance Assessment for the Waste Isolation Pilot Plant, December 1992. Volume 5: Uncertainty and Sensitivity Analyses of Gas and Brine Migration for Undisturbed Performance. SAND92 - 0700/5. Sandia National Laboratories, Albuquerque NM.