EGG-NE-10137
December 1991

Received by OSTI
APR 27 1992

INFORMAL REPORT

VENTURE/PC MANUAL
A MULTIDIMENSIONAL MULTIGROUP NEUTRON DIFFUSION CODE SYSTEM
VERSION 3

A. Shapiro
H. C. Huria
K. W. Cho

Idaho National Engineering Laboratory

Managed by the U.S. Department of Energy

EG&G Idaho

Work performed under DOE Contract No. DE-AC07-76ID01570

DISTRIBUTION OF THIS DOCUMENT IS UNLIMITED

EGG-NE--10137

DE92 012361

VENTURE/PC MANUAL

A MULTIDIMENSIONAL MULTIGROUP NEUTRON DIFFUSION CODE SYSTEM

VERSION 3

A. SHAPIRO, H. C. HURIA, K. W. CHO

DECEMBER 1991

UNIVERSITY OF CINCINNATI
NUCLEAR ENGINEERING PROGRAM
CINCINNATI, OHIO 45221

Prepared for EG&G Idaho, Inc.
Under Subcontract No. C-87-101212
and the U.S. Department of Energy
Under Contract No. DE-AC07-76ID01570
Sponsored by DOE Offices of Energy Research and Nuclear Energy

DISCLAIMER

This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency thereof, nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.

MASTER

DISTRIBUTION OF THIS DOCUMENT IS UNLIMITED

Table of Contents

Abstract

VENTURE/PC is a recompilation of part of the Oak Ridge BOLD VENTURE code system, which will operate on an IBM PC or compatible computer. Neutron diffusion theory solutions are obtained for multidimensional, multigroup problems. This manual contains information associated with operating the code system. The purpose of the various modules used in the code system, and the input for these modules are discussed. The PC code structure is also given.

Version 2 included several enhancements not given in the original version of the code. In particular, flux iterations can be done in core rather than by reading and writing to disk, for problems which allow sufficient memory for such in-core iterations. This speeds up the iteration process.

Version 3 does not include any of the special processors used in the previous versions. These special processors utilized formatted input for various elements of the code system. All such input data is now entered through the Input Processor, which produces standard interface files for the various modules in the code system. In addition, a Standard Interface File Handbook is included in the documentation which is distributed with the code, to assist in developing the input for the Input Processor.

ACKNOWLEDGEMENT

This work is a recompilation of codes developed at Oak Ridge over a period of several years. In particular, the excellent work and reports of D.R. Vondy, T.B. Fowler, and G.W. Cunningham, who developed the original code at Oak Ridge, is hereby acknowledged and credited. Also, the work of R.D. Odell and others at Los Alamos resulting in a generalized Input Processor is acknowledged. This manual is a reorganization of reports received from the Radiation Shielding Information Center on the BOLD VENTURE code system.

We also acknowledge the Reactor Physics group at INEL, in particular, Dr. D.W. Nigg, for their financial support, and for beginning the task of developing reactor physics and shielding codes for the microcomputer.

List of Figures

VENTURE/PC Code Abstract

1. **Program Identification** VENTURE/PC is a PC version of that part of the BOLD VENTURE code system developed at Oak Ridge for IBM mainframes, which includes the Control Module, an Input Processor, a Cross-Section Processor, the VENTURE neutronics code, and an Exposure Module which utilizes the BURNER code for depletion calculations.

2. **Function** The VENTURE code solves the usual neutronics eigenvalue, adjoint, fixed source, and criticality search problems. It treats up to three dimensions, maps power density, and does first order perturbation analysis at the macroscopic cross section level. The Burner code solves the nuclide chain equations to estimate the nuclide concentrations at the end of an exposure time, and also after a shutdown period.

3. **Method of Solution** The VENTURE module applies the Finite-Difference Diffusion or a simple P1 Approximation. VENTURE uses an outer-inner iteration scheme with several different data handling methods. Overrelaxation is applied to the inner and outer iterations, and succeeding flux iterates may be accelerated with the Chebychev process. The BURNER code uses a difference formulation based on average generation rates, or a matrix exponential formulation to approximate the solution of the coupled burn-up differential equations, or an explicit solution for simply coupled nuclide chains. Space dependence is included by working with zone averaged fluxes.

4. **Related Material** A Control Module, Input Processor, and a Cross Section Processor interface with input files to produce standard interface files for use by VENTURE and BURNER. Standard interface files are binary sequential files which follow a prescribed or standardized format.

5. **Restrictions** The code is variably dimensioned, but the data arrays are limited to 36000 words, or 144000 bytes, to work within the 640K memory limit of the present DOS operating system.

6. **Computer** The code will work on IBM or IBM compatible microcomputers working under the DOS operating system.

7. **Running Times** Running times are variable, and very problem and machine dependent. Many two or one dimensional problems should complete within 30 minutes on an original IBM PC, and in less than half that time on AT type machines. Three dimensional problems should probably be reserved for AT or higher class machines.

8. **Programming Language** FORTRAN 66 or 77. The program was originally written in ASA 1966 FORTRAN, but was compiled for the microcomputer with FORTRAN 77.

9. **Operating System** The program was compiled under DOS 3.1, and should run with earlier versions of DOS.

10. **Machine Requirements** The program requires about 5 megabytes of disk storage, to hold the executable files and files generated by the code. It also requires 640K of memory, and a math co-processor.

11. **Authors** A. Shapiro, H.C. Huria and K.W. Cho
Nuclear Engineering
Mail Location 72
University of Cincinnati
Cincinnati, Ohio 45221
(513) 556-2014

12. **References** "BOLD VENTURE IV, A Reactor Analysis Code System, Version IV"
Radiation Shielding Information Center
Oak Ridge National Laboratory
Post Office Box X
Oak Ridge, Tenn. 37831
CCC-459 (Computer Code Collection)

13. **Materials Available** VENTURE/PC Manual, Standard Interface File handbook, VIP (Venture Interactive (Input) Processor), VENTURE Code System, executable files and source decks on 1.2 MB 5 1/4 inch diskettes.

GETTING STARTED

VENTURE/PC is a multidimensional, multigroup, neutron diffusion code system, with provisions for processing cross sections, and for calculating burnup. VENTURE/PC is a PC compilation of part of the BOLD VENTURE code system, developed over several years at Oak Ridge National Laboratory. The conversion to the microcomputer was done as part of the INEL program for establishing a PC based reactor physics package.

VENTURE/PC is a very large code system, with many input options, which makes the code complicated. The executable code requires almost 3 million bytes for storage, and requires almost all of the maximum DOS memory of 640K to run. The code was linked with overlays, with the linked executable file being about 522K, but file buffers, and the operating system take up most of the remaining memory within the 640K DOS limit. In addition, the code comes with an interactive input processor, VIP for VENTURE Interactive Processor, which requires about 900,000 bytes for storage. VIP was also linked with an overlay structure to allow it to run with DOS. Additional disk space must be made available for files. Thus, to run VENTURE with the interactive input processor, a subdirectory of about 5 MB will be required. The VENTURE/PC code system was compiled under DOS 3.1 with the Lahey FORTRAN-77 compiler, version 2.22. Overlay linking was accomplished through the Phoenix PLINK86 Plus overlay linker, version 2.24.

To run VENTURE/PC , the CONFIG.SYS file on the hard disk root directory should be set for FILES = 50 and BUFFERS = 10. In addition, to run VIP the ANSI.SYS file from DOS should be placed on the root directory, and the line DEVICE=ANSI.SYS should be added to the CONFIG.SYS file. The system should be booted with these configuration specifications.

Prior to running any problems, the input file for the problem should be copied to the file VENTURE.INP. The output will be written to the file VENTURE.OUT. Several sample problems are given with the code. They all have names with the .INP extension. They can be copied to the VENTURE subdirectory for checking the VENTURE operation.

All standard interface files generated during a run are maintained by name. Text versions of standard interface files are retained on option in the file named STFILE.TXT, which may be edited with any good PC editor.

The file F77L.EER is now included with the code package. This file should be placed on the VENTURE subdirectory. It will provide statements associated with FORTRAN errors, such as improperly formatted input. It is part of the Lahey FORTRAN compiler package.

PART I. DESCRIPTION OF CODE SYSTEM

1. Introduction

VENTURE/PC is an IBM-PC or compatible microcomputer version of the BOLD VENTURE [1] system of connected codes or modules used to analyze the core of a nuclear reactor by applying multigroup diffusion theory. The code system can analyze 1, 2, or 3 dimensions in various geometries. Variable dimensioning is used throughout the codes, which allows for any number of energy groups and mesh points, with the limitation that the problem fit into core memory. Upscattering as well as downscattering is accommodated by the codes. A depletion module is included for burnup calculations.

An important feature of this code system is that each code module receives input from, and writes output to, standard interface files. Standard Interface Files (SIF's) are unformatted binary sequential files which have been specified as to format and structure by the Committee on Computer Code Coordination [2], or CCCC. An Input Processor [3] reads standard interface card image (or ASCII) format, and converts the input to standard interface files for use by the code modules. Version 3 of VENTURE/PC differs from previous versions in that all special processors have been removed from the code system. These special processors were designed to read formatted input developed for the various code modules prior to CCCC standardization. All input for the code modules in version 3 is accomplished through the Input Processor.

A Standard Interface File Handbook which accompanies the documentation should help significantly in developing the input.

2. VENTURE/PC Code System

The structure of version 3 of VENTURE/PC is shown in Figure 1. It includes a Driver, a Control module, an Input Processor, a Cross Section Processor, and the two main calculational modules VENTURE and EXPOSURE.

The Driver reads the input data on the file VENTURE.INP. All input data must reside on the file VENTURE.INP prior to making a run. To store multiple input data sets on the same disk, each data set should have its own unique name. If the data set is to be used to make a run with VENTURE/PC, it should be renamed or copied to VENTURE.INP. The Driver first calls the Control Module which initializes a file catalog and a control file. All interface files are cataloged as to name, unit number and version number, and must be recorded on the catalog file on unit 09. The standard interface file CONTRL, which is initialized by the Control Module, contains records of control information required by the modules for selecting various options or calculational pathways. It is written on unit number 10. Upon return from the Control Module, the Driver calls the other modules in an order specified by the Control Module.

The primary function of the Cross Section Processor is to convert isotope ordered cross sections in an ISOTXS file, to group ordered cross sections in a GRUPXS file, as required by VENTURE.

The VENTURE neutronics module calculates the neutronics of a problem, while the EXPOSURE module solves the isotopic rate equations for number density variations associated with fuel burnup and fission product buildup.

The numbers associated with the modules on Figure 1 are the input identification numbers for these modules.

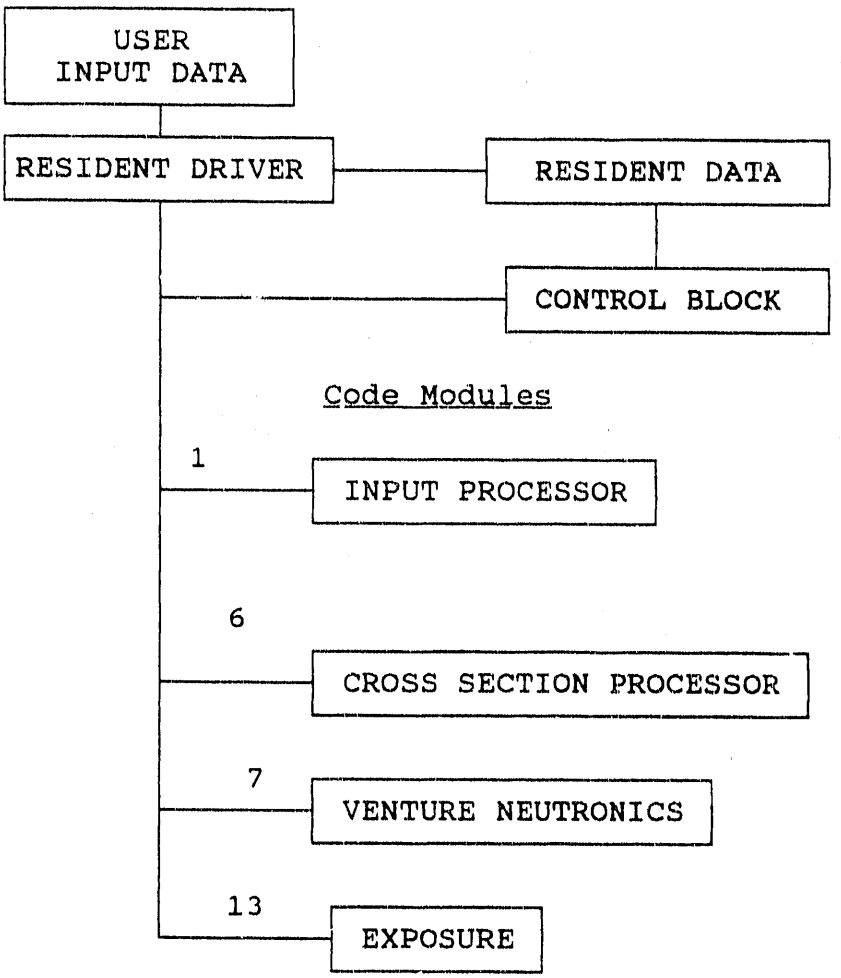


Figure 1. Components of Computation System
(From Reference 1)

3. Driver.

The Driver used for VENTURE/PC was established from the partial FORTRAN version of the Los Alamos Driver given in reference [4]. The detailed FORTRAN listing of the Driver developed for this work is given in Appendix I. The function of the Driver is to first call the Control Module, and read that block of data associated with the Control Module on the input file, VENTURE.INP. The VENTURE.INP file is assigned to unit 1. This Control Module data block is then written on unit 5, for the Control Module to access. This procedure of reading data blocks from the input file, and overwriting the data on unit 5 is done for all input sections. Upon return from the Control Module, The Driver calls the Input Processor and/or the VENTURE neutronics module as specified in the control file. The Input Processor develops all necessary standard interface files required to run the other modules. The other modules required by the problem are then called by the Driver in the sequence given in the control data block.

4. Control Module.

The primary functions of the Control Module are to initialize the file which catalogs the standard interface files on unit 9, initialize the standard interface file CONTRL on unit 10, set the problem data storage in memory, identify standard interface files to be used initially, and perform wrap up procedures when a problem prematurely aborts or when it finishes the calculation

correctly. The catalog file is referenced by the various modules to obtain the unit numbers for standard interface files required by that module. The CONTRL file contains global information records pertinent to the problem, and a specific record for each computational module. The individual records on the CONTRL file for the computational modules are developed by the Input Processor. Computational modules read the applicable control information on these records from the CONTRL file when they are accessed.

5. Input Processor.

The input Processor was developed at Los Alamos [3], and has the function of converting ASCII or BCD input into Standard Interface Files. The ASCII or BCD input is structured in the same manner as the Standard Interface Files. See the "Standard Interface File Handbook" for a more detailed discussion of Standard Interface Files and the Input Processor.

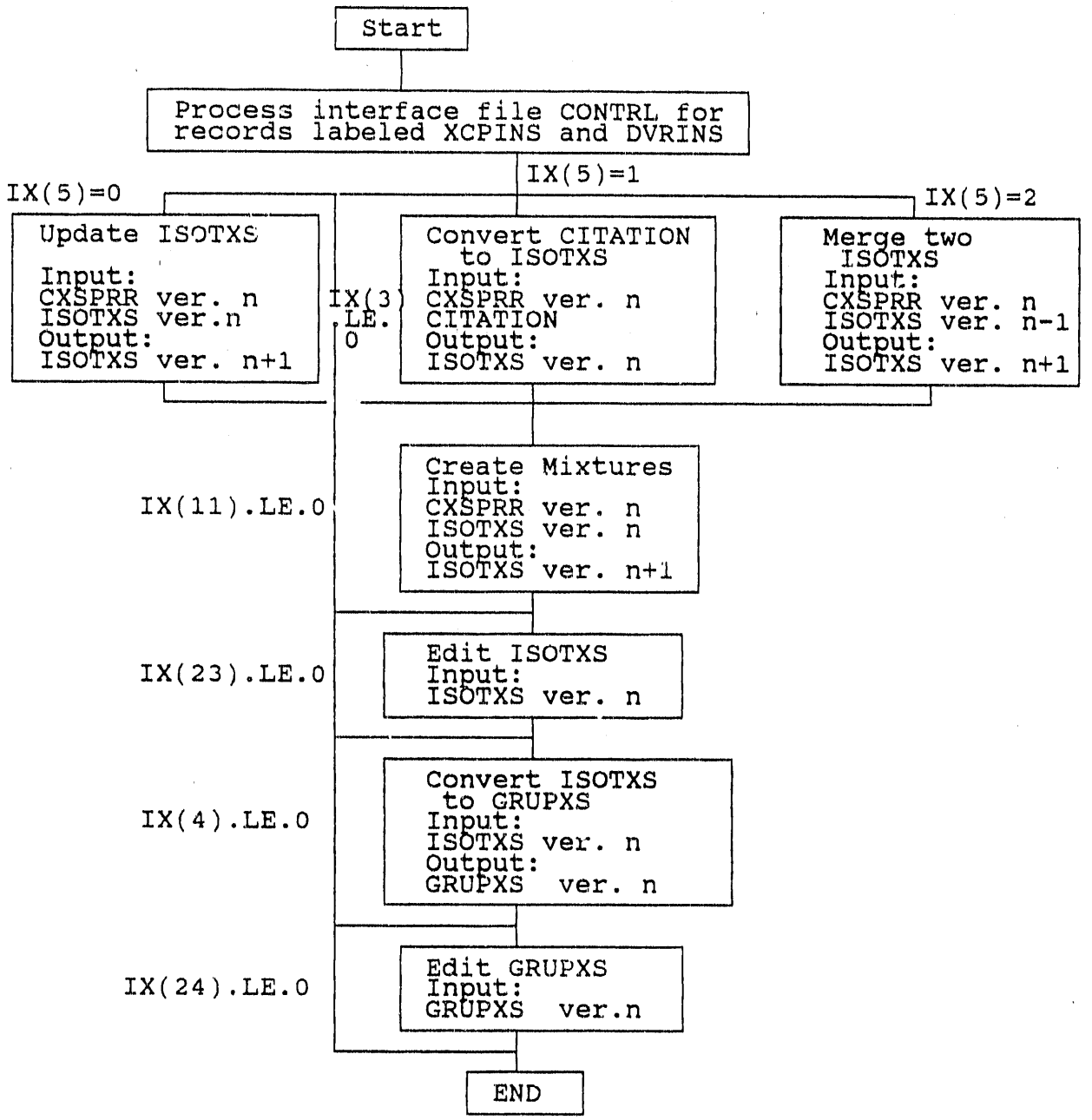
6. Cross Section Processor.

The Cross Section Processor is documented in reference [5]. A primary function of this processor is to convert a nuclide ordered ISOTXS file to a group ordered GRUPXS file required by the VENTURE neutronics module. It can also create a nuclide ordered file from the ORNL CITATION code format, update an existing nuclide ordered file, and merge two existing nuclide ordered files. The processor has the capability of creating nuclide mixtures and macroscopic cross sections in the nuclide ordered format. It can handle up to 1000 energy groups, 500 nuclides and a Legendre expansion order as high as 20. The capabilities and variations of

this processor are summarized in Figure 2. It reads the records labeled XCPINS and DVRINS on the CONTRL file to determine the path to follow.

6.1 CITATION Cross Sections.

The CITATION code [6] was the prcursor to VENTURE, and many macroscopic cross section sets are available in CITATION format. A Special Processor, DCMACR [1], converts the CITATION macroscopic cross sections to microscopic cross sections, and writes the microscopic cross sections on unit 8. The Cross Section Processor can then process the microscopic cross section set, and convert the set to a form which can be used by the VENTURE neutronics module. The code DCMACR is provided as a separate code with the VENTURE/PC version 3 package. It must be used to process CITATION cross sections before VENTURE/PC is run, and should reside on the same subdirectory as VENTURE/PC.



Note: Ver. n refers to the current latest version at that stage of processing.

Figure 2. Calculational Modes of Cross section Processor (from ref. 5)

7. VENTURE Neutronics Module.

The VENTURE module calculates the neutronics of a given problem and is the main module of the code system. Its primary documentation is reference [5], with VENTURE additions given in reference [7]. The interested reader can find the details of the VENTURE code in these references. An outline of the code will be reproduced here.

7.1 VENTURE STRUCTURE.

A flow chart for the calculational procedure is shown in Figure 3. The necessary macroscopic cross sections and equation constants must be calculated from the data given in the standard interface files used for input. After the procedures are initialized and scratch files prepared, an outer iteration loop is started. In criticality problems, the fission source and inscatter source are then calculated to provide the source term for the inner iteration. The inner iteration loop is then done for each group, with accelerated overrelaxation used to speed convergence. The k_{eff} eigenvalue is calculated and the iterated results edited. The convergence criteria are tested, and if convergence is not achieved, the outer iteration loop is repeated. If a direct search problem was specified, the formulation is constructed as an eigenvalue problem, with the eigenvalue as a multiplier on the search parameters of interest (e.g., buckling or dimensions, or nuclide concentrations). The desired k_{eff} is kept constant. This eigenvalue multiplier is evaluated upon the completion of the inner iteration for each energy group. Thus, for direct search problems,

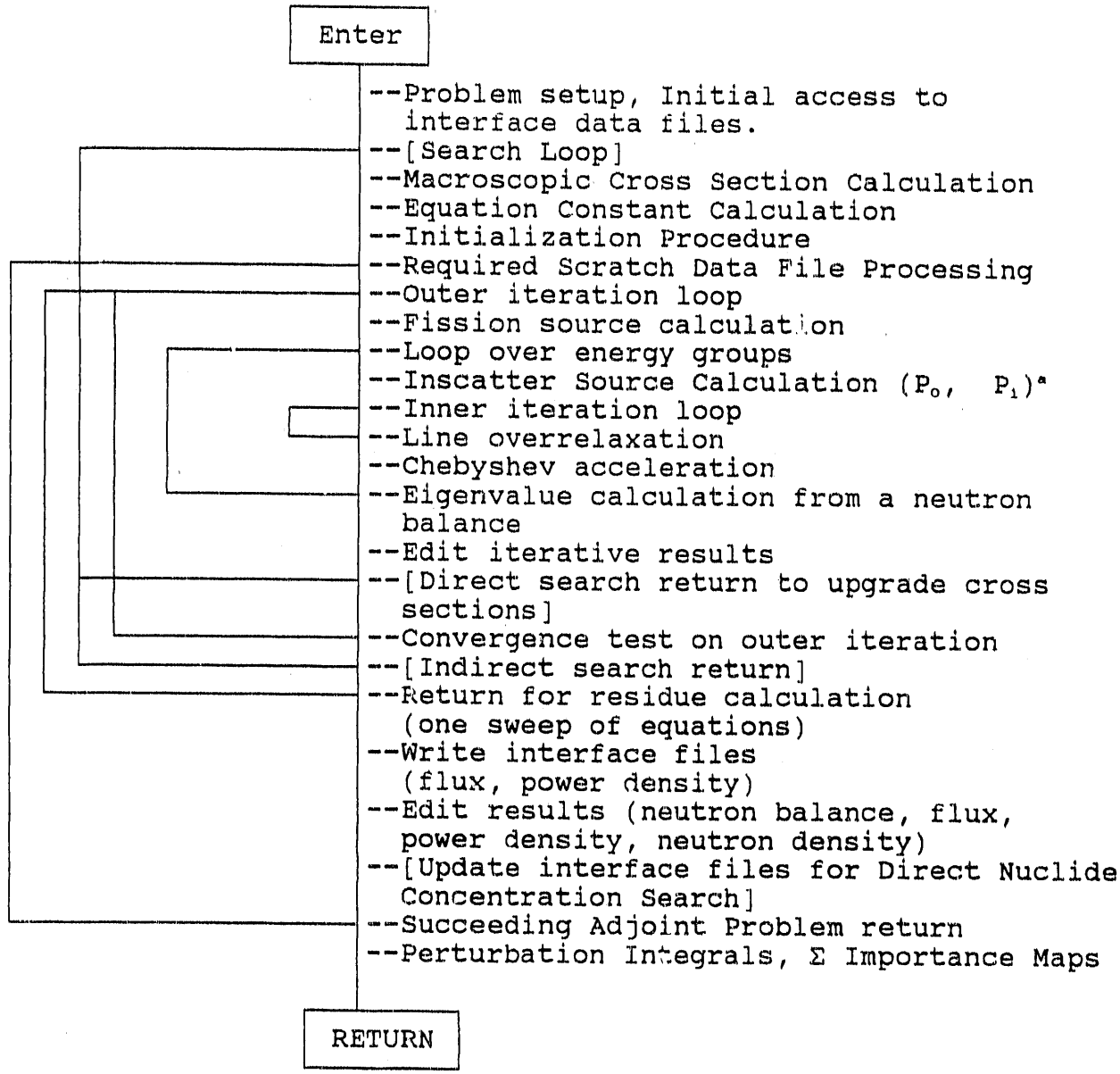
the search parameters are estimated after the inner iterations, without the necessity of an outer iteration. A loop to upgrade cross sections is made after the inner iteration, if a direct search on nuclide concentrations was specified. The indirect search loop changes cross sections or dimensions after the outer iteration, to effect a change in k_{eff}, until the desired k_{eff} is achieved. Thus, in the indirect search analysis, both the search parameters and k_{eff} are changed, and this can only be done through an outer iteration. Upon completion of the outer iteration, the interface files of flux and power density are written, and the results are edited and updated. If an adjoint problem is required it is then done, after which perturbation integrals and importance maps can be generated.

7.2 VENTURE Subroutines.

The VENTURE subroutines, as taken from reference [5], are shown in Appendix II. They are grouped together in this Appendix as to function.

7.3 VENTURE File Requirements.

The standard interface files used by VENTURE are shown in Figure 4, along with the files which can be generated by the code on option. The scratch files with their associated unit numbers are shown in Figure 5. Additional information regarding the individual file records is given in reference [5].



[a] The inscatter source calculation is normally done outside the inner iteration loop; however, in one data handling mode this source is calculated inside the inner iteration loop to minimize data transfer.

Figure 3. User Flow Chart, VENTURE Finite-Difference Diffusion Theory Neutronics Code Block, (from ref. 6)

VENTURE Interface Files

Interface Data Files Used

GRUPXS - Group ordered microscopic cross sections

GEODST - Geometry Description

NDXSRF - Nuclide to cross section referencing data

ZNATDN - Zone nuclide atomic densities

SEARCH - Search data (required only for search)

FIXSRC - Fixed source (required for a fixed source problem only)

RTFLUX - Total neutron flux (if supplied and for successive cases)

ATFLUX - Total adjoint neutron flux (if supplied)

RZFLUX - Zone average total neutron flux (for successive cases)

Interface Data Files Generated by Option

RTFLUX - Total neutron flux

ATFLUX - Total adjoint flux

RZFLUX - Zone average total neutron flux

PWDINT - Point power density

GEODST - Geometry description upon dimension search

NDXSRF - Nuclide to cross section referencing data upon dimension search

ZNATDN - Zone nuclide atomic densities upon concentration search

FIXSRC - Fixed source result

PERTUB - Regular, adjoint flux integrals

RSTRTR - Data for problem restart

FISSOR - Special fixed source data

Figure 4. VENTURE Interface files (from ref. 5)

Scratch Files by Unit Number

21		Macroscopic scattering cross sections
22		Principal macroscopic cross sections
23	(Direct Access)	Equation coupling constants in space, normally not used
24	(Direct Access)	Total neutron flux
27	(Direct Access)	Flux copy
28	(Direct Access)	Flux copy
29	(Direct Access)	Del dot J times volume (current in the P_1 sense
40	(Direct Access)	Equation constants
41		Fixed source
42		Fission source
43		Total source
44		Neutron balance data
45		Miscellaneous[b]
46		Miscellaneous
47		Search data, misc.

[b] By miscellaneous is meant that these files are generally used to store different information at different stages of a calculation, but the required storage space is usually not large relative to those for which requirements are given in detail.

Figure 5. VENTURE Scratch Files (from ref. 5)

7.4 Data Handling Modes.

There are several different methods of storing the necessary flux and coupling constant data built into the VENTURE code module. These are:

All Stored Mode. In this mode, all data necessary to complete a calculation is stored in memory. There is little if any data transfer between disk files and memory. This mode can only be used for small problems with few space-energy points.

Space Stored Mode. This is the preferred mode for problems of moderate size, which should apply to most problems which can be run by VENTURE/PC. The equation constants, flux values, and the necessary source values to complete a one group inner iteration for the flux values at each mesh point are stored in memory. To obtain source values, the flux values and scattering data must be read from disk files between iterations.

Multirow Stored Mode. This is a data handling technique for two dimensional problems. Data for only several rows of fluxes for a single group are stored in memory, thereby reducing the memory required to complete an inner iteration.

Multiplane Stored Mode. In this mode, data for several planes of a three dimensional problem are stored in memory. Storing equation constants and source values for n planes, and flux values for n+2 planes, allows inner iterations to be done for n planes, with one access of equation constants, one access of old flux values, and one transfer of new flux values for each plane of the problem.

One Row Stored Mode. For large three dimensional problems with many space points, it may be necessary to limit the data in memory to that required to complete an inner iteration for a single row. Storage is allocated for the necessary five rows of flux values and equation constants to complete the inner iteration. This method applies only to three dimensional problems, and is slow due to the many disk transfers required to complete a problem.

7.5 Units.

Dimensions are in centimeters, nuclide densities in atoms/barn-cm, microscopic cross sections in barns, and macroscopic cross sections in cm^{-1}.

7.6 Access to Microscopic Cross Sections.

Except for nuclide concentration searches, the microscopic cross sections and nuclide densities are accessed only once. All subsequent calculations use macroscopic cross sections which are calculated in the beginning.

7.7 Fatal Errors.

Fatal errors are of the following types:

Error Number 666 - error encountered in processing interface data files;

Error Number 444 - data transfer errors;

Error Number 555 - other interpreted errors and system errors.

7.8 Geometry and Boundary Conditions.

The one dimensional geometries available are the slab, infinitely high cylinder, and sphere. The two dimensional geometries include X-Y, R-Z, Theta-R, equilateral triangle, T, and equilateral hexagon, H. Three dimensional geometries include X-Y-Z, Theta-R-Z, T-Z, and H-Z. Left boundary conditions include zero flux, reflected, extrapolated, and repeating. Right boundary conditions are zero flux, reflected, extrapolated, repeating, and rotational symmetry conditions. For multidimensional problems, additional column and plane boundary conditions must be specified, the latter required for three dimensional problems.

7.9 Special Boundary Conditions.

VENTURE allows for some special boundary conditions. These include 90° and 180° rotational symmetry for slab and rectangular geometries, and 120° and 60° rotational symmetry for corresponding triangular coordinates. Rotations are keyed to the right hand or third side of these geometries. See reference [5] for additional details.

7.10 Types of Problems Solved.

VENTURE can solve the following types of neutronics problems:

1 - Usual Eigenvalue Problem.

2 - Fixed Source Problem

3 - Adjoint Flux Problem

4 - Direct Buckling Search

5 - Direct Reciprocal Velocity Search

6 - Direct Nuclide Concentration Search

7 - Indirect Searches - Concentrations and Dimensions

7.11 Iteration Procedures.

Solutions to eigenvalue and search problems are done through the usual inner-outer iteration procedures. Overrelaxation is applied to speed convergence. Chebyshev polynomials are utilized to obtain overrelaxation coefficients. The eigenvalue is estimated after each outer iteration as the ratio of the total production rate to loss rate. A very detailed discussion of the rather complicated acceleration techniques utilized in VENTURE is given in reference [5].

7.12 Perturbation Calculations.

VENTURE provides information associated with perturbation calculations. Given forward and adjoint flux solutions, the derivative of k_{eff} with respect to each zone cross section and diffusion constant are calculated and edited on option. The zone integrals of volume times the product of the flux and its adjoint are written on an interface data file for future use. Pointwise importance maps of $\nu\Sigma_f$, Σ_a, and their differences are edited. In addition, the change in k_{eff} produced by 100% changes in cross section are calculated. Also, the effect of uncertainties in cross section are generated.

7.13 The Uncommon Reactor Core Neutronics Problem.

For stability studies, the dominant higher harmonic solution is needed. A procedure has been implemented in VENTURE to remove the fundamental contribution to the iterated solution after each outer iteration, leaving the dominant higher harmonic. This can be done for both forward and adjoint distributions. See reference [7] for additional details.

7.14 Space-Energy Rebalance.

Space-energy rebalance has been incorporated as an option for accelerating outer iteration convergence. It can be applied to one dimensional slab (X), cylindrical (R), or spherical (R) geometry; two dimensional X-Y or cylindrical R-Z geometry; and three dimensional X-Y-Z geometry. Rebalance cannot be applied to problems with internal black absorbers, or zone dependent fission spectra.

Problems applying rebalance must be run in the "space stored" mode. Additional information on rebalance is given in reference [7].

7.15 Adjustable Diffusion Coefficients.

The diffusion coefficients can be adjusted in all zones as follows:

$$D'_{K,Z} = \alpha_1 + \alpha_2 D_{K,Z}$$

where α_1 and α_2 are input values, and $D_{K,Z}$ is the calculated diffusion coefficient for group K and zone Z. This may be applied, e.g., if data is available on the increase in diffusion coefficient produced by heterogeneity.

7.16 Equilibrium Xenon.

Equilibrium Xenon concentration has been programmed into VENTURE, and may be used on option. The effects of equilibrium Xenon can then be evaluated without the need to use the small exposure time steps required to build Xenon to its equilibrium value. This saves computer time and money. Reference [7] has additional information associated with equilibrium Xenon.

7.17 Temperature Correlation.

VENTURE and EXPOSURE include a temperature correlation for microscopic cross sections, so that temperature effects on reactivity can be evaluated. This requires two sets of microscopic cross sections at each of two reference temperatures. The correlation is as follows:

$$\sigma(T) = \sigma(T_1) + \frac{\tan^{-1}\left[\alpha(T-T_1)/(T_2-T_1)\right]}{\tan^{-1}(\alpha)}\left[\sigma(T_2)-\sigma(T_1)\right]$$

where α and the reference temperatures $T_1$and T_2 must be given.

8. EXPOSURE Code Module.

The code module for exposure calculations is the BURNER code [8]. Input for the EXPOSURE module is accomplished by generating the EXPINS record on the CONTRL file, and the standard interface file EXPOSE generated by the Input Processor. The EXPINS record and the Standard Interface File EXPOSE are given in the SIF Handbook.

Three techniques are used to solve the burnup equations: the matrix exponential method, the average generation rate method, and the explicit chain equation method. The methods differ by the numerical approximation techniques used to solve the coupled burnup equations. The average generation rate method uses the average precursor concentration over the time step in the precursor generation or production term. The matrix exponential method arises from expanding the exponentials in the solutions of the first order differential equations associated with burnup. The expansion order is carried out to the number of isotopes in the chain, in order to

account for all the nuclide couplings in the chain. The explicit chain equation method directly evaluates the solutions of the burnup equations for simple chain coupling, without making numerical approximations.

The matrix exponential method is recommended for complicated couplings in the nuclide chain, while the average generation method is recommended for simple chains. The full matrix exponential method requires about three or more times the computation when compared to the average generation method. The explicit chain method requires the most computational effort.

The BURNER code uses zone dependent fluxes and cross sections, so burnup as a function of core position can be evaluated.

A flow diagram for the BURNER code is shown in Figure 6. The subroutine listing for the code is given in Appendix III. The standard interface files associated with BURNER are shown in Figure 7. The primary zone exposure calculations require the availability of the following files: CONTRL; NDXSRF; GRUPXS; EXPOSE; RZFLUX; and ZNATDN.

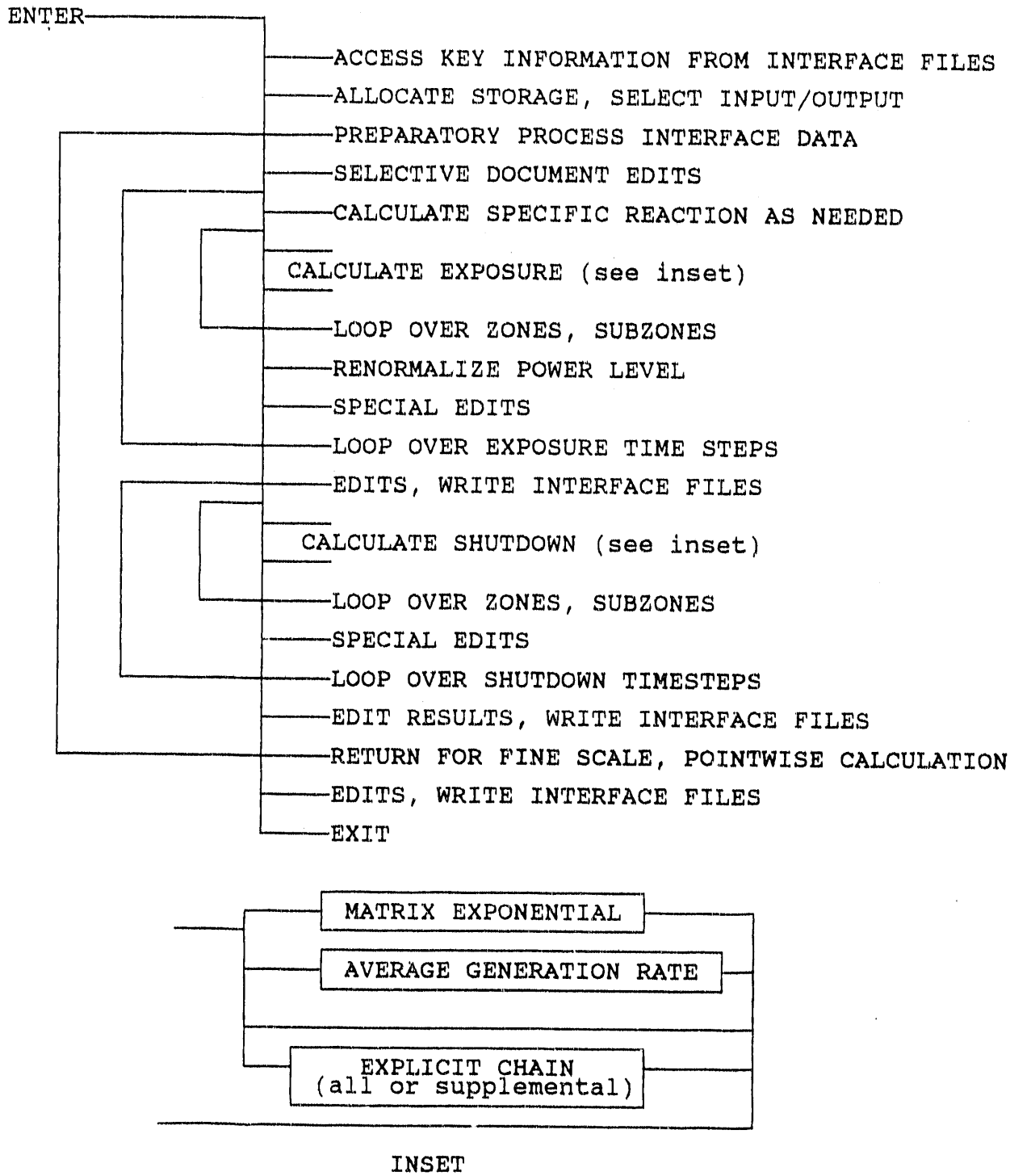


Figure 6. User Flow Diagram of BURNER Module (from ref. 8)

External Data Files Addressed in BURNER

CONTRL (read only)	Instruction records EXPINS, DVRINS, and PROINS
NDXSRF (read only)	Nuclide referencing data, and nuclide concentration assignment data
GRUPXS (read only)	Microscopic cross section data-group ordered
EXPOSE (read only)	Basic exposure data
RZFLUX (read only)	Zone average flux - also flux values at selected points for a geometry independent calculation (if modified)
ZNATDN (read/write)	Nuclide concentrations (zone and subzone)
PTATDN (read/write)	Nuclide concentrations (at selected points)
EXPOHT (read/write)	Continuously updated integrals of exposure conditions
ZNTEMP (read only)	Temperature data (zone and subzone)
QNATDN (write only)	Nuclide concentrations leaving the zones and subzones for the continuous fueling model (same form as ZNATDN)
ZNPOWD (write only)	Power density data (zone and subzone)
GEODST (read only)	Zone class data - also complete geometry processing for a geometry dependent calculation at selected points
RTFLUX (read only	Regular total flux - for a geometry dependent calculation at selected points

Figure 7. Interface Files Required by BURNER code (from ref. 8)

PART II. DESCRIPTION OF INPUT FOR VENTURE/PC

9. Input Structure.

The input consists of two main sections, i.e, the Control Module section and the Input Processor section. The input blocks associated with each input section begins with a header card, which contains the name of the input section, and terminates with an END card. An example of the input structure is shown in Figure 8. The first input section is always associated with the Control Module, which must be called first by the Driver to initialize the catalog file and the interface file CONTRL. The CONTRL file is referenced by all the modules for control information. The first card of the Control Module data contains an equal sign followed by the name of the Control Module, in this case "=CONTROL1", since CONTROL1 is the present name assigned to the Control Module. The Control Module input is formatted, so care must be used in placing the input data. The next input section is that of the Input Processor. The Input Processor will write standard interface files from free format card image or ASCII files written to comply with the standard CCCC file structure. A "STOP" card exists before the "END" card of the Input Processor.

```
=CONTROL1

                    CONTROL MODULE DATA

END
INPUT PROCESSOR

                    INPUT PROCESSOR DATA


STOP
END
```

Figure 8. Example of Input Structure

10. Control Module Input.

The user input instructions for the initial block of data on the input file associated with the Control Module is shown in Figure 9. The name of the present version of the Control module as developed at Los Alamos [4] is "CONTROL1" and must be on card 1 following an equal sign. This input is automatically inserted by the interactive input processor, VIP, supplied with the code, and is therefore not required when the interactive input processor is utilized to generate an input file.

10.1 Memory Allocation

After the title card, the first record on the third card (or record) is the memory allocation for the data of the problem. The maximum data memory that can be accommodated for the PC version of VENTURE is 36000 words (or 144000 bytes), and this is the default value used in VIP for running the code. Smaller memory allocations can be used, but not higher, since, otherwise, the 640K memory limit of DOS would be exceeded. IT IS ADVANTAGEOUS TO USE THE MAXIMUM MEMORY ALLOWABLE, since then if the problem dimensions permit, flux files used for iteration will be written as arrays in core. In addition, if there is still sufficient memory, source and other data files will be contained in core. The use of core for data storage and retrieval, significantly speeds the iteration process, as compared to disk storage and retrieval. There are times, however, when reducing the data storage allocation will help

run a problem that may not run with the maximum data storage. This is a result of the dependence of the data handling mode on the storage allocation. The data handling mode determines whether data is stored in memory or on disk, and, therefore, changes with the data memory allocation. Some problems may not run with the data handling mode associated with the maximum storage allocation. As an example, direct access file records may be larger than the Lahey compiler allows (i.e., 32 kilobytes) with the data handling mode associated with the maximum storage allocation, but these record lengths would be less than the maximum with a data handling mode associated with a smaller storage allocation. Thus, if a problem doesn't run with the maximum storage allocation, adjusting the allocation to a lower value which changes the data handling mode may help.

For application to computers other than those limited by the 640K DOS limit, core storage can be increased by increasing the dimension of the data container array, designated by "A", in the FORTRAN of the DRIVER module. This container array is presently dimensioned at 36000 words. For larger core computers, increasing this dimension will allow larger problems to run within core, and greatly speed convergence. Of course, recompilation would be necessary to change the container array dimension.

10.2 Using Binary Standard Interface Files as Input.

The parameter IP7 in columns 37 to 42 on record 3 must be set greater than zero if an interface data file is to be used as input.

In particular, if an interface data file such as an ISOTXS or GRUPXS file is to be input, IP7 must be set to 1 or higher. This option would be used, for example, if a GRUPXS file was generated in a previous run, and is to be used in the present run. The names of the interface files to be used as input are given by the parameter H(I) on record 5. Each interface file has a 6 letter name, and is stored as a double word containing 8 bytes, i.e., as REAL*8.

10.3 Calculational Path.

The calculational path is defined by the sequence of numbers on the next, or fourth record. This sequence is associated with the numbers assigned to the modules on Figure 1. This data record defining the calculational path is crucial for the proper operation of the modules in the code, and must be constructed with care and a good understanding of how the modules interact with each other. Interface files must be available to the modules as required when they are called.

10.4 Machine Dependent and Miscellaneous Data.

Data associated with processor time is not applicable to VENTURE/PC, since such data is machine dependent, and is designed for an IBM mainframe installation. Default values of zero should be used for such input, as IP5 on record 3. Several data locations have been reserved by the originators of BOLD VENTURE for future use, and should be left blank or assigned a zero value.

RECORD NUMBER	COLUMNS (FORMAT)	REFERENCE NAME	TYPICAL ENTRY	USE
1	1		=	PRECEDES CONTROL MODULE NAME.
1	2-9		CONTROL1	CONTROL MODULE NAME
2	(12A6)	TITLE		THE RUN TITLE CARD
3	1-6	IP1	36000	PRIMARY MEMORY ALLOCATION FOR DATA (NOTE: 36000 CAN BE USED FOR ALL PROBLEMS PROVIDED 640K OF MEMORY IS AVAILABLE.)
3	7-12	IP2	0	RESERVED
3	13-18	IP3	0	RESERVED
3	19-24	IP4	0	NA
3	25-30	IP5	0	NA
3	31-36	IP6	0	NA
3	37-42	IP7		ADDITIONAL INPUT IS INCLUDED TO PRESENT INTERFACE DATA FILE INFORMATION, IF > 0
3	43-48	IP8		A MODULAR NUMBER. DATA FILES WILL BE SAVED AFTER EACH ACCESS OF MODULE NUMBERED IP8
3	49-54	IP9		RESERVED
3	55-57	IP10	0	
3	58-60	IP11	0	

FIGURE 9. USER INPUT INSTRUCTIONS TO CONTROL MODULE CONTROL1

RECORD NUMBER	COLUMNS (FORMAT)	REFERENCE NAME	TYPICAL ENTRY	USE
3	61-63	IP12	1	INITIALIZE INTERFACE DATA FILE TABLE INITIALIZATION OPTION 0 - USE SEEK DEFAULT TABLE (SETS UP 15 INCLUDING CONTRL AS THE FIRST)[b] > 0 - SET UP THIS MANY FROM THE LIST OF WHICH CONTRL IS THE FIRST
3	64-66	IP13	0	OPTION ON REINITIALIZATION OF INTERFACE DATA FILE TABLES FOR ANY SUBSEQUENT ACCESS OF THE INPUT PROCESSOR AFTER THE FIRST 0 - NO ACTION (LEAVE AS IS) > 0 - RETAIN THIS MANY (PLUS CONTRL)
3	67-69	IP14	0	EDIT LEVEL FOR CONTROL MODULE PRIMARILY FOR DEBUGGING IF > 0
3	70-72	IP15	0	PRIMARY TERMINATION OPTIONS
4	(24I3)	IM(I)		STRING OF INTEGERS PRESENTING THE CODE MODULE NUMBERS TO DEFINE THE CALCULATIONAL PATH, TERMINATING WITH A BLANK ENTRY. A LOOP OVER A SUBSTRING OF TWO OR MORE CODE MODULES IS DEFINED BY A NEGATIVE NUMBER WHICH IS THE NUMBER OF PREVIOUS ENTRIES INCLUDED IN THE LOOP, AND THE NEXT ENTRY IS THE NUMBER OF PASSES THROUGH THE LOOP.

FIGURE 9. USER INPUT INSTRUCTIONS TO CONTROL MODULE CONTROL1 (cont.)

5	(9(2X,A6))	H(I)	THIS IS A STRING OF INTERFACE DATA FILE NAMES. TERMINATION OF EACH SET OF NAMES IS WITH A BLANK 8 COLUMN FIELD, OR AN ENTRY "X" IN THE LAST COLUMN OF A SEPARATED 8 COLUMN FIELD. THIS PROVIDES NAMES OF FILES MADE AVAILABLE FROM A PREVIOUS RUN.[d]
6	(A3)	END	NORMAL DATA BLOCK TERMINATOR.

[b] These files are CONTRL(10), GRUPXS(11), GEODST(12), NDXSRF(13), ZNATDN(14), SEARCH(15), RSTRTR(16), RTFLUX(17), ATFLUX(18), RZFLUX(19), PWDINT(20), CXSPRR(30), ISOTXS(31), ISOTXS(32), FIXSRC(33). The numbers in parenthesis are the logical unit numbers assigned to the files. The two ISOTXS files are required for the application of temperature effects.

[d] IP12 should be 1. Assignment of unit numbers is in increasing order beginning with unit 11.

FIGURE 9. USER INPUT INSTRUCTIONS TO CONTROL MODULE CONTROL1 (cont.)

10.5 Example of Input for Control Module.

An example of input for the Control Module is shown in Figure 10. The memory allocated for data in this problem is 15000 words. The "1" on card 3 in column 63 indicates that only one initial entry will be made on the catalog file, that entry associated with the file CONTRL, which must always be available. The sequence of numbers on card 4 indicate the order of the modules to be accessed. The numbers are associated with the module number assignments on Figure 1. For this problem, the Input Processor, designated number 1, will be accessed first. The VENTURE code is labeled number 7, is then called. The Input Processor must generate all of the standard interface files required by the VENTURE neutronics module. The final "0" on card 4 indicates that no additional modules will be called.

```
=CONTROL1
TWO DIMENSIONAL, NON SEPARABLE WATER REACTOR PROBLEM
 15000     0     0     0     0     0     0     0     0 0 0 1 0 0 0
  1  7  0
END
```

Figure 10. Example of Input to Control Module

11. Input for Input Processor.

After the Control Module input is processed, additional input must be available for the Input Processor. The primary function of the Input Processor is to create interface files entirely from card input. The card input follows a structure similar to the structure associated with the interface files themselves, and uses free format. Blanks are recommended between data entries, but other than that the data can reside anywhere on a record or card. See the Standard Interface File Handbook for details associated with using the Input Processor to generate input data.

12. Standard Interface Files Required by Modules.

The standard interface files required by the various modules are shown in Figure 11. Figure 11 is an extremely important figure for the proper understanding of input to VENTURE/PC. It shows the necessary control records on the file CONTRL, and indicates which standard interface files are required to run the various modules, and which files may be written by the various modules. Thus, this figure contains invaluable information associated with setting up a calculational pathway. Figure 11 should be referred to when determining VENTURE/PC input.

The first Standard Interface File produced by the Input Processor must be the CONTRL file. The Control Module requires the PROINS and DVRINS records on the CONTRL file in that order, and these records must be the first records written on the CONTRL File. Refer to the Standard Interface File Handbook for details of the CONTRL file and the records for this file. The PROINS and DVRINS

records are global in nature, and are required for all runs with VENTURE/PC. Additional records which are problem dependent must be placed on the CONTRL file. If the Cross Section Procesor is required, the XCPINS record must be written on the CONTRL File. The DTNINS record is required to run the VENTURE neutronics module, so would be required for all neutronics problems. If an exposure or burnup problem is to be run, the EXPINS record is required for the BURNER module. After the PROINS and DVRINS records, the other records can appear in any order on the CONTRL file.

Additional Standard Interface Files are required by the Cross Section Processor, VENTURE neutronics module, and the Exposure Module. If the Cross Section Processor reads CITATION microscopic cross sections, or updates existing ISOTXS files, or merges ISOTXS files, or is used to create mixtures, the CXSPRR Standard Interface File (SIF) is required (see Figure 1). If, on the other hand, the Cross Section Processor is used only to edit an existing ISOTXS file, or convert an existing ISOTXS file to a GRUPXS file, or edit a GRUPXS file, the CXSPRR SIF is not necessary.

In addition to the control information given on the DTNINS record for VENTURE, other important information must be made available to run a multigroup diffusion code like VENTURE. All cross sections must be in group ordered form, so a GRUPXS file is required. The geometry must be specified, the mesh spacing set up, and the boundary conditions must be given. Thus, a GEODST SIF is necessary. Cross sections must be identified and assigned to mixtures, and mixtures must be asigned to zones. This is accomplished through the NDXSRF SIF. In addition, the atomic

densities of the isotopes in mixtures assigned to zones must be given. Thus a ZNATDN SIF is required for VENTURE. The manner in which number densities of isotopes are calculated for VENTURE is discussed in section VII of the SIF Handbook.

If a burnup problem is run, in additiion to the EXPINS record on the CONTRL file, the EXPOSE SIF must be made available to the BURNER module. This is in addition to other SIF's produced by VENTURE, which is run prior to BURNER.

Module	No.	Name of Record in File CONTRL	Required	Required on Option	Written on Option	Ref.
CONTROL		PROINS DVRINS		(any)	(any)	1
INPUT PROCESSOR	1			(any)	(any)	1,8
CROSS SECTION PROCESSOR	6	XCPINS	ISOTXS	GRUPXS CXSPRR	GRUPXS ISOTXS	2
VENTURE NEUTRONICS	7	DTNINS	GEODST NDXSRF ZNATDN GRUPXS	RTFLUX ATFLUX RZFLUX FIXSRC RSTRTR SEARCH ZNTEMP	RTFLUX[a] ATFLUX RZFLUX FIXSRC RSTRTR PWDINT PERTUB FISSOR ZNPOWD GEODST	2
EXPOSURE	13	EXPINS	EXPOSE NDXSRF ZNATDN GRUPXS RZFLUX	PTATDN ZNTEMP GEODST[b] TRIGOM[b] EXPOHT CFHIST RTFLUX	PTATDN QNATDN ZNATDN ZNPOWD EXPOHT[a] CFHIST[a]	

[a]May be created if it does not exist
[b]Will be used if it exists

Figure 11. Module Control Records and Interface Files (From ref. 1)

13. Some Examples of Input Structure.

Examples of input structure, including the structure for the sample problems [9], will be given in a format which will provide the calculational path given in the Control Module input, and the Standard Interface Files and control records generated by the Input Processor. The details of the files can be examined by viewing the sample problem input provided with the code package. From the point of view of learning the input, however, it is best to give and explain the basic structure without the file details.

13.1 NEW2DXY.INP - The input structure for the revised two dimensional X-Y problem is shown in Figure 12. The Control Module input indicates that the Input Processor is to be called first, followed by the VENTURE neutronics module. This is indicated by the calling sequence 1 7 0 in the Control Module data, where the "0" indicates the end of the problem. These numbers correspond with the module numbers given on Figure 1. The Input Processor must generate all the files required by the VENTURE neutronics module for this problem. The control records on the Standard Interface File CONTRL are generated first. All of these records are considered to be 1D Records. The PROINS record is always given first, and always followed by the DVRINS record. The DTNINS record provides the control information for the VENTURE neutronics module and follows the other two records which are used for global information. The CONTRL file always ends with a blank record designated by 6 blanks between asterisks. VENTURE needs the following additional SIF's to proceed with the calculation: GRUPXS for cross sections; GEODST for geometry specifications; NDXSRF for cross section referencing; and

ZNATDN for zone atomic densities. These are all developed through the Input Processor. The Input Processor always concludes data input with a STOP card followed by an END card.

```
=CONTROL1
TWO DIMENSIONAL, NON SEPARABLE WATER REACTOR PROBLEM
36000
1  7  0
END
INPUT PROCESSOR
0V CONTRL
1D PROINS
1D DVRINS
1D DTNINS
1D *      * 0.0 100R 0 100R
0V GRUPXS
0V GEODST
0V NDXSRF
0V ZNATDN
STOP
END
```

Figure 12. Input Structure for Two Dimensional X-Y Problem.

13.2 NEWBWR.INP - The input structure for the revised BWR assembly X-Y analyses is shown in Figure 13. This data combines the 12 X 12 and 24 X 24 cases given separately in previous versions of VENTURE/PC. The Control Module indicates the following sequence for calling modules: 1 6 7 1 7 0. Thus, the Input Processor (1) is called first, followed by the Cross Section Processor (6), and the VENTURE neutronics module (7). These are followed by the Input Processor (1) and VENTURE module (7) to run the second problem. CITATION cross sections were used for this problem. The CITATION cross sections must be processed by the separate DCMACR code to convert macroscopic cross sections to microscopic and write them on unit 8. The DCMACR processing must be done on the same subdirectory as the VENTURE code.

The control records are generated on the CONTRL file by the Input Processor. These include, as always, the global records PROINS and DVRINS in that order. The XCPINS control record is required for this problem, since the Cross section Processor is used to process the CITATION cross sections and convert them to GRUPXS form. Then, of course, the DTNINS file is required to run the VENTURE module.

The CXSPRR Standard interface file is also necessary for this problem, since CITATION cross sections are processed.

In addition to cross sections, VENTURE also needs the GEODST, NDXSRF and ZNATDN files.

The same cross sections are used for the second problem, but different GEODST, NDXSRF and ZNATDN files are required. Thus, the Input Processor is rerun to generate the new files.

```
=CONTROL1
36000
1  6  7  1  7  0
END
INPUT PROCESSOR
0V CONTRL
1D PROINS
1D DVRINS
1D XCPINS
1D DTNINS
1D *      *
0V CXSPRR
0V GEODST
0V NDXSRF
0V ZNATDN
STOP
END
INPUT PROCESSOR
0V GEODST
0V NDXSRF
0V ZNATDN
STOP
END
```

Figure 13. Input Structure for BWR Assembly Sample Problems.

13.3 Three Theta-R Cases - The input for three theta-R cases run in succession is shown in Figure 14. The calling sequence is 1 6 7 1 7 1 7 0, indicating that the Input Processor is called first, followed by the Cross Section Processor and the VENTURE neutronics module, which completes the first problem. The remaining two problems require the Input Processor and VENTURE module in that order called twice, once for each problem.

This problem also uses CITATION cross sections. The Input Processor first writes the necessary records on the CONTRL file. These include: PROINS, DVRINS, XCPINS and DTNINS, as in the previous problem. The CXSPRR Standard Interface File must be written for the Cross Section Processor to convert CITATION cross sections for use by VENTURE. The GEODST, NDXSRF and ZNATDN files must be written for VENTURE. The Input Processor is called twice more, in between calls to VENTURE, to write new GEODST, NDXSRF and ZNATDN files to run the remaining two problems.

```
=CONTROL1
36000
1   6   7   1   7   1   7   0
END
INPUT PROCESSOR
0V CONTRL
1D PROINS
1D DVRINS
1D XCPINS
1D DTNINS
1D *       *
0V CXSPRR
0V GEODST
0V NDXSRF
0V ZNATDN
STOP
END
INPUT PROCESSOR
0V GEODST
0V NDXSRF
0V ZNATDN
STOP
END
INPUT PROCESSOR
0V GEODST
0V NDXSRF
0V ZNATDN
STOP
END
INPUT PROCESSOR
0V EXPOSE
STOP
END
```

Figure 14. Input Structure for Three Theta-R Problems.

13.4 Exposure Problem. - An input structure for an exposure problem is shown in Figure 15. In this problem the calling sequence is: 1 6 1 7 13 7 13 7 13 7 0. The Input Processor (1) is called to write the CONTRL file, followed by the Cross Section Processor (6) required to convert an ISOTXS file to a GRUPXS file, followed by the Input processor again to write the EXPOSE Standard Interface File required by the BURNER module, followed by the VENTURE (7) neutronics module which calculates the flux for the BURNER module (13) and the initial burnup step, afterwhich The VENTURE and BURNER modules are alternated for the number of burnup steps desired (3 in the example shown). The Input Processor first writes the PROINS and DVRINS control records which are always required. It then writes the XCPINS record for the Cross Section Processor, the DTNINS record for the VENTURE module, and the EXPINS record for the BURNER module. It also writes the ISOTXS file which is converted by the Cross Section Processor to a GRUPXS file, and the three additional Standard interface Files required by VENTURE, i.e., the GEODST, NDXSRF, and the ZNATDN files. In addition, it is called again to write the EXPOSE file required to run the BURNER module.

```
=CONTROL1
REFERENCE CALCULATION FOR DPT SAMPLE PROBLEMS
36000
1  6  1  7  13  7  13  7  13  7  0
END
INPUT PROCESSOR
0V CONTROL
1D PROINS
1D DVRINS
1D XCPINS
1D DTNINS
1D EXPINS
1D *     *
0V ISOTXS
0V GEODST
0V NDXSRF
0V ZNATDN
STOP
END
INPUT PROCESSOR
0V EXPOSE
STOP
END
```

Figure 15. Input Structure for an Exposure Problem.

13.5 Using Standard Interface Files on Input.

If Standard Interface Files are available on the VENTURE subdirectory, they can be used directly without using the Input Processor to generate them. This is accomplished through the Control Module, as previously explained (Section 10.2). An example for running a VENTURE problem with all of the necessary SIF's available, i.e., GRUPXS, GEODST, NDXSRF and ZNATDN is shown in Figure 16.

```
=CONTROL1
EXAMPLE OF VENTURE RUN WITH NECESSARY INTERFACE FILES AVAILABLE
 36000                                    1                              1
7  0
  GRUPXS  GEODST  NDXSRF  ZNATDN      X
END
```

Figure 16. Example of Input with Standard Interface Files

14. Relocating Fuel Bundles.

In version 2 of VENTURE/PC, the special processor DENMAN was used to change mixtures assigned to zones or subzones, and to change nuclide concentrations, which in effect could be used to simulate core bundle reshuffling. In version 3, however, all special processors have been removed, including DENMAN. Fuel reshuffling can be simulated, however, by using the Input Processor to overlay and change the NDXSRF and ZNATDN standard interface files.

15. Data Transfer, File Management and Input-Output.

Data transfer and file management are accomplished with standardized routines which are used by all code modules and subroutines.

15.1 Standardized Routines.

The standardized subroutines and their functions are:

REED - Transfers data from disk to memory.

RITE - Transfers data from memory to disk.

DOPC - Basic input-output management. Opens and closes files, and differentiates between sequential and direct access files. Only scratch files can use direct access. Direct Access Files cannot be named with the Lahey compiler used.

SEEK - Maintains file catalog, keeping track of file name, unit number, and version number. Subroutines often call SEEK to establish the unit number for a given file.

15.2 Input and Output Files.

The input to run a given problem must be on a file named VENTURE.INP, which is assigned unit number one. The Driver overwrites the input for the various modules, i.e., the input between the header card and END card, on unit 5, and all data is read from this unit. This file on unit 5 has been named VENTURE.TMP. All output is written on unit 6, in the file labeled VENTURE.OUT, so the output can be read by editing this file. The output is quite extensive, so the VENTURE.OUT file should be edited prior to printing. A condensed output file is written on unit 99, which provides file management and data access information. This file has been labeled CONDENSE.OUT.

15.3 Saving of Standard Interface Files.

The standard interface files generated during a run are saved with their name, so they may be used in subsequent runs. These include, with their usual unit number assignments:

CONTRL - Unit 10

GRUPXS - Unit 11

GEODST - Unit 12

NDXSRF - Unit 13

ZNATDN - Unit 14

SEARCH - Unit 15

RSTRTR - Unit 16

RTFLUX - Unit 17

ATFLUX - Unit 18

RZFLUX - Unit 19

PWDINT - Unit 20

CXSPRR - Unit 30

ISOTXS - Unit 31

ISOTXS(2) - Unit 32

FIXSRC - Unit 33

15.4 Scratch and Direct Access Files.

Other files which are generated during a run are named VENTNO, where the NO is the unit number of the file. Thus, VENT36 is a file generated during the run on unit 36. These files are deleted at the end of a successful run, but will appear on the disk in the case of an aborted run. Direct access files are not saved.

The general assignment of files is as follows:

Unit 09 - Catalog File

Unit 10 - CONTRL file

Units 21-29 and 40-69 - Scratch files

Units 10-20 and 30-39 and 70 - 97 - Standard Interface Files

Unit 98 - Control module instructions

15.5 Saving of Standard Interface File in Text Format.

If the selection was made to save results as formatted data (see parameter IX(60) of DTNINS input), the results will be on a file named STFILE.TXT, for Standard Files in text format.

16. Correspondence Between DVENTR and DTNINS.

The control information for the VENTURE neutronics code could be input through the special processor DVENTR in version 2 of VENTURE/PC. The same information is included in the DTNINS control record, and is input through this record in version 3 of VENTURE/PC. The correspondence between the input parameters in DVENTR and DTNINS, taken from reference [1] is shown in Table 16.1.

Table 16.1 Correspondence Between DVENTR and DTNINS Parameters

DVENTR Input Sec.001	DTNINS Record	DVENTR Input Sec. 001	DTNINS Record	DVENTR Input Sec. 001	DTNINS Record
RRX1	XX(1)	ICX1	IX(4)	IXE1	IX(32)
RRX2	XX(2)	ICX2	IX(5)	IXE2	IX(33)
RXX3	XX(3)	ICX3	IX(10)	IXE3	IX(35)
RXX4	XX(4)	ICX4	IX(6)	IXE4	IX(38)
RXX5	XX(5)	ICX5	IX(9)	IXE5	IX(39)
RXX6	XX(6)	ICX6	IX(8)	IXE6	IX(40)
RXX7	XX(14)	ICX7	IX(2)	IXE7	IX(41)
RXX8	XX(9)	ICX8	IX(26)	IXE8	IX(42)
RXX9	XX(10)	ICX9	IX(23)	IXE9	IX(37)
RXX10	XX(8)	ICX10	IX(22)	IXE10	IX(31)
RXX11	XX(11)	ICX11	IX(21)	IXE11	IX(29)
RXX12	XX(12)	ICX12	IX(20)	IXE12	IX(30)
RXX13	XX(13)	ICX13	IX(27)	IXE13	IX(61)
RXX14	XX(15)	ICX14	IX(25)	IXE14	IX(62)
RXX15	XX(16)	ICX15	IX(24)	IXE15	IX(51)
RXX16	XX(7)	ICX16	IX(12)	IXE16	IX(58)
RXX17		ICX17	IX(17)	IXE17	IX(53)
RXX18		ICX18	IX(18)	IXE18	IX54
		ICX19	IX(19)	IXE19	IX(52)
		ICX20	IX(16)	IXE20	NO
		ICX21	IX(13)	IXE21	IX(81)
		ICX22	IX(14)	IXE22	IX(45)
		ICX23	IX(15)	IXE23	IX(59)
		ICX24	IX(70)	IXE24	IX(60)

Table 16.1 Correspondence Between DVENTR and DTNINS Parameters (continued)

DVENTR Input Sec. 002	DTNINS Record	DVENTR Input Sec. 002	DTNINS Record	DVENTR Input Sec. 002	DTNINS Record
RXXN1	XX(19)	IXCN1	IX(11)	IXEN1	IX(46)
RXXN2	XX(20)	IXCN2	IX(7)	IXEN2	IX(47)
RXXN3	XX(21)	IXCN3	IX(73)	IXEN3	IX(48)
RXXN4	XX(22)	IXCN4	IX(74)	IXEN4	IX(55)
RXXN5	XX(23)	IXCN5	IX(75)	IXEN5	IX(28)
RXXN6	XX(24)	IXCN6	IX(76)	IXEN6	IX(34)
RXXN7	XX(25)	IXCN7	IX(77)	IXEN7	IX(36)
RXXN8	XX(26)	IXCN8	IX(78)	IXEN8	IX(43)
RXXN9	XX(27)	IXCN9	IX(79)	IXEN9	IX(44)
RXXN10	XX(28)	IXCN10	IX(80)	IXEN10	IX(49)
RXXN11	XX(29)	IXCN11		IXEN11	IX(50)
RXXN12	XX(30)	IXCN12	IX(82)	IXEN12	IX(56)
		IXCN13	IX(83)	IXEN13	IX(57)
		IXCN14	IX(84)	IXEN14	IX(63)
		IXCN15	IX(85)	IXEN15	IX(64)
		IXCN16	IX(86)	IXEN16	IX(65)
		IXCN17	IX(87)	IXEN17	IX(66)
		IXCN18	IX(88)	IXEN18	IX(67)
		IXCN19	IX(89)	IXEN19	IX(68)
		IXCN20	IX(90)	IXEN20	IX(69)
		IXCN21		IXEN21	
		IXCN22		IXEN22	
		IXCN23		IXEN23	
		IXCN24		IXEN24	

17. Compiler and Overlay Structure for VENTURE Code.

The VENTURE/PC code modules were obtained from the BOLD VENTURE code system by downloading the source files from a mainframe tape to a PC via modem. The source codes were edited, and the subroutines extracted and combined with the BRIEF PC Editor [10], and compiled with the Lahey Fortran 77 compiler [11].

The executable file for VENTURE/PC requires 3 million bytes, so cannot be run within the 640K of DOS without overlay. The PHOENIX86 Plus [12] overlay linker was used to reduce the size of the runtime code to about 540K. This overlay linker allows for several levels of overlay. The overlay structure is such that only subroutines which call each other, or depend on each other, are in memory simultaneously, while the others reside on disk waiting to be called. When they are called, they overlay the existing routines in memory. In this manner, large codes can be run within memory restrictions. The overlay structure for VENTURE, as taken from the PLINK86 input, is shown in Figure 17. Files which are on the same line in this figure, are in memory simultaneously. This is the case for the DRIVER and VENTNEUT, for example, and for the subroutines VENT and IONO. The DRIVER and VENTNEUT are in the root, and are always in memory. Files which are in the same sections, but on different lines, overlay one another in memory. Thus, CONTROL1 and INPROSER do not reside in memory together, but overlay each other. Sections which are indented, reside in another level of overlay, and require their parent file in memory. Thus, in using the Input Processor, INPROSER must be in memory along with the root files DRIVER and VENTNEUT, but the files CGE, CIO, CGR etc., can overlay

each other as they are called by INPROSER. Actually, the file names shown on the figure consist of more than one subroutine which were compiled together, and which must be in memory together since they call each other. The subroutines assigned to the various names of Figure 17 are shown in Figure 18. Figure 18 can be compared with Appendix II and III which define the subroutines for VENTURE and BURNER, respectively.

In addition to overlaying code, the PLINK86 overlay linker provides an option to overlay data segments [13]. When this option was selected, the container array for data could be increased from 16000 words to 36000 words. This is a very significant increase, and allows for much larger problems to be run.

```
OUT VENTURE
FILE DRIVER3,VENTNEUT
LIB F77L
LIB OVERLAY
OVERLAY F77LCODE,F77LDATA
BEGIN SECTION FILE CONTROL1
      SECTION FILE INPROSER
              BEGIN SECTION FILE CGE
                    SECTION FILE CIO
                    SECTION FILE CGR
                    SECTION FILE CIS
                    SECTION FILE CFX
                    SECTION FILE CCX
                    SECTION FILE CTF
                    SECTION FILE CRZ
                    SECTION FILE CDL
                    SECTION FILE CEX
                    SECTION FILE CZN
                    SECTION FILE CRF
              END
      SECTION FILE CROSPROS
              BEGIN SECTION FILE ITI
                    SECTION FILE CTI
                    SECTION FILE M2I
                    SECTION FILE MIX
                    SECTION FILE IXS
                    SECTION FILE XSC
                    SECTION FILE GXS
              END
      SECTION FILE VENT,IONO
      SECTION FILE SGX
      SECTION FILE COR
      SECTION FILE MAC
      SECTION FILE COC
      SECTION FILE ORL
      SECTION FILE PHI
      SECTION FILE COP
      SECTION FILE COM
      SECTION FILE OUT
              BEGIN SECTION FILE MUE
                    SECTION FILE DOI
                    SECTION FILE FO1
                    SECTION FILE FO2
                    SECTION FILE FO3
                    SECTION FILE FO4
                    SECTION FILE FO5
                    SECTION FILE FO6
                    SECTION FILE FOU
             END
```

Figure 17. VENTURE Overlay Structure

```
        SECTION FILE EDI,DARE
        SECTION FILE SAV
        SECTION FILE PER
                BEGIN SECTION FILE PET
                      SECTION FILE JET
               END
        SECTION FILE EXPOSURE
        BEGIN SECTION FILE BIN
        SECTION FILE BZI
        SECTION FILE BUR
                BEGIN SECTION FILE OEX
                      BEGIN SECTION FILE OFI
                            SECTION FILE OMO
                                    BEGIN SECTION FILE ZON
                                          SECTION FILE PON
                                     END
                      END
                      SECTION FILE OXP
                              BEGIN SECTION FILE EXH
                                    SECTION FILE QXP
                              END
                      SECTION FILE EDE
                      SECTION FILE OOW
                END
        SECTION FILE FOL
        SECTION FILE BPN
                BEGIN SECTION FILE BPA
                      SECTION FILE BPB
                      SECTION FILE BPC
                END
        SECTION FILE PUR
        END
END;
```

Figure 17. VENTURE Overlay Structure (Cont.)

SUBROUTINE COMBINATIONS USED FOR LINKING VENTURE/PC

CONTROL1 - CONTROL1, INTL, NORM, CFIL, DLET, PUTF, TABL, WCTL. WRAP

DRIVER - GABY, FERR, RITE, SEEK, RCVI, STOR, SKER, PRTH, PRTT, PRTI, PRTR, PRTD, CMPI, CMPH, CRIT, UROC, CLOSFI, CSCRCLO

INPROSER - INPROSER, ABEL, CDINP, CAIN, KEEP, FBSAM, STOW, INCHEK, INCHKD, ERMESG, GENRD, CDINPT, ERRMSG, READHM, DCODNC, INTERP, REPEST, SETFMT

VENTNEUT - VENTNEUT, DIFF, DOPC

CGE - CGEODS, CNDXSR, CZNATD

CIO - CISOTX

CGR - CGRPXS

CIS - CISOGX

CFX - CFXSRC, CSEARCH, CSNCON

CCX - CCXSPR, CVENTR, CPRINT, CISOGR

CTF - CTFLUX, CCURNT, CAFLUX

CRZ - CRZFLUX, CPERTU, CPWDNT, CFISOR

CDL - CDLYXS, CBRKXS, CWORTH, CANSIN, CDACIN

CEX - CEXPOS, CZCONC, CRODST

CZN - CZNTMP, CZNPOW, CEXPOH, CPTATD

CRF - CRFUEL, CTRIGM

Figure 18. Subroutines in VENTURE/PC Overlay Structure

ITI - ITI1, ITI2

CTI - CTI1, CTI2, CTI3, CHOL

M2I - M2I1, M2I2

MIX - MIX1, MIX2, MIX3, MIX4, MIX5, MIXC

IXS - IXS1, IXS2

XSC - XSC1, XSC2, XSC3, XSC4, XSC5_

GXS - GXS1, GXS2

INDVENTR - INDVENTR, INTL, INP1, VONT

GEM - GEOM, GOMN

REG - REGD

TRF - TRIF, HEXF

THK - THKD, VOLS, MSHP

KOM - KOMP, KMOT

OVL - OVLY, MOSH, OVLP

SSE - SSET

GOM - GOMA

DEN - DENS

GDN - GDNA

SES - SETS

SEA - SEAR

GMO - GOM1, GOM2

Figure 18. Subroutines in VENTURE/PC Overlay Structure (continued)

ZND ZND1, ZND2

NDR NDR1, NDR2

CTL - CTL1

VENT - VENT

IONO - IONO

SGX - SGX0, SGX1, SGX2, SGX3, SCAL_

COR - CORE, CORI, CORP, GNAM, CORD, CORB, DDSP, DASM, JPRT, RBLA, RBLB

MAC - MAC1, VZT2, MACA, MACB, MAC2, MAC3, MAC5, CHDM, MAC4, MAC6, SERM

COC - CON1, CON2, CON3, CKCT, CON4, CON5, CON7, CON9, GEOQ, MSH0, NRCF, MSH1, MSH3

ORL - ORLX, ORLA, ORLB, ORLC, ORLD, ORLE, ORLF, LAXP, LAXR, BATO, ORLR, CON6, RCOV

PHI - PHIA, PHI1, PHI2, PHI3, PHI4, PHI5, PHI6, EDBN, SDBN, PHI7, TOIP, QDBN, RDBN, GRXP, PAN1, PAN2

COP - ADN1, ADN2, ADN3, DCID, DSDF, DIMS, DIM1, DIM2, CMES, CHVL, ALDS, DIM3, ZVRV, CRGV, FLRD, FLMH

COM - COMC, LCAL, FLXR, FXSR, BSQV, AJNT, REV1, PROS, ZIO3, FEFS, CORR, IFTD, ONES, TWOS, TRES, HST1, HST2, HST3

OUT - OUTR, BALC, ZINS, CHBF, CHEV, RDAB, XTRP, JUSB, ATED, FFGG, RDUE, RELX, PSOR, SSOR, FSOR, FLUX, LTRG, BHAV, OELX, NEWB

MUE - MUEX, ETR1, ETR2, SGDA

DOI - DOIN, RRES, WRES, PREC

FO1 - FOU1, SOU1, POU1, INR1, LOU1, LEK1

FO2 - FOU2, SOU2, POU2, INR2, LOU2, LEK2

FO3 - FOU3, SOU3, POU3, INR3, LOU3, LEK3, RBL1, RBL2, RBL3, RBL4, RBL5

FO4 - FOU4, SOU4, POU4, INR4, LOU4, LEK4, QDUE, QELX, SOUX, J1C4

Figure 18. Subroutines in VENTURE/PC Overlay Structure (continued)

FO5 - FOU5, SOU5, POU5, INR5, LOU5, LEK5, J1C5

FO6 - FOU6, SOU6, INR6, DELX

FOU - FOUX, SOUY, POUX, INRX, LOUX, LEKX, SOUZ, J1CX

SAV - SAV1, SAV2, SAV3, SAV4, SAV5, SAV6, SAV7

PER - PERO, RTUB, MRPT, QOUT, BBB1, BBB2, EASU

EDI - EDIT, POUT, NBAL, SOBL, FISS, FLXW, BSQS, PNDN, PTVL, PTZF, JINT, PND1, PND2, PND3, PND4, PND5

DARE

PET - PERT, TUFY, LIFE, DAFA, MAPS, PMAP

JET - JERT, JUFY, JIFE, JAFA, JAPS, JMAP, JGET

THE NEXT SET OF SUBROUTINES ARE ASSOCIATED WITH EXPOSURE:

BIN - BINP, GNZC, BRN7, BRN4, BGXS, BZT1, BRNF, BRNW, ZJC2, ZJCY, ZIGY, EPFD, BRNX, HQUE, SKNU

BZI - BZIN, DEEF, CMOV, BRNS, BRN3, BRNZ, BRNT, BZT2, BRNA, BRND, BRRF, PRRF

OFI - OFIX, BFIX

OEX - OEXP, POWL, ARRI, ZCRI, PARI, BRNO

BUR - BURN, BRNY, ZNAW, ZZPD, AUXE, TPNE

ZON - ZOND, ZZPF, ZONI, POWP

OMO - OMOV, BMOV

PON - PONI, PPOE, QNAW, QNAT

OXP - OXPH, CPHI, ECHK, ESET, FLUE

EXH - EXPH, REHT, CPH2, EPH2

QXP - QXPH, QPTD, QFLU, QEHT, CPH3, EPH3

EDE - EDEP, EDED, ETAB

OOW - OOWN, DOWN

Figure 18. Subroutines in VENTURE/PC Overlay Structure (cont.)

FOL - FOUL

BPN - BPIN, PTNS

BPA - BPIA, PTAT, PGEO, GCHK, MSHK, VOLP, PLOC, PRN3, REOR, CHEK, MSHO, MSH1, MSH3, NRCF

BPB - BPIB, ZFMP, ZFM3, ZFMV

BPC - BPIC, PRNZ, PRNT, PZT2, PRNA, PRND, BRPF, PRPF, PRNS

PUR - PURN, PRNY, PFIX, POWN, PNAW, PDST

Figure 18. Subroutines in VENTURE/PC Overlay Structure (continued)

18. VENTURE/PC Interactive Processor, "VIP".

An interactive input processor is included with the code. The processor, called VIP for VENTURE INTERACTIVE PROCESSOR, was written in FORTRAN and compiled with the Lahey FORTRAN-77 compiler, version 2.22. The processor is rather large, utilizing about 900,000 bytes, and, therefore, required overlaying during the linking process. The overlay structure for the VIP executable program is shown in Figure 19.

The processor prompts the user for the input, and should be reasonably self explanatory. Reviews of the input are provided after each major section, at which point erroneous input can be corrected.

```
OUT VIP
FILE VIP
LIB C:\LAHEY\F77L
LIB OVERLAY
OVERLAY F77LCODE,F77LDATA
BEGIN SECTION FILE CONTRLU,CONTRL2
      SECTION FILE GRUPXS
      SECTION FILE ISOTXS
      SECTION FILE GEODST
      SECTION FILE NDXSRF
      SECTION FILE ZNATDN
      SECTION FILE SEARCH
      SECTION FILE EXPOSE
END;
```

Figure 19. VIP Overlay Structure

REFERENCES

1. "BOLD VENTURE IV, A Reactor Analysis Code System, Version IV", RSIC Computer Code Collection, CCC-459, Radiation shielding Information Center, Oak Ridge National Laboratory, June, 1984.
This reference provides the extensions made to the previous version of BOLD VENTURE. It also provides the updated input requirements.

2. R. Douglas O'Dell, "Standard Interface Files and Procedures for Reactor Physics Codes, Version IV", LA-6941-MS, Los Alamos Scientific Laboratory, September, 1977.
An excellent description of the purpose and format of Standard Interface Files and DOE code standardization.

3. Bosher, G.E., Odell, R.D., Resnik, W.M, "LASIP-III, A Generalized Processor for Standard Interface Files", LA-6280-MS, Los Alamos Scientific Laboratory, April, 1976.
A description and discussion of the Los Alamos Input Processor for converting card image format to Standard Interface Files.

4. Vondy, D.R., Fowler, T.B., Cunningham, G.W., Petrie, L.M., "A Computation System for Nuclear Reactor Core Analysis", Oak Ridge National Laboratory, ORNL-5518, April, 1977.
A description of the system and codes used with VENTURE for nuclear reactor core analysis.

5. Vondy, D.R., Fowler, T.B., Cunningham G.W., "VENTURE: A Code Block for Solving Multigroup Neutronics Problems Applying the Finite Difference Diffusion Theory Approximation to Neutron Transport, Version II", ORNL-5062/R1, Oak Ridge National Lab, Nov. 1977.
An earlier version of VENTURE, but the most definitive report on the VENTURE neutronics module, providing a detailed account of the theory and equations associated with the code.

6. Fowler, T.B., Vondy, D.R., Cunningham, G.W., "Nuclear Reactor Core Analysis Code: CITATION", ORNL-TM-2496, Rev. 2, July, 1969
The precursor code to VENTURE.

7. Vondy, D.R., Fowler, T.B., Cunningham, G.W., "The Bold Venture Computation System for Nuclear Reactor Core Analysis, Version III', Oak Ridge National Lab, ORNL-5711, June, 1981.
Essentially the same as reference 1, but given as an Oak Ridge report rather than as a Computer Code Collection.

8. Vondy, D.R. and Cunningham, G.W., "Exposure Calculational Code Module for Reactor Core Analysis: BURNER", ORNL-5180, Oak Ridge National Lab., Feb. 1979.
A description of, and the theory used, in the BURNER code module for isotope depletion and production.

9. Vondy, D.R. and Fowler T.B., "Reference Test Problems for the VENTURE Neutronics and Related Computer Codes", ORNL/TM-5887, Oak Ridge National Lab, August, 1977.
A listing with input for VENTURE sample problems.

10. "BRIEF, Basic Reconfigurable Interactive Editing Facility, Ver 2", Underware Inc., 84 Gainsborough St., Suite 103W, Boston, Mass., 02115.
An excellent PC Editor for code development and editing.

11. "F77L FORTRAN 77 Language System, Ver 2.22", Lahey Computer Systems, Inc., P.O. Box 6091, Incline Village, Nv. 89450-6091, (702)831-2500.
A very excellent FORTRAN compiler, providing rapid compilation and running.

12. "PLINK86PLUS Overlay Linker, Ver 2.24", Phoenix Technologies Ltd, 320 Norwood Park South, Norwood MA. 02062 (800)344-7200.
A multilevel overlay linker for PC's.

13. Nigg, D.W., INEL, EG&G Idaho Inc., Personal Communication

APPENDIX I

FORTRAN LISTING OF DRIVER

```
CDRIVER FORTRAN VERSION OF THE DRIVER
C
C
C      PRIMARY DATA USE BY THE DRIVER
C
C      IC(1)           COUNT OF ACCESSES OF THE CONTROL MODULE,
C                      SET 0 FOR SUBSEQUENT CASE, -1 FOR TERMINATION
C      IC(2) - IC(6)   STOP RETURN NUMBERS ALLOWED FOR CODE MODULE ACCESSES
C      IC(7)           TASK COMPLETION FLAG, DRIVER SETS ZERO IF SUCCESSFUL
C      IC(8)           INSTRUCTION TO DRIVER TO TRANSFER USER INPUT DATA
C                      SET TO 0 FOR SUCCESSFUL TRANSFER
C      IC(9)           COUNT OF SUCCESSFUL CODE MODULE TASK COMPLETIONS
C      IC(10)          RESERVED
C
C      AC(1)           NAME OF CONTROL MODULE
C      AC(2)-AC(6)     NAMES OF CODE MODULES TO BE ACCESSED
C      AC(7)           RESERVED FOR FUTURE DRIVER CONTROL
C      AC(8)           LATEST INPUT DATA HEADER (NAME OF SPECIAL PROCESSOR)
C      AC(9)-AC(10)    RESERVED FOR FUTURE DRIVER CONTROL
C
C
       PROGRAM DRIVER
       REAL*8 AC,BLANK,CMODNM,C8,END
       REAL*8 HNCTL,RCD
       REAL*8 TITLE,FILEIN,FLN,GONOR,GLN,DDN,RSTKA,WSTKA,RSTKB,WSTKB
       LOGICAL LUNIT
       REAL*8 MODNAM
       INTEGER*2 IC,JP,JD
C
       COMMON/MEM/ MEMORY
       COMMON/CDATA/ AC(40), IC(80), JP(72), JD(48)
       COMMON/VCTRL/ HNCTL,RCD(100),ICD(100)
C
       COMMON/CINPT/TITLE(24),FILEIN(72),FLN(72,5),GONOR(100),GLN(72,5),
      *DDN(103),RSTKA,WSTKA,RSTKB,WSTKB,
      *NFLN(5),IX(15),IZ(201),NZ(900),NE(72),NVR(72,5),MVERS(100),
      *MEDUM(100),MMODD(100),NFNO,NMOD
C
       DIMENSION A(36000)
       DIMENSION C8(10)
C
       DATA BLANK/'          '/
       DATA END/'END'/
       DATA EQ/'='/
C
C
       IOINP = 1
       IOUT = 6
       IOFIVE = 5
       OPEN(IOINP, FILE='VENTURE.INP',BLANK='ZERO')
       OPEN(IOUT, FILE='CONDENS.OUT')
       OPEN(IOFIVE,FILE='VENTURE.TMP')
       WRITE(IOUT,1000)
```

```
      INAME = 0
      CALL VNAME(IOUT,INAME)
      INAME = 1
      DO 100 N = 1,40
      IC(N) = 0
      IC(N+40) = 0
      AC(N) = BLANK
  100 CONTINUE
      GO TO 103
  101 CONTINUE
      IC9 = IC(9)
      DO 102 N = 1,10
      IC(N) = 0
      AC(N) = BLANK
  102 CONTINUE
      IC(9) = IC9
  103 CONTINUE
      WRITE(IOUT,1005)
C
C     READ CONTROL MODULE NAME
      READ(IOINP,1001,END=115) X,CMODNM
      WRITE(IOUT,1006) CMODNM
      IF(X.NE.EQ) GO TO 114
      AC(1)= CMODNM
      REWIND IOFIVE
C
C     READ CONTROL MODULE DATA
  104 CONTINUE
      READ(IOINP,1004,END=113) C8
      WRITE(IOUT,1005) C8
      IF(C8(1).EQ.END) GO TO 105
      WRITE(IOFIVE,1004) C8
      GO TO 104
  105 CONTINUE
      ENDFILE IOFIVE
      REWIND IOFIVE
  106 CONTINUE
      IC(1) = IC(1)+1
      IF(IC(1).EQ.1) CALL GABY(IP,ID,IC,AC)
C*****************************************************************
C
C     ACCESS CONTROL MODULE HERE USING THE OPERATING SYSTEM LOADER,
C     COMMUNICATING THE COMMON DATA BLOCK CDATA.
C
C*****************************************************************
      IF(IC(1).LT.0) GO TO 111
      IF(IC(1).EQ.0) GO TO 101
      IC8 = IC(8)
      CALL VNAME(IOUT,INAME)
      DO 400 IMD=1,NMOD
      NZMOD = NZ(IMD)
      IF(NZ(IMD).LT.3.) THEN
C
```

```
C       READ INPUT OR SPECIAL PROCESSOR NAME.
        INQUIRE(UNIT=IOFIVE,OPENED=LUNIT)
        IF(.NOT.LUNIT) OPEN(IOFIVE,FILE='VENTURE.TMP')
        READ(IOINP,1004,END=110) C8
        WRITE(IOUT,1005) C8
        AC (8) = C8(1)
        MODNAM = C8(1)
        IF(IC8.EQ.2) AC(2) = C8(1)
        REWIND IOFIVE
C
C       READ INPUT OR SPECIAL PROCESSOR DATA
  107   CONTINUE
        READ(IOINP,1004,END=110) C8
        WRITE(IOUT,1005) C8
        IF (C8(1).EQ.END) GO TO 108
        WRITE(IOFIVE,1004) C8
        GO TO 107
  108   CONTINUE
        ENDFILE IOFIVE
        REWIND IOFIVE
        IC(8) = 0
  109   CONTINUE
        IC(7) = 0
        ENDIF
        GO TO (210,220,230,400,400,260,270,280,290,400,310,320,330,
       * 400,350,400,400,400,390,400) NZMOD
  210   CALL CLOSFI
        CALL INPROSER(A,MEMORY)
        GO TO 400
  220   CONTINUE
        GO TO 400
  230   CALL FILEDTOR
        GO TO 400
  260   CALL CLOSFI
        CALL CROSPROS(A,MEMORY)
        GO TO 400
  270   CLOSE(3)
        CLOSE(5)
        CLOSE(9)
        CLOSE(98)
        CALL CLOSFI
        CALL VENTNEUT(A,MEMORY)
        IF(IMD.NE.NMOD) THEN
        CALL CLOSFI
        CLOSE(23)
        CLOSE(24)
        CLOSE(27)
        CLOSE(28)
        CLOSE(40)
        OPEN(5,FILE='VENTURE.TMP',BLANK='ZERO')
        ENDIF
        GO TO 400
  280   CALL VALENEUT
```

```
      GO TO 400
  290 CALL CLOSFI
      CALL REACRATE(A,MEMORY)
      GO TO 400
  310 CALL VANCNEUT
      GO TO 400
  320 CALL CNTRODPO
      GO TO 400
  330 CALL CLOSFI
      CALL EXPOSURE(A,MEMORY)
      CLOSE(23)
      CLOSE(24)
      CLOSE(27)
      CLOSE(28)
      CLOSE(40)
      GO TO 400
  350 CALL PERTUBAT(A,MEMORY)
      GO TO 400
  390 CALL FUELMANG
  400 CONTINUE
C*********************************************************************
C
C     ACCESS CODE MODULES HERE USING THE OPERATION SYSTEM LOADER -
C     (MODULE NAMES ARE AC(2) THROUGH AC(6) UP TO A BLANK) ,
C     ADD 1 IC(9) FOR EACH SUCCESSFUL MODULE ACCESS,
C     IF THE RETURN STOP NUMBER FROM AN ACCESSED MODULE AC(N) EXCEEDS
C     THE ALLOWED VALUE IC(N), IC(7) IS SET TO THE RETURNED NUMBER AND
C     THE CONTROL MODULE IS ACCESSED WITHOUT FURTHER CODE MODULE
C     ACCESSES.
C
C*********************************************************************
      GO TO 115
  110 CONTINUE
      WRITE(IOUT,1008)
      GO TO 116
  111 CONTINUE
      WRITE(IOUT,1009)
      GO TO 116
  113 CONTINUE
      WRITE(IOUT,1007)
      GO TO 116
  114 CONTINUE
      WRITE(IOUT,1003)
      GO TO 116
  115 CONTINUE
      WRITE(IOUT,1002)
  116 CONTINUE
      CALL DELFIL
      STOP
C
 1000 FORMAT(' FORTRAN DRIVER FOR A MODULAR CODE SYSTEM FOR TESTING
     *(7-1-76)')
 1001 FORMAT(A1,A8)
```

```
 1002 FORMAT(' NORMAL END OF RUN - EOF IN INPUT STREAM')
 1003 FORMAT(' NO ''='' PRECEDING THE CONTROL MODULE NAME')
 1004 FORMAT(10A8)
 1005 FORMAT(10A8)
 1006 FORMAT(' CONTROL MODULE NAME = ',A8)
 1007 FORMAT(' END FILE READING CONTROL MODULE DATA')
 1008 FORMAT(/' END FILE ENCOUNTERED READING INPUT')
 1009 FORMAT(' DRIVER INSTRUCTED TO TERMINATE')
 1010 FORMAT(24I3)
 1011 FORMAT(A6)
      END
      SUBROUTINE VNAME(IOUT,INAME)
C     Printing the header page
      IF(INAME.EQ.0) THEN
      WRITE(*,100)
      RETURN
      ENDIF
      WRITE(IOUT,100)
  100 FORMAT(
     18X,'*************************************************************'/
     28X,'*                                                           *'/
     38X,'*                                                           *'/
     48X,'* V     V EEEEE N    N TTTTT U    U RRRR  EEEEE   PPPP  CCCCC *'/
     58X,'* V     V E     NN   N   T   U    U R   R E       P   P C   C *'/
     68X,'* V     V E     NNN  N   T   U    U R   R E       P   P C     *'/
     78X,'* V     V EEEE  N N  N   T   U    U RRRR  EEEE -- PPPP  C     *'/
     88X,'*  V   V  E     N  NNN   T   U    U R R   E       P     C     *'/
     98X,'*   V V   E     N   NN   T   U    U R  R  E       P     C   C *'/
     18X,'*    VV   EEEEE N    N   T   UUUUU R   R EEEEE   P     CCCCC *'/
     28X,'*                                                           *'/
     38X,'*                      VERSION 3.0                          *'/
     48X,'*                                                           *'/
     58X,'*                   DEVELOPED FOR THE                       *'/
     68X,'*                                                           *'/
     78X,'*            INEL REACTOR PHYSICS PC CODE SYSTEM            *'/
     88X,'*                                                           *'/
     98X,'*                        BY THE                             *'/
     18X,'*                                                           *'/
     28X,'*                NUCLEAR ENGINEERING PROGRAM                *'/
     38X,'*                  UNIVERSITY OF CINCINNATI                 *'/
     48X,'*          H. C. HURIA, A. SHAPIRO, AND K. W. CHO           *'/
     58X,'*              (Under Subcontract C87-101212)               *'/
     68X,'*                                                           *'/
     78X,'*                                                           *'/
     88X,'*************************************************************')
  110 FORMAT(1H1)
      WRITE(IOUT,110)
      RETURN
      END
CGABY CONTROL MODULE ENTRANCE ROUTINE, TYPICAL                        GABY  10
C                                                                     GABY  20
      SUBROUTINE GABY (IP, ID, IB, AB)                                GABY  30
C                                                                     GABY  40
```

```
      INTEGER*2 IC,JP,JD,IB,IP,ID                                    GABY  50
      REAL*8 AC,AB                                                   GABY  60
      COMMON/MEM/ MEMORY
      COMMON /CDATA/ AC(40), IC(80), JP(72), JD(48)                  GABY  80
      DIMENSION IP(1), ID(1), IB(1), AB(1)                           GABY 100
C                                                                    GABY 120
      DO 100 I=1,40                                                  GABY 130
      AC(I) = AB(I)                                                  GABY 140
      IC(I) = IB(I)                                                  GABY 150
      IC(I+40) = IB(I+40)                                            GABY 160
  100 CONTINUE                                                       GABY 170
      DO 110 I=1,72                                                  GABY 180
      JP(I) = IP(I)                                                  GABY 190
  110 CONTINUE                                                       GABY 200
      DO 120 I=1,48                                                  GABY 210
      JD(I) = ID(I)                                                  GABY 220
  120 CONTINUE                                                       GABY 230
C                                                                    GABY 240
C     GO TO THE MAIN PROGRAM OF THE CONTROL MODULE                   GABY 250
      CALL  CONTROL1
   GABY 260
C                                                                    GABY 270
C     RETURN ROUTE FROM THE MAIN PROGRAM                             GABY 280
      DO 130 I=1,40                                                  GABY 290
      AB(I) = AC(I)                                                  GABY 300
      IB(I) = IC(I)                                                  GABY 310
      IB(I+40) = IC(I+40)                                            GABY 320
  130 CONTINUE                                                       GABY 330
C                                                                    GABY 340
C     RETURN TO THE DRIVER                                           GABY 350
      RETURN                                                         GABY 360
      END                                                            GABY 370
CC
```

APPENDIX IV

VENTURE SUBROUTINES
(from reference 5)

THE ACCESS, CONTROL, AND GENERAL PURPOSE ROUTINES

MAIN	ENTRY POINT TO NEUTRONICS CODE BLOCK. CALLS ERRSET, TIMER, DOPC, IONO, VENT, DRIV
IONO	ASSIGNS INPUT/OUTPUT UNIT NUMBERS
VENT	ACCESSES CODE BLOCK CONTROL INFORMATION CALLS SKER, FERR
DRIV	PASSES INFORMATION TO THE CONTROLLER ROUTINE ALLOCATES CORE STORAGE CALLS GETCOR, ROXX, ROXY, DIFF, DOPC, FRECOR
DIFF	CONTROLS THE CALCULATION CALLS CORE, MAC1, CON1, PHIA, ORLX, COMC, LCAL, FLXR, FXSR, BSQV, AJNT, PROS, DOPC, OUTR, DSDF, DCID, DIMS, AJDS, FLRD, ADN1, EDIT, SAV1, PERT, JERT, FERR, TIMER
CORE	DETERMINES STORAGE REQUIREMENTS AND DATA HANDLING MODES. CALLS CORI,CORP, GNAM, CORD, CORB, DDSP, DASU, SKER, JPRT, FERR.
DASU	SETUP DIRECT ACCESS FILES
EASU	SETUP DIRECT ACCESS FILES CALLS DOPC, FERR
TIMER*	SERVICE ROUTINE FOR COMPUTER TIME, ETC.
STOR	SERVICE ROUTINE FOR MOVING DATA IN MAIN MEMORY
SKER	FILE MANAGEMENT RELATED ERROR MESSAGES
FERR	ALL OTHER FATAL ERROR MESSAGES
KEEP*	DUMMY ROUTINE USED TO OUTFOX THE OPTIMIZING COMPILER

THE INPUT/OUTPUT ROUTINES

DOPC	INITIALIZES, OPENS, AND CLOSES DATA FILES ENTRY ROXY COMMUNICATES DATA ARRAYS CALLS SEEK, RITE, DEFILE, CLOSDA, (FBSAM AND ENTRIES)
RITE	DATA TRANSFER MANAGER AND WRITES DATA (FORTRAN WRITE) -CALLED BY MOST ROUTINES. ENTRY REED READS DATA (FORTRAN READ) - CALLED BY MOST ROUTINES ENTRY ROXX* COMMUNICATES DATA ARRAYS CALLS CRIT, CRED, (FBSAM, AND ENTRIES)
SEEK	INTERFACE DATA FILES MANAGER CALLS RITE, REED
CRIT*	ASSEMBLY LANGUAGE ROUTINE FOR CORE TO EXTENDED CORE DATA TRANSFER (SEE SECTION 203 FOR THE FORTRAN EQUIVALENT) ENTRY CRED EXTENDED CORE TO CORE DATA TRANSFER
DEFILE*	ASSEMBLY LANGUAGE ROUTINE TO EXECUTE THE FORTRAN DEFINE FILE TATEMENT USING PROBLEM DEPENDENT VARIABLES (OPENS DIRECT ACCESS FILES) - ACCESSES SYSTEM ROUTINE IHCEDIOS
CLOSDA*	ASSEMBLY LANGUAGE ROUTINE TO CLOSE DIRECT ACCESS FILES
FBSAM*	LOCAL I/O ROUTINE USED ALONG WITH THE I/O PACKAGE TO PRODUCE SPECIAL CAPABILITY

THE CALCULATION OF MACROSCOPIC CROSS SECTIONS

MAC1	CONTROLS MACROSCOPIC CROSS SECTION CALCULATION CALLS MACA, MACB, MAC2, SCAL, MAC3, MAC5, CHDM, MAC4, MAC6, SKER, FERR
MACA	INITIAL PROCESSING OF GRUPXS CALLS STOR
MACB	CHECK NAMES AND CLASSES ON NDXSRF AND GRUPXS FOR AGREEMENT
MAC2	CALCULATE MACROSCOPIC PRINCIPAL CROSS SECTIONS
SCAL	LOCATES POSITION OF SCATTERING RECORDS ON GRUPXS
MAC3	CALCULATE MACROSCOPIC SCATTERING CROSS SECTIONS
MAC5	ADJUST DIFFUSION CONSTANT AND SCATTERING DATA FOR P1 CALCULATION.
CHDM	CHECK DIMENSION SEARCH DATA CALLS SKER
MAC4	CALCULATE MACROSCOPIC SEARCH DATA
MAC6	EDIT MACROSCOPIC CROSS SECTIONS

THE CALCULATION OF EQUATION CONSTANTS

CON1	CONTROLS EQUATION CONSTANT CALCULATION CALLS MSHO,NRDF,MSH1,CON2, GEOQ, CON3, MSH3, CKCT, CON4, CON5, CON7,CON9, STOR, SKER,FERR
MSHO	SETUP COARSE MESH PARAMETERS FOR 1D AND 2D CASES
NRCF	CONVERT REGION ASSIGNMENTS BY COARSE MESH TO FINE MESH
MSH1	CALCULATE FINE MESH DISTANCES
CON2	SETUP BOUNDARY CONSTANTS AND BUCKLING
GEOQ	CHANGE FROM 3D YO 2D CASE
CON3	RESTRUCTURE MACROSCOPIC DATA AND ZERO ROD CROSS SECTIONS CALLS NROD, STOR
MSH3	EDIT FINE MESH DISTANCES
CKCT	SETUP INDEXING FOR DIFFUSION CONSTANTS
CON4	CALCULATES LEAKAGE CONSTANTS CALLS NROD, BNDY
CON5	CALCULATES LEAKAGE CONSTANTS (TRIAGONAL)
CON7	CALCULATES LEAKAGE CONSTANTS (HEXAGONAL)
CON9	CALCULATES ZONE VOLUMES FROM REGION VOLUMES AND DETERMINE ZONE WITH MAXIMUM NU-SIG-VOL CALLS NROD
NROD	FUNCTION TO DETERMINE INTERNAL BLACK ABSORBER ZONES
BNDY	FUNCTION TO CALCULATE NON-RETURN LEAKAGE CONSTANT

THE INITIALIZATION PROCESS

ORLX	CONTROLS ITERATIVE PROCESS PARAMETER INITIALIZATION CALLS ORLA,ORLB, ORLC, ORLD, ORLE, ORLF,BATG, ORLR, CONG, FERR, RCOV
ORLA	LOCATES A REFERENCE POINT IN MESH TO USE AS A BASIS FOR INITIALIZATION PROCEDURES (2, 3-D PROBLEMS ONLY) CALLS KEEP
ORLB	DETERMINES AN ENERGY DISTRIBUTION FUNCTION FROM EQUATION ONSTANTS AT THE REFERENCE POINT (2, 3-D PROBLEMS ONLY) CALLS KEEP
ORLC	SETUP 1-D EQUATION CONSTANTS ALONG THE ROW CONTAINING THE EFERENCE POINT (2, 3-D PROBLEMS ONLY) CALLS KEEP
ORLD	SETUP DATA FOR THE 1-D INITIALIZATION CALCULATION (2, 3-D PROBLEMS ONLY) CALLS NROD
ORLE	SETUP CROSS SECTIONS FOR THE I-D INITIALIZATION CALCULATION (2, 3-D PROBLEMS ONLY)
ORLF	SOLVES THE 1-D PROBLEM FOR INNER AND OUTER ITERATION BEHAVIOR (2, 3-D PROBLEMS ONLY) CALLS LAXR, LAXP
BATG	CALCULATES OVERRELAXATION COEFFICIENTS AND INNER ITERATIONS AND HEBYSHEV PARAMETER AND SETS DEFAULT OPTIONS (2, 3-D PROBLEMS ONLY) CALLS LUCK
ORLR	BYPASS INITIALIZATION DURING SEARCH OR PERTURBATION ONLY CALCULATIONS (2, 3-D PROBLEMS ONLY)
RCOV	RECOVERS DATA FOR SUCCESSIVE NEUTRONICS PROBLEMS
LAXR	LINE RELAXATION FOR 1-D INITIALIZATION PROBLEM
LAXP	POINT RELAXATION FOR 1-D INITIALIZATION CALCULATION
MUCK	FUNCTION TO LOCATE REFLECTED BOUNDARY
LUCK	FUNCTION TO DETERMINE MESH DEPENDENT PARAMETER FOR LAMDA
CONG	PREPARE MACROSCOPIC CROSS SECTIONS AND OTHER DATA FOR ITERATIVE PROCESS
PHIA	CONTROLS FLUX INITIALIZATION CALLS PHI1, PHI7, PHI2, PHI3, FERR
PHI1	INITIAL FLUX IS CONSTANT CALLS NROD
PHI7	INITIAL FLUX IS SYNTHESIZED FROM THE RESULT OF THE 1-D NITIALIZATION CALCULATION (2, 3-D PROBLEMS ONLY) CALLS SDBN, TOIP, NROD
TOIP	SIMPLE LINEAR INTERPOLATION
PHI2	INITIAL FLUX IS AFUNCTION OF SPACE AND ENERGY CALLS EDBN, SDBN, NROD
EDBN	CALCULATE ENERGY DISTRIBUTION FUNCTION

SDBN	CALCULATE SPATIAL DISTRIBUTION FUNCTIONS
PHI3	PROCESS INITIAL FLUX FROM FLUX INTERFACE (MAY BE EXPANDED TO NEW MESH EXCEPT FOR HEXAGONAL GEOMETRY) CALLS PHI4, PHI5, PHI6, GRIP, PAN1, PAN2, NROD, SKER
PHI4	ID FLUX EXPANSION
PHI5	2D FLUX EXPANSION CALLS PBND, PC2D
PHI6	3-D FLUX EXPANSION CALLS PBND, PC2D, PC3D
GRXP	GROUP EXPANSION
PAN1	TRIANGULAR EXPANSION ON PLANES
PAN2	TRIANGULAR EXPANSION BETWEEN PLANES
PBND	FUNCTION TO DETERMINE ARTIFICIAL FLUX POINT
PC2D	FUNCTION TO DETERMINE ARTIFICIAL CORNER POINT - 2D
PC3D	FUNCTION TO DETERMINE ARTIFICIAL CORNER POINT - 3D CALLS PC2D

THE ITERATIVE PROCESS

COMC	UTILITY SUBROUTINE
LCAL	CALCULATES STARTING ADDRESSES IN DATA ARRAY CALLS FERR
FLXR	OBTAINS INITIAL FLUX
FISR	OBTAINS A FIXED SOURCE CALLS SKER
BSQV	SEARCH CALCULATION UTILITY ROUTINE
AJNT	SETS UP INPUT/OUTPUT FILES FOR THE ADJOINT PROBLEM CALLS REV1
REV1	PROCESSES SCATTERING DATA FOR ADJOINT PROBLEM
PROS	SETS UP INPUT OUTPUT FILES CALLS ZIO3, FEFS
ZIO3	PROCESSES PRINCIPAL CROSS SECTIONS
FEFS	SETS INITIAL FLUX TO FIXED SOURCE WHEN FIXED SOURCE LT 0
DSDF	CALCULATES INDIRECT NUCLIDE SEARCH CHANGE EIGENVALUE
DCID	CONTROLS SEARCH CALCULATION EXIT OPTIONS
DIMS	CALCULATES DIMENSION SEARCH CHANGE FACTOR
ADJS	CONTROLS DIMENSION SEARCH CHANGES CALLS DIM1, DIM2, DIM3
DIM1	READS COARSE MESH MODIFIERS FROM SEARCH INTERFACE FILE
DIM2	CONTROLS COARSE MESH AND VOLUME CHANGES - WRITES NEW GEODST CALLS CMES, CRGV
DIM3	CONTROLS CHANGE ZONE VOLUMES - WRITES NEW NDXSRF INTERFACE CALLS ZVRV
ZVRV	CHANGES ZONE VOLUMES
FLRD	READS GEODST FOR FINAL EDIT OF MESH - DIMENSION SEARCH CALLS FLMH
CHES	CHANGES COARSE MESH
CRGV	CALCULATES REGION VOLUMES FROM POINT VOLUMES CALLS CHVL

CHVL	CHANGES REGION VOLUMES
FLMH	EDITS FINAL MESH - DIMENSION SEARCH
OUTR	OUTER ITERATION CONTROLLER CALLS DOIN, ZINS, FSOR, SSOR, FLUX, PSOR, JUSB, BALC, ITRP, WRES, PREC, MUEX, CHEV, ETR1, ETR2, ATED, SGDA, FERR
BALC	NEUTRON BALANCE EQUATION
ZINS	CALCULATES THE DIRECT SEARCH PROBLEM EIGENVALUE
CHBF	CHEBYSHEV ACCELERATION ROUTINE
CHEV	CHEBYSHEV ACCELERATION ROUTINE
RDAB	CALCULATES ROD ABSORPTIONS
LTRG	CALCULATES IN-LEAKAGE FOR TRINAGULAR GEOMETRY
ITRP	ASSESSES FLUX CONVERGENCE CALLS FFGG, BHAV
BHAV	CALCULATES ITERATIVE CONVERGENCE PARAMETERS
JUSB	OVERRELAXATION COEFFICIENT CONTROL
ATED	EDITS ITERATION DATA
FFGG	CALCULATES FLUX EXTRAPOLATION FACTORS
RDUE	RESIDUE ESTIMATE OF THE MULTIPLICATION FACTOR
RELX	SOLVES FOR THE FLUX VALUES ALONG A ROW AND OVERRELAXES THEM
OELX	SOLVES FOR THE FLUX VALUES ALONG A ROW NO OVERRELAXATION
NEWB	CALCULATES NEW OVERRELAXATION FACTORS
FSOR	FISSION SOURCE CALCULATION CONTROLLER CALLS FOU1, FOU2, FOU3, FOU4, FOU5, FOU6
SSOR	SCATTERING SOURCE CALCULATION CONTROL CALLS SOU1, SOU2, SOU3, SOU4, SOU5, SOU6
PSOR	P-1 SCATTERING SOURCE CALCULATION CONTROL CALLS POU1, POU2, POU3, POU4, POU5
FLUX	INNER ITERATION CONTROL CALLS INR1, INR2, INR3, INR4, INR5, INR6, INRX, BHAV
MUEX	EXTRAPOPLATION PARAMETER PROCESSING
ETR1	SINGLE ERROR MODE FLUX EXTRAPOLATION
ETR2	DOUBLE ERROR MODE FLUX EXTRAPOLATION
SGDA	SAVES AND RETRIEVES DATA DURING DIRECT NUCLIDE SEARCH
DOIN	FLUX CALCULATION UTILITY ROUTINE
RRES	READS RESTART FILE
WRES	WRITES RESTART FILE
PREC	CALCULATES ONE-DIMENSIONAL SWEEP PARAMETERS
ADN1	CONTROLLER FOR UPDATING ATOMIC DENSITIES
ADN2	UPDATES ATOMIC DENSITIES
ADN3	EDITS ATOMIC DENSITIES
INR1	INNER ITERATION CONTROL (1 ROW STORED MODE) CALLS LOU1, RDUE, RELX, LEK1, CHEV, OELX, NEWB
LOU1	IN-LEAKAGE CALCULATION
FOU1	FISSION SOURCE CALCULATION
SOU1	SCATTERING SOURCE CALCULATION
POU1	P-1 SCATTERING SOURCE CALCULATION
LEK1	OUT-LEAKAGE CALCULATION

INR2	INNER ITERATION CONTROL (ALL DATA STORED MODE) CALLS LOU2, RDUE, RELX, LEK2, CHEV, RDAB, OELX, NEWB
FOU2	FISSION SOURCE CALCULATION
LOU2	IN-LEAKAGE CALCULATION
SOU2	SCATTERING SOURCE CALCULATION
POU2	P-1 SCATTERING SOURCE CALCULATION
LEK2	OUT-LEAKAGE CALCULATION
INR3	INNER ITERATION CONTROL (SPACE PROBLEM DATA STORED MODE) CALLS LOU3, RDUE, RELX, LEK3, CHEV, LTRG, RDAB, OELX, NEWB
LOU3	IN-LEAKAGE CALCULATION
FOU3	FISSION SOURCE CALCULATION
SOU3	SCATTERING SOURCE CALCULATION
POU3	P-1 SCATTERING SOURCE CALCULATION
LEK3	OUT-LEAKAGE CALCULATION
INR4	INNER ITERATION CONTROL (MULTIPLE PLANE DATA STORED MODE) CALLS LOU4, QDUE, QELX, LEK4, SOUX, J1C4, CHEV, LTRG, RDAB, NEWB
LOU4	IN-LEAKAGE CALCULATION
LTRG	SPECIAL IN-LEAKAGE CALCULATION FOR TRIANGULAR GEOMETRY
FOU4	FISSION SOURCE CALCULATION
SOU4	SCATTERING SOURCE CALCULATION
POU4	P-1 SCATTERING SOURCE CALCULATION
LEK4	OUT-LEAKAGE CALCULATION
J1C4	DEL DOT J CALCULATION
QDUE	ACCESSES RESIDUE CALCULATION CALLS RDUE
QELX	ACCESSES FLUC CALCULATION CALLS RELX, OELX
INR5	INNER ITERATION CONTROL (MULTI-ROW STORED MODE) CALLS LOU5, RDUE, RELX, LEK5, J1C5, CHEV, OELX, NEWB
LOU5	IN-LEAKAGE CALCULATION
FOU5	FISSION SOURCE CALCULATION
SOU5	SCATTERING SOURCE CALCULATION
POU5	P-1 SCATTERING SOURCE CALCULATION
J1C5	DEL DOT J CALCULATION
LEK5	OUT-LEAKAGE CALCULATION
INR6	CONTROLLER ROUTINE FOR THE SPECIAL 1-D PROCEDURE CALLS CHEV
FOU6	FISSION SOURCE CALCULATION
SOU6	SCATTERING SOURCE CALCULATION
DELX	LINE RELAXATION WITHOUT OVERRELAXATION
INRX	INNER ITERATION CONTROL (MULTI-LEVEL DATA TRANSFER MODE) CALLS LOUX, RDUE, RELX, LEKX, SOUZ, J1CX, CHEV, RDAB, OELX, NEWB

LOUX	IN-LEAKAGE CALCULATION
FOUX	FISSION SOURCE CALCULATION
SOUY	SCATTERING SOURCE CALCULATION
SOUZ	SCATTERING SOURCE CALCULATION
POUX	P-1 SCATTERING CALCULATION
LEKX	OUT-LEAKAGE CALCULATION
J1CX	DEL DOT J CALCULATION

THE EDIT ROUTINES

EDIT	CONTROLS EDITS CALLS NBAL, PNDN, FLXW, PERT, BSQS, FISS, JINT, PTVL, PTZF
CORP	EDIT PROBLEM DESCRIPTION
GNAM	EDIT GEOMETRY AND CHECK FOR VALIDTY
CORD	EDIT MAJOR PROBLEM PARAMETERS
JPRT	CALCULATES PERTURBATION STORAGE REQUIREMENTS
CORB	EDIT BOUNDARY INDICATORS AND CHECK FOR VALIDTY
DDSP	EDIT SYMBOLIC PARAMETERS FOR DISK SPACE (IBM 360 JCL)
JINT	CALCULATES AND EDITS ADJOINT ZONE FLUX RESULTS
POUT	PRINTS FLUX, POWER DENSITY, NEUTRON DENSITY
NBAL	PRINTS NEUTRON BALANCE CALLS SOBL, SKER
SOBL	CALCULATES NEUTRON BALANCE SCATTERING DATA
FISS	WRITES FISSION SOURCE INTERFACE (FISSOR) CALLS SKER
BSQS	CALCULATES BUCKLING IN 3-D PROBLEMS
PNDN	CALCULATES POWER AND NEUTRON DENSITY CALLS POUT, SKER
PTVL	GETS DATA FOR SUBROUTINE PTZP
PTZP	WRITES FLUX FOR 2 ZONES ON RZFLUX FOR DEPLETION
FLXW	WRITES FLUX INTERFACE DATA FILE CALLS POUT, SKER
SAV1	SPECIAL DATA OUTPUT IN BCD FORM CALLS SAV2, SAV4, SAV6
SAV2	SPECIAL DATA OUTPUT IN BCD FORM (GEODST) CALLS SAV3
SAV3	SPECIAL DATA OUTPUT IN BCD FORM (GEODST)
SAV4	SPECIAL DATA OUTPUT IN BCD FORM (PWDINT) CALLS SAV5
SAV5	SPECIAL DATA OUTPUT IN BCD FORM (PWDINT)
SAV6	SPECIAL DATA OUTPUT IN BCD FORM (RTFLUX) CALLS SAV7
SAV7	SPECIAL DATA OUTPUT IN BCD FORM (RTFLUX)

THE PERTURBATION ROUTINES

QOUT	EDITS SPACE POINT IMPORTANCE MAPS
RTUB	WRITES INTERFACE FILE PERTURB CALLS SKER
MRPT	CALCULATES CHANGE IN KEFF DUE TO SIGMAS
PERO	EDITS PERTURBATION INTEGRALS
BBB2	PERTURBATION UTILITY ROUTINE
BBB1	PERTURBATION UTILITY ROUTINE (FOR ALL EXCEPT ONE ROW STORED MODE)
JBET	PERTURBATION CONTROL CALLS JAFA, JIFE, JOFY, PERO, JAPS, RTUB, QOUT, SKER, JGET
JAFA	SETS UP INPUT/OUTPUT FILES FOR PERTURBATION CALCULATION
JIFE	CALCULATES BASIC PERTURBATION INTEGRALS
JOFY	CALCULATES TRANSPORT PERTURBATION INTEGRALS CALLS BBB1,BBB2
JAPS	CALCULATES SPACE POINT IMPORTANCE MAPS CALLS JMAP
JMAP	CONTROLS EDIT OF IMPORTANCE MAPS CALLS QOUT
JGET	GETS DCONS (FOR 1-ROW STORED MODE)
PERT	PERTURBATION CONTROL CALLS DAFA, LIFE, TUFY, PERO, MAPS, RTUB, QOUT, SKER
DAFA	SETS UP INPUT/OUTPUT FILES FOR PERTURBATION INTEGRALS
LIFE	CALCULATES BASIC PERTURBATION INTEGRALS
TUFY	CALCULATES TRANSPORT PERTURBATION INTEGRALS CALLS BBB1, BBB2
MAPS	CALCULATES SPACE POINT IMPORTANCE MAPS CALLS PMAP
PMAP	CONTROLS EDIT OF IMPORTANCE MAPS CALLS QOUT

SPECIAL ROUTINES*

GETCORE	ASSEMBLY LANGUAGE ROUTINE TO ALLOCATE CORE DYNAMICALLY FOR THE VARIABLY DIMENSIONED ARRAYS AT RUN TIME
FRECORE	ASSEMBLY LANGUAGE ROUTINE TO FREE CORE ALLOCATED BY GETCORE
ERRSET	SUPPLIES THE LEVEL OF ERROR STOPS TO THE SYSTEM

LABELED COMMON BLOCKS

CNTRL
VCTRL
MGMTIO
IOUNT
AFLUY
AOSUB
LIMITS
ADRES
FSWAP
DEASU
COMSAM
USRID

* NOT USED BY VENTURE\PC

APPENDIX III.

BURNER SUBROUTINES

(FROM REF.8)

BURNER SUBROUTINE DESCRIPTION

ANOR	DETERMINE NORM OF MATRIX
ARRI	SUM INVENTORY AND REACTION RATES (ABSORPTION, FISSION, PRODUCTION, AND CAPTURE (N,G)) BY ABSOLUTE NUCLIDE
AUXE	WRITE CONDENSED EDIT
BFIX	NORMAL EXPOSURE CALCULATION
BGXS	PROCESS NEXT-TO-LATEST GRUPXS
BINP	INITIAL INTERFACE PROCESSING (NDXSRF, GEODST, GRUPXS, AND EXPOSE) AND DATA PREPARATION
BMOV	CONTINUOUS FUELING EXPOSURE CALCULATION
BPIA	CONTROL GEOMETRY (GEODST) AND POINT FLUX RTFLUX) PROCESSING FOR POINT CALCULATION (METHOD 1)
BPIB	CONTROL POINT FLUX (RZFLUX-MODIFIED) PROCESSING FOR POINT CALCULATION (METHOD 2)
BPIC	INITIAL DENSITY PREPARATION, COMPUTE REACTION RATES AND SETUP STORAGE FOR POINT EXPOSURE AND SHUTDOWN CALCULATION
BPIN	CONTROL SETUP FOR POINT EXPOSURE AND SHUTDOWN
BRCI	OBTAIN EXPOSURE CONTROL INFORMATION FROM INTERFACE CONTROL
BRDS	SETUP DYNAMIC DATA STORAGE SPACE
BRNA	COMPUTE SPECIFIC REACTION RATES FOR ABSORPTION, FISSION, NU-FISSION, (N,G), (N,A), (N,P), (N,2N), (N,D), AND (N,T)
BRND	EDIT SPECIFIC REACTION RATES
BRNF	SETUP INTERNAL CROSS-REFERENCING INFORMATION FOR ABSOLUTE NUCLIDE, NUCLIDE CLASS, AND ZONE CLASS
BRNO	PREPARE AND EDIT FINAL SUMMARY TABLE
BRNS	DETERMINE STORAGE REQUIRED AND MODE OF SOLUTION AND INITIALIZE DIRECT ACCESS UNITS IF NEEDED
BRNT	PRE-WRITE DIRECT ACCESS UNITS IF NEEDED
BRNW	EDIT CONTENTS OF EXPOSE FILE - CHECKS DECAY, YIELD, AND MATRIX DATA FOR ERRORS
BRNX	SETUP DECAY CONSTANTS AND CORRESPONDENCE BETWEEN DENSITY AND EXPOSURE DATA
BRNY	EDIT ATOM DENSITIES
BRNZ	PROCESS ZNATDN AND WRITE INITIAL DENSITIES ON SCRATCH ONE ZONE/SUBZONE AT A TIME
BRN1	OVERALL CALCULATION CONTROL
BRN3	PROCESS RZFLUX AND WRITE ZONE AVERAGE FLUX ON SCRATCH ONE GROUP AT A TIME - PERFORM INITIAL POWER ADJUSTMENT
BRN4	CHECK NUCLIDE NAMES AND CLASSES FRPM 2 SOURCES
BRN7	COPY PRINCIPAL CROSS SECTIONS FROM GRUPXS TO SCRATCH

BRPF	COMPUTE SPECIFIC REACTION RATE FOR FISSION IN ENERGY RANGES OF FIELD DATA FOR POINT CALCULATION
BRRF	COMPUTE SPECIFIC REACTION RATE FOR FISSION IN ENERGY RANGES OF YIELD DATA
BURN	CONTROLS EXPOSURE AND SHUTDOWN CALCULATION
BZIN	ADDITIONAL INTERFACE PROCESSING (RZFLUX AND ZNATDN) AND COMPUTE REACTION RATES AND SETUP STORAGE FOR EXPOSURE AND SHUTDOWN CALCULATION
BZT1	DETERMINE IF ZNTEMP EXISTS AND CHECK INPUT DATA
BZT2	PROCESS TEMPERATURES FROM ZNTEMP
CHEK	DEBUG FLUX CHECK FOR POINT CALCULATION (METHOD 1)
CMOV	CHECK NUCLIDE SET REFERENCES FOR CONTINUOUS FUELING MODEL
CMPH	COMPARE 2 HOLLERITH ARRAYS
CMPI	COMPARE 2 INTEGER ARRAYS
CPH1	COPY ONE SET OF EXPOHT DATA FROM ONE UNIT TO ANOTHER
CPH2	EDIT ONE SET OF EXPOHT DATA
DEEF	SETUP AND CHECK INPUT PARAMETERS FOR CONTINUOUS FUELING MODEL
DOEX	EXPOSURE BY VARIOUS METHODS
DOPC	SCRATCH FILE DATA TRANSFER MANAGEMENT FOR SPECIAL ACCESS METHODS (NOT SEQUENTIAL)
DOSH	SHUTDOWN BY VARIOUS METHODS
DOWN	SHUTDOWN CALCULATION
ECHK	CHECK NEUTRON ENERGY GROUP STRUCTURE
EDED	EDIT SECONDARY ENERGY DEPOSITION DATA FROM EXPOSE
EDEP	SETUP FOR SECONDARY ENERGY DEPOSITION EDITS
EPFD	SET DEFAULT VALUE FOR ENERGY/FISSION AND ENERGY/CAPTURE IF NECESSARY
EPH2	EDIT MAXIMUMS AND SYSTEM TOTALS OF EXPOHT DATA
ESET	DETERMINE WHICH ENERGY GROUP NUMBER IS CUTOFF AND FRACTIONAL PART FOR FLUENCE CALCULATION
ETAB	CALCUL;ATE AND EDIT SECONDARY ENERGY DEPOSITION
EXPH	SETUP AND CONTROL FOR WRITING INTERFACE EXPOHT
FERR	WRITE FATAL ERROR MESSAGE AND STOP
FLUC	FUNCTION TO DETERMINE (FLUX)*(EXPOSURE TIME) CONSTANT
FLUE	SUM ZONE FLUX OVER RANGE OF GROUPS SPECIFIED
FOUL	EDIT MONITORING INFORMATION
GCHK	CHECK FOR IMPLEMENTED GEOMETRY FOR POINT CALCULATION (METHOD 1)
GNZC	OBTAIN ZONE CLASSES FROM GEODST
HQUE	CHECK FOR UNIQUENESS IN LIST OF HOLLERITH NAMES
ISTR	FUNCTION TO ASSIGN A REAL VARIABLE TO AN INTEGER VARIABLE LOCATION WITHOUT TYPE CONVERSION

IX2D	FUNCTION TO DETERMINE SUBSCRIPTS OF A TWO DIMENSIONAL ARRAY, GIVEN DIMENSIONS AND POSITION IN ARRAY
IX3D	FUNCTION TO DETERMINE SUBSCRIPTS OF A THREE DIMENSIONAL ARRAY, GIVEN DIMENSIONS AND POSITION IN ARRAY
JAGY	AVERAGE GENERATION RATE SOLUTION FOR EXPOSURE
JAOD	SETUP OFF-DIAGONAL MATRIX ELEMENTS FOR MATRIX EXPONENTIAL AND AVERAGE GENERATION RATE SOLUTIONS (EXPOSURE)
JENY	SETUP MATRIX EXPONENTIAL SOLUTION FOR EXPOSURE
JUCY	EXPLICIT CHAIN SOLUTION FOR EXPOSURE
LAGY	AVERAGE GENERATION RATE SOLUTION FOR SHUTDOWN
LAOD	SETUP OFF-DIAGONAL MATRIX ELEMENTS FOE MATRIX EXPONENTIAL AND AVERAGE GENERATION RATE SOLUTIONS (SHUTDOWN)
LEGP	FUNCTION TO COMPARE (LT,EQ,GT) TWO REAL NUMBERS WITHIN EPSILON
LEMY	SETUP MATRIX EXPONENTIAL SOLUTION FOR SHUTDOWN
LUCY	EXPLICIT CHAIN SOLUTION FOR SHUTDOWN
MAIN	INITIALIZE INPUT/OUTPUT UNITS
MEIT	MATRIX EXPONENTIAL SOLUTION
MEMA	MATRIX EXPONENTIAL - ELIMINATE NUCLIDES ASSUMED TO BE IN EQUILIBRIUM
MEPA	MATRIX EXPONENTIAL - COMPUTE DENSITIES FOR NUCLIDES IN EQUILIBRIUM
MESA	MATRIX EXPONENTIAL 1 TERM METHOD
METS	MATRIX EXPONENTIAL - TRANSPOSE MATRIX ELEMENTS
MNRP	LOCATE SMALLEST POSITIVE VALUE IN AN ARRAY
MSHK	CHECK COARSE MESH DATA FROM GEODST FOR POINT CALCULATION (METHOD 1)
MSHO	SETUP COARSE MESH PARAMETERS FOR 1-D AND 2-D GEOMETRIES FOR POINT CALCULATION (METHOD 1)
MSH1	CALCULATE FINE MESH DISTANCES FOR POINT CALCULATION (METHOD 1)
MSH3	EDIT FINE MESH SPACING FOR POINT CALCULATION (METHOD 1)
NXRP	LOCATE LARGEST POSITIVE VALUE IN AN ARRAY
NRCP	CONVERT REGION ASSIGNMENTS FOR COARSE MESH INTERVALS TO REGION ASSIGNMENTS FOR FINE MESH INTERVALS FOR POINT CALCULATION (METHOD 1)
OEXP	EXPOSURE CALCULATIONS (FOR OVERLAY CONVENIENCE)
OFIX	NORMAL EXPOSURE CALCULATION (FOR OVERLAY CONVENIENCE)

OMOV	CONTINUOUS FUELING EXPOSURE (FOR OVERLAY CONVENIENCE)
OOWN	SHUTDOWN CALCULATION (FOR OVERLAY CONVENIENCE)
PARI	EDIT START AND END OF STEP INVENTORY AND REACTION RATES BY ABSOLUTE NUCLIDE
PDPT	CALCULATE POWER DENSITY
PDST	POWER DENSITY STATISTICS FOR POINT CALCULATION
PFIX	POINT EXPOSURE CALCULATION
PGEO	PROCESS GEODST GEOMETRY FILE FOR POINT CALCULATION (METHOD 1)
PLOC	LOCATE POINTS WITHIN SELECTED ZONES AND COMPUTE POINT VOLUMES FOR POINT CALCULATION (METHOD 1)
PNAW	WRITE POINT NUCLIDE DENSITIES ON INTERFACE FILE PTATDN FOR POINT CALCULATION
PONI	EDIT FEED AND DISCHARGE RATES IN KG/DAY
POWL	ACCUMULATE POWER AND LOCATE MAXIMUM POWER DENSITY
POWN	POINT SHUTDOWN CALCULATION
POWP	ACCUMULATE POWER ALONG PATH FOR CONTINUOS FUELING MODEL
PPOE	EDIT POWER, ACTINIDE FEED RATE, AND EXPOSURE BY ZONE PATH AND SUBZONE PATH FOR CONTINUOUS FUELING MODEL
PRNA	COMPUTE SPECIFIC REACTION RATES FOR ABSORPTION , FISSION, NU*FISSION, (N,G), (N,A), (N,2N), (N,D), AND (N,T) FOR POINT CALCULATION
PRND	EDIT SPECIFIC REACTION RATES FOR POINT CALCULATION
PRNS	DETERMINE STORAGE REQUIRED AND MODE OF SOLUTION AND INITIALIZE DIRECT ACCESS UNITS IF NEEDED FOR POINT CALCULATION
PRNT	PRE-WRITE DIRECT ACCESS UNITS IF NEEDED FOR POINT CALCULATION
PRNY	EDIT ATOM DENSITIES FOR POINT CALCULATION
PRNZ	SETUP INITIAL DENSITIES FOR POINT CALCULATION
PRN3	PROCESS RTFLUX AND WRITE SELECTED POINT FLUXES ON SCRATCH ONE GROUP AT A TIME FOR POINT CALCULATION (METHOD 1)
PRPF	EDIT SPECIFIC REACTION RATE FOR FISSION IN ENERGY RANGES OF YIELD DATA FOR POINT CALCULATION
PRRF	EDIT SPECIFIC REACTION RATE FOR FISSION IN ENERGY RANGES OF YIELD DATA
PRTD	PRINT DOUBLE PRECISION ARRAY
PRTH	PRINT HOLLERITH ARRAY
PRTI	PRINT INTEGER ARRAY
PRTR	PRINT REAL ARRAY

PRTT	PRINT HOLLERITH TITLE
PTAT	OBTAIN REFERENCE ZONE NUMBERS FROM PTATDN IF IT EXISTS FOR POINT CALCULATION (METHOD 1)
PTNS	DETERMINE NUCLIDE SET AND INITIAL DENSITY INDEX (ZONE OR SUBZONE) FOR POINT CALCULATION
PURN	CONTROLS POINT EXPOSURE AND SHUTDOWN CALCULATION
PZT2	PROCESS TEMPERATURES FROM ZNTEMP (POINT CALCULATION)
QNAT	WRITE INTERFACE FILE QNATDN
QNAW	WRITE INTERFACE FILE ZNATDN (CONTINUOUS FUELING EXPOSURE)
REED	ENTRY IN RITE - DATA TRANSFER (EXTERNAL DEVICE TO MEMORY)
REHT	CALCULATE REACTION RATE TYPE DATA FOR EXPOHT
REOR	CHANGE VOLUME AND LOCATION DATA ORDER FOR POINT CALCULATION (METHOD 1)
RITE	DATA TRANSFER (MEMORY TO EXTERNAL DEVICE)
ROXX	ENTRY IN RITE - SPECIAL ADDRESS INITIALIZATION
ROXY	ENTRY IN DOPC - SPECIAL ADDRESS INITIALIZATION
RSTI	FUNCTION TO ASSIGN AN INTEGER VARIABLE TO A REAL VARIABLE LOCATION WITHOUT TYPE CONVERSION
SEEK	INTERFACE FILE MANAGEMENT
SERM	WRITE UNTERFACE FILE PROCESSING ERROR MESSAGE
SKER	WRITE SEEK RELATED ERROR MESSAGE AND STOP
SKNU	DETERMINE NUCLIDES IN SUPPLEMENTAL EXPLICIT CHAINS NOT TO BE TREATED WITH MATRIX EXPONENTIAL OR AVERAGE GENERATION RATE METHODS
STOR	MOVE ARRAY Y TO ARRAY X
TIMER	MULTI-PURPOSE ROUTINE TO PROVIDE CPU TIME, CLOCK TIME, CPU TIME REMAINING, I/O COUNT REMAINING, COMPUTER MODEL, JOB NAME, DATE AND TIME INFORMATION
TPNE	EDIT POWER NORMALIZATION FACTORS, EXPOSURE SUBSTEP TIMES, AND SHUTDOWN SUBSET TIMES
VOLP	COMPUTE REGION VOLUMES AND ZONE VOLUMES FROM POINT VOLUMES FOR POINT CALCULATION (METHOD 1) (DEBUG ONLY)
XEQC	INITIALIZE AN ARRAY WITH A CONSTANT
XEXC	MULTIPLY ARRAY X BY A CONSTANT
XEYC	MOVE DATA FROM ARRAY Y TO ARRAY X AND MULTI[PLY BY A CONSTANT
XPYC	ADD ARRAY Y MULTIPLIED BY A CONSTANT TO ARRAY X

ZCRI	SUM BY ZONE CLASS ABSORPTIONS BY NUCLIDE CLASS, FISSILE ABSORPTIONS, FERTILE CAPTURES, FISSILE DESTRUCTION RATE, AND FISSILE INVENTIRY
ZFMP	PROCESS RZFLUX (MODIFIED) FOR ZONE NUMBERS AND POINTS PER ZONE FOR POINT CALCULATION (METHOD 2)
ZFMV	DUMMY VOLUME AND LOCATION DATA FOR POINT CALCULATION (METHOD 2)
ZFM3	PROCESS RZFLUX (MODIFIED) AND WRITE POINT FLUXES ON SCRATCH ONE GROUP AT A TIME FOR POINT CALCULATION (METHOD 2)
ZIGY	SETUP INTEGRATION RANGE FOR FISSION REACTION RATE
ZNAW	WRITE INTERFACE FILE ZNATDN
ZOND	EDIT ATOM DENSITIES FOR ONE ZONE /SUBZONE
ZONI	ACCUMULATE MASS RATES IN KG/SEC
ZUCY	CHECK AND EDIT EXPLICIT CHAIN DATA
ZUCZ	DETERMINE MAXIMUM EXPLICIT CHAIN LENGTH
ZZPD	EDIT ZONE POWER DENSITY AND WRITE INTERFACE FILE ZNPOWD
ZZPF	CALCULATE ACTINIDE FEED RATE (KG/SEC) BY PATH FOR CONTINUOUS FUELING MODEL

END

DATE
FILMED
6/12/92