**A Computation System for Nuclear
Reactor Core Analysis**

D. R. Vondy
T. B. Fowler
G. V. Cunningham
L. M. Petrie

MASTER

BLANK PAGE

**Printed in the United States of America. Available from
National Technical Information Service
U.S. Department of Commerce
5285 Port Royal Road, Springfield Virginia 22161
Price: Printed Copy \$5.50; Microfiche \$3.00**

This report was prepared as an account of work sponsored by the United States Government. Neither the United States nor the Energy Research and Development Administration/United States Nuclear Regulatory Commission, nor any of their employees, nor any of their contractors, subcontractors, or 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.

ORNL-5158
Distribution Category
UC-79d - LMFBF Physics

Contract No. W-7405-eng-26

Neutron Physics Division

A COMPUTATION SYSTEM FOR NUCLEAR
REACTOR CORE ANALYSIS

D. R. Vondy
T. B. Fowler
G. W. Cunningham*
L. M. Petrie*

*Computer Sciences Division.

Date Published: April 1977

OAK RIDGE NATIONAL LABORATORY
Oak Ridge, Tennessee 37830
operated by
UNION CARBIDE CORPORATION
for the
ENERGY RESEARCH AND DEVELOPMENT ADMINISTRATION

Manuscript prepared by Virginia Glidewell and Nancy Layman

TABLE OF CONTENTS

<u>Section</u>	<u>Title</u>	<u>Page No.</u>
	ABSTRACT -----	v
1.	INTRODUCTION -----	1-1
2.	DESCRIPTION OF THE SYSTEM -----	2-1
3.	THE DRIVER -----	3-1
4.	THE CONTROL MODULE -----	4-1
	User Instructions to Control Module	
	CONTROLL (Table 4.3) -----	4-8
5.	DATA FILES -----	5-1
6.	CODE MODULE CONTROL AND ESSENTIAL DATA -----	6-1
7.	DATA FILE MANAGEMENT AND DATA TRANSFER -----	7-1
8.	JOB CONTROL INSTRUCTIONS -----	8-1
	User Supplied Job Control Instructions	
	(Table 8.2) -----	8-5
9.	THE INPUT PROCESSOR -----	9-1
	User Instructions for Input Processor -----	9-3
10.	THE FILE EDITOR -----	10-1
11.	ABOUT SPECIAL PROCESSORS -----	11-1
12.	SAMPLE USER INPUT AND EDIT -----	12-1
App. A:	Local Computer Facilities -----	A-1

ABSTRACT

This report documents a system which contains computer codes as modules developed to evaluate nuclear reactor core performance. The diffusion theory approximation to neutron transport may be applied with the VENTURE code treating up to three dimensions. The effect of exposure may be determined with the BURNER code, allowing depletion calculations to be made. The features and requirements of the system are discussed and aspects common to the computational modules, but the latter are documented elsewhere. User input data requirements, data file management, control, and the modules which perform general functions are described. Continuing development and implementation effort is enhancing the analysis capability available locally and to other installations from remote terminals.

1. INTRODUCTION

This report presents information about a modular code system under development. Reactor core analysis problems may be solved with application of the multigroup diffusion theory approximation to neutron transport, treating up to three space coordinates and long-time exposure (depletion). Emphasis in the methods development has been placed on satisfying the requirements for analysis of fast breeder reactors. The capability of the system is being extended by continuing effort funded by ERDA-RRD, Reactor Physics Branch.

Documentation of the computation system aspects in this report allows direct coverage of the individual code modules in other reports. Thus this report supersedes an earlier report¹ about the VENTURE code regarding the aspects covered herein. The way the system works, how it is used, and how control is exercised over the calculational path and over the instructions to the individual modules, are described.

The individual code blocks in this system were developed under guidelines established in an inter-installation effort sponsored by ERDA-RRD, Reactor Physics Branch. These guidelines address certain important aspects:

1. Hard interface data file specifications to effect compatible data communication of codes developed locally and at several other installations,
2. Programming requirements to ease the implementation on a major computer of a code operational on a computer of a different type, and
3. Standardization of a specific set of free-form user input data format rules.

Thus essential interface data file specifications have evolved, as for broad-group microscopic neutron, nucleus reaction cross sections, defining the essential link between a cross-section generation code and neutronics and related codes. These specifications continue to be upgraded and extended to satisfy the requirements. The programming

requirements are of special interest in providing for easy conversion of a program which uses both short and long words to a long-word machine (as IBM to CDC). Also, the adoption of special names for certain service routines and defined functions and arguments admits the capability to manage the interface data files and to transfer all data efficiently in large blocks without indexing; indeed, either the available "FORTRAN" input/output routines can be used for scratch data or a local implementation of the system routines, without major program changes. Standardization of the user input data form has been followed up by the development of an input data processor and a file editor at LASL which are incorporated in this system.

This system has evolved under the usual complications of maintaining viable capability for routine application on a production basis. It is used locally and from other installations via remote terminals on the local computers. The techniques used have been impacted by the requirement to satisfy both the IBM 360/91 and the IBM 360/195 computers which are operated under different multi-tasking schemes. It has been necessary to use some machine language routines to effect desired capability and efficient computer use, but this use has been held to a minimum to avoid serious problems in conversion to other computers. Thus the code modules are programmed in the FORTRAN language and the computation system is committed to the computer operating system FORTRAN data input/output. Certain machine language service routines are used.

We do expect that changes will have to be made to this system as requirements for enhanced capability are identified. Hopefully these can be incorporated within the existing framework covered by this basic documentation without impacting routine application. It would be attractive to change the dedication of logical unit numbers to simplify the user instructions; however, such change will not be made because it would adversely affect current application regarding the identification of data files and the allocation of space in the job instructions. New calculational modules will be added to the system after they have been developed and satisfy our reliability requirements for routine use as proven by validation testing.

END OF SECTION

2. DESCRIPTION OF THE SYSTEM

A small driver program is resident in the computer memory during a run. Access of the code modules in the system is indicated in Fig. 2.1. The driver summons code modules and performs such simple tasks as transfer of a block of data from the user input file to another unit in accordance with instructions from the control module. A small data block in memory is used for communication between the driver and the control module and to store key information required by the control module.

Thus primary control is vested in a control module which interprets instructions presented to it by the user. Initialization procedures are executed the first time it is accessed. These include actions required to effect management of data files. Upon the failure of or a successful completion of assigned tasks, the control module is again accessed to execute termination procedures.

Member Code Modules of the computation system satisfy certain requirements: restrictions on the use of files (logical units), using and generating files within the scheme of file management, compatible data transfer techniques regarding the use of operating system routines, compatible allocation and release of memory, and termination conditions. Other codes called special processors are in use which do not satisfy basic requirements of the system. They can be used only with careful control to protect and upgrade such basic data as the file management tables. Code modules shown in Fig. 2.1 are not yet all in production service, but will become available as the development progresses.

The code modules in this system are large and perform major tasks minimizing the amount of data which must be interfaced and allowing execution individually when desired. Numbers are assigned to the individual code modules for identification as shown in Fig. 2.1. These

Fig. 2.1. Components of the Computation System

modules and the tasks they carry out are described elsewhere. A new module can be added to the system to perform an additional function or to parallel an existing module only if it satisfies the rather severe center requirements including the tasks which it must perform. For example, a fully operational neutronics code module must be capable of the following task assignments:

1. Generate macroscopic cross sections from interface data files of the microscopic cross sections and nuclide concentrations for its own use.
2. Use available data from an interface flux file or a restart data file for initialization.
3. Solve any one of a variety of neutronics problems on demand, including the adjoint problem which may involve a fixed source.
4. Generate the necessary interface data files needed by other modules in the system, including the average zone flux file by group for subsequent reaction rate or exposure calculations, the regular, adjoint flux integrals, and upgrade the files, as of nuclide concentrations after a criticality search.
5. Perform the tasks within the interface data file management scheme and instructions to add new files and place new data in old files without destroying others.
6. Use the major task options from a special interface data file record which is available to the control module for control in sophisticated application (such control can be exercised only if the control module contains the necessary logic to interpret user instructions as control parameters for the code module).

In this system, many of the modules use a group-ordered microscopic cross-section file. Since a nuclide-ordered cross-section file is usually available from a cross-section processing code, a group-ordered file must be generated, one of the tasks of the cross-section processing code module.

END OF SECTION

3. THE DRIVER

The specifications for the modular code system driver were drawn up to satisfy rather general functional requirements under the requirements and restrictions of the local IBM computer systems. This driver is used locally in two modular code systems which require quite different modes of operation regarding module access. It is a resident machine language routine. In this code system, primary control is rested in a control module.

Execution starts in the driver and data is obtained from the user input data file. The first data card contains the name of the control module to be accessed. After this card is read, the next block of input data is transferred to unit 05 and the control module is accessed. Upon return from the control module, the assigned tasks are carried out by the driver to completion or apparent failure, and the control module is again summoned. Shown in Table 3.1 is the FORTRAN listing of the primary logic of a driver.

The control module assigns tasks to the driver. The first task assignment is to transfer data from the user input data file to unit 05, or not. Then the driver is to summon in order a series of code modules. Upon completion or apparent failure to complete any of these assigned tasks, the control module is again summoned to allow further decision making. Each block of input data is terminated by an END card; this terminal card is not transferred by the driver. A STOP zero number returned from a code module signals that the task assigned to the code was completed successfully. If the returned STOP number exceeds that value allowed, control is returned to the control module for special action following task failure. (A failure identified by the operating system, such as a data transfer error, causes return of a large STOP return number.)

Direct communication between the driver and the control module is provided by a shared block of storage passed by array arguments to the control module. Machine language is used for the driver for the following reasons:

BLANK PAGE

1. To achieve efficient data transfer without incurring the penalty of considerable memory required for the operating system FORTRAN input/output package,
2. To use the operating system loader directly to bring code modules into memory (which requires a machine language routine in any event), and
3. To achieve direct communication with the control module using data stored in memory.

The driver also writes, on option, an output unit consisting of the contents of each 80 column card processed. This edit may be suppressed by the user by a "dummy" assignment of the output unit dedicated to this task. (The user input data file and this special output unit are not assigned FORTRAN unit numbers.)

The driver allows cases to be stacked in a job. Upon completion of a case, a card containing the control module name is expected next in the user input data, or the end of the user input data file is a normal terminating condition. When additional user input data is expected, the end of this file causes termination without further code module access.

Certain operating system information may be supplied on the card containing the control module name. This data admits use of operating system routines for special purposes not now implemented in this code system.

TABLE 3.1. PORTMAN LISTING SHOWING LOGIC OF A DRIVER.

C	CDRIV A RESIDENT MACHINE LANGUAGE DRIVER IS USED - ITS REQUIREMENTS	DRIV 10
C	ARE ILLUSTRATED HERE WITH A PORTMAN LISTING.	DRIV 20
C		DRIV 30
C		DRIV 40
C	PRIMARY DATA USE BY THE DRIVER	DRIV 50
C		DRIV 60
C	IC (1) COUNT OF ACCESSES OF THE CONTROL MODULE,	DRIV 70
C	SET 0 FOR SUBSEQUENT CASE, -1 FOR TERMINATION	DRIV 80
C	IC (2) - IC (6) STOP RETURN NUMBERS ALLOWED FOR CODE MODULE ACCESSES	DRIV 90
C	IC (7) TASK COMPLETION FLAG, DRIVER SETS ZERO IF SUCCESSFUL	DRIV 100
C	IC (8) INSTRUCTION TO DRIVER TO TRANSFER USER INPUT DATA	DRIV 110
C	SET TO 0 FOR SUCCESSFUL TRANSFER	DRIV 120
C	IC (9) COUNT OF SUCCESSFUL CODE MODULE TASK COMPLETIONS	DRIV 130
C	IC (10) RESERVED	DRIV 140
C		DRIV 150
C	AC (1) NAME OF CONTROL MODULE	DRIV 160
C	AC (2) - AC (6) NAMES OF CODE MODULES TO BE ACCESSED	DRIV 170
C	AC (7) RESERVED FOR FUTURE DRIVER CONTROL	DRIV 180
C	AC (8) LATEST INPUT DATA HEADER (NAME OF SPECIAL PROCESSOR)	DRIV 190
C	AC (9)-AC (10) RESERVED FOR FUTURE DRIVER CONTROL	DRIV 200
C		DRIV 210
C		DRIV 220
C	REAL*8 AC,BLANK,CNOBEN,C0,END	DRIV 230
C	INTEGER*2 IC,JP,JD	DRIV 240
C		DRIV 250
C	COMMON /CDATA/ AC(40), IC(80), JP(72), JD(48)	DRIV 260
C		DRIV 270
C	DIMENSION C0(10)	DRIV 280
C	DIMENSION CNOBP(2)	DRIV 290
C		DRIV 300
C	DATA BLANK/'	DRIV 310
C	DATA END/'END	DRIV 320
C	DATA EQ/'='/	DRIV 330
C		DRIV 340
C		DRIV 350
C	IOIN = 1	DRIV 360
C	IOUT = 6	DRIV 370
C	IOFIVE = 5	DRIV 380
C	WRITE(IOUT,1000)	DRIV 390
C	DO 100 N = 1,40	DRIV 400
C	IC(N) = 0	DRIV 410
C	IC(N+40) = 0	DRIV 420
C	AC(N) = BLANK	DRIV 430
C	100 CONTINUE	DRIV 440
C	GO TO 103	DRIV 450
C	101 CONTINUE	DRIV 460
C	IC9 = IC(9)	DRIV 470
C	DO 102 N = 1,10	DRIV 480
C	IC(N) = 0	DRIV 490
C	AC(N) = BLANK	DRIV 500
C	102 CONTINUE	DRIV 510
C	IC(9) = IC9	DRIV 520

(CONT)

103 CONTINUE	DRIV 530
WRITE(IOUT,1005)	DRIV 540
C	DRIV 550
C READ CONTROL MODULE NAME.	DRIV 560
READ(IOIN,1001,END=115) X,CRODNN	DRIV 570
WRITE(IOUT,1006) CRODNN	DRIV 580
IF (X.NE.EQ) GO TO 114	DRIV 590
AC(1) = CRODNN	DRIV 600
REWIND IOFIVE	DRIV 610
C	DRIV 620
C READ CONTROL MODULE DATA.	DRIV 630
104 CONTINUE	DRIV 640
READ(IOIN,1004,END=113) C8	DRIV 650
WRITE(IOUT,1005) C8	DRIV 660
IF (C8(1).EQ.END) GO TO 105	DRIV 670
WRITE(IOFIVE,1004) C8	DRIV 680
GO TO 104	DRIV 690
105 CONTINUE	DRIV 700
END FILE IOFIVE	DRIV 710
REWIND IOFIVE	DRIV 720
106 CONTINUE	DRIV 730
IC(1) = IC(1)+1	DRIV 740
C*****	DRIV 750
C	DRIV 760
C ACCESS CONTROL MODULE HERE USING THE OPERATING SYSTEM LOADER,	DRIV 770
C COMMUNICATING THE COMMON DATA BLOCK CDATA.	DRIV 780
C	DRIV 790
C*****	DRIV 800
IF (IC(1).LT.0) GO TO 111	DRIV 810
IF (IC(1).EQ.0) GO TO 101	DRIV 820
IC8 = IC(8)	DRIV 830
IF (IC(8).EQ.0) GO TO 109	DRIV 840
C	DRIV 850
C READ INPUT OR SPECIAL PROCESSOR NAME.	DRIV 860
READ(IOIN,1004,END=110) C8	DRIV 870
WRITE(IOUT,1005) C8	DRIV 880
AC(8) = C8(1)	DRIV 890
IF (IC8.EQ.2) AC(2) = C8(1)	DRIV 900
REWIND IOFIVE	DRIV 910
C	DRIV 920
C READ INPUT OR SPECIAL PROCESSOR DATA.	DRIV 930
107 CONTINUE	DRIV 940
READ(IOIN,1004,END=110) C8	DRIV 950
WRITE(IOUT,1005) C8	DRIV 960
IF (C8(1).EQ.END) GO TO 108	DRIV 970
WRITE(IOFIVE,1004) C8	DRIV 980
GO TO 107	DRIV 990
108 CONTINUE	DRIV1000
END FILE IOFIVE	DRIV1010
REWIND IOFIVE	DRIV1020
IC(8) = 0	DRIV1030
109 CONTINUE	DRIV1040
IC(7) = 0	DRIV1050
IF (IC8.GE.3) GO TO 106	DRIV1060

(CONT)

```

C*****DRIV1070
C                                         DRIV1090
C ACCESS CODE MODULES HERE USING THE OPERATING SYSTEM LOADER - DRIV1090
C (MODULE NAMES ARE AC(2) THROUGH AC(6) OR TO A BLANK). DRIV1100
C ADD 1 TO IC(9) FOR EACH SUCCESSFUL MODULE ACCESS. DRIV1110
C IF THE RETURN STOP NUMBER FROM AN ACCESSED MODULE AC(M) EXCEEDS DRIV1120
C THE ALLOWED VALUE IC(M), IC(7) IS SET TO THE RETURNED NUMBER AND DRIV1130
C THE CONTROL MODULE IS ACCESSED WITHOUT FURTHER CODE MODULE DRIV1140
C ACCESSSES. DRIV1150
C DRIV1160
C*****DRIV1170
C GO TO 106 DRIV1180
110 CONTINUE DRIV1190
WRITE(IOUT,1008) DRIV1200
GO TO 106 DRIV1210
111 CONTINUE DRIV1220
WRITE(IOUT,1009) DRIV1230
GO TO 116 DRIV1240
112 CONTINUE DRIV1250
WRITE(IOUT,1006) DRIV1260
GO TO 116 DRIV1270
113 CONTINUE DRIV1280
WRITE(IOUT,1007) DRIV1290
GO TO 116 DRIV1300
114 CONTINUE DRIV1310
WRITE(IOUT,1003) DRIV1320
GO TO 116 DRIV1330
115 CONTINUE DRIV1340
WRITE(IOUT,1002) DRIV1350
GO TO 116 DRIV1360
116 CONTINUE DRIV1370
STOP DRIV1380
C DRIV1390
1000 FORMAT(1H0,'FORTRAN DRIVER FOR A MODULAR CODE SYSTEM FOR TESTING DRIV1400
      '(7-1-76)')/1H0) DRIV1410
1001 FORMAT(A1,A9) DRIV1420
1002 FORMAT(1H0,'NORMAL END OF RUN - EOF IN INPUT STREAM') DRIV1430
1003 FORMAT(1H0,'NO "" PRECEDING THE CONTROL MODULE NAME') DRIV1440
1004 FORMAT(10A9) DRIV1450
1005 FORMAT(1H ,10A9) DRIV1460
1006 FORMAT(1H0,'CONTROL MODULE NAME = ',A9) DRIV1470
1007 FORMAT(1H0,'END FILE READING CONTROL MODULE DATA') DRIV1480
1008 FORMAT(1H0/1H0,'END FILE ENCOUNTERED READING INPUT') DRIV1490
1009 FORMAT(1H0,'DRIVER INSTRUCTED TO TERMINATE') DRIV1500
      END DRIV1510

```

END OF SECTION

4. THE CONTROL MODULE

This section discusses the requirements of a control module. It is a manager program which causes a desired calculational path to be followed in accordance with user input instructions.

The control module is made available in the data set containing the other code modules, often a concatenation of several individual data sets. More than one control module likely exists in this data set.

A control module has a MAIN program, but entrance to start execution is to a subroutine of this module, effected when the program is assembled and placed in the data set. The FORTRAN version of a complete subroutine for entrance called GABY is shown in Table 4.1 and a skeleton main program. Use of the data in the communication block is documented in Table 4.2.

A simple control module is in use to allow continuing application of the codes in a relatively crude mode. User input instructions to this module named CONTROL1 are described in Table 4.3. Example instructions to this control module are shown in the sample run, Section 12. When a succession of eigenvalue problems are solved in a single job, this control module alters the neutronics control record to effect recovery of iteration and initialization data for problems solved after the first neutronics code access. When the exposure module is to be accessed in a job, instructions in the neutronics control record may be altered to cause essential files to be written.

Edits from the control module go to unit 99 and these, along with edits from the code modules, make up a brief history through the calculational path. This output may be sent to a remote terminal as described below (page 8-6).

Writing a control module is not a simple task. In this system, rather severe limitations are imposed by the scheme of data interfacing between the code modules. The control module plays a crucial role of initialization which quite generally cannot be avoided.

BLANK PAGE

TABLE 4.1. REQUIREMENTS OF CONTROL MODULE.

CHAIN REQUIREMENTS FOR THE CONTROL MODULE MAIN PROGRAM	MAIN	10
C	MAIN	20
C PRIMARY DATA USE BY THE DRIVER AND THE CONTROL MODULE	MAIN	30
C	MAIN	40
C IC (1) COUNT OF ACCESSES OF THE CONTROL MODULE,	MAIN	50
C SET 0 FOR SUBSEQUENT CASE, -1 FOR TERMINATION	MAIN	60
C IC (2) - IC (6) STOP RETURN NUMBERS ALLOWED FOR CODE MODULE ACCESS	MAIN	70
C IC (7) TASK COMPLETION FLAG, DRIVER SETS ZERO IF SUCCESSFUL	MAIN	80
C IC (8) INSTRUCTION TO DRIVER TO TRANSFER USER INPUT DATA	MAIN	90
C 0- NO	MAIN	100
C 1- YES	MAIN	110
C 2- YES, AND ACCESS CODE BASED ON FIRST CARD (A6)	MAIN	120
C 3- YES, RETURN TO CONTROL MODULE, NO CODE ACCESS	MAIN	130
C DRIVER SETS TO ZERO IF TRANSFER IS SUCCESSFUL	MAIN	140
C IC (9) COUNT OF SUCCESSFUL CODE MODULE TASK COMPLETIONS	MAIN	150
C IC (10) RESERVED	MAIN	160
C IC (11) MODE OF CURRENT OPERATION OF CONTROL MODULE	MAIN	170
C 0- NORMAL	MAIN	180
C 1- INITIALIZATION	MAIN	190
C 2- TERMINATION PROCEDURES IN EXECUTION	MAIN	200
C 3- SPECIAL PROCESSOR IS BEING USED	MAIN	210
C IC (12) ALLOWED PROCESSOR TIME	MAIN	220
C IC (13) ALLOWED INPUT/OUTPUT REQUESTS	MAIN	230
C IC (14) - IC (19) RESERVED FOR CONTROL MODULE USE	MAIN	240
C IC (20) NUMBER OF TIMES THE INPUT PROCESSOR IS ACCESSED	MAIN	250
C IC (21) IF GT 0 - DRIVER IS TRANSFERRING DATA FOR AND	MAIN	260
C ACCESSING INPUT PROCESSOR	MAIN	270
C IC (22) - IC (76) RESERVED FOR CONTROL MODULE USE	MAIN	280
C IC (77) INDICATOR FOR TRANSFER OF DATA	MAIN	290
C 0- NO	MAIN	300
C 1- TRANSFER DATA AND THEN ACCESS SPECIAL OR	MAIN	310
C INPUT PROCESSOR	MAIN	320
C -1- TRANSFER DATA AND SAVE HEADER NAME IN	MAIN	330
C AC (39) AND MODULE NAME IN AC (40)	MAIN	340
C IC (78) NUMBER OF MODULES LEFT TO BE ACCESSED	MAIN	350
C IC (79) MODULE ORDER NUMBER NOW BEING ACCESSED	MAIN	360
C IC (80) NUMBER OF MODULES TO BE ACCESSED IN THE RUN	MAIN	370
C	MAIN	380
C AC (1) NAME OF CONTROL MODULE	MAIN	390
C AC (2) - AC (6) NAMES OF CODE MODULES TO BE ACCESSED	MAIN	400
C AFTER DATA TRANSFER IS DONE IF REQUESTED	MAIN	410
C AC (7) RESERVED FOR DRIVER USE	MAIN	420
C AC (8) LATEST INPUT DATA HEADER (NAME OF SPECIAL PROCESSOR)	MAIN	430
C AC (9) - AC (10) RESERVED FOR DRIVER USE	MAIN	440
C AC (11) - AC (30) RESERVED FOR CONTROL MODULE USE	MAIN	450
C AC (39) HEADER NAME FOR LOOKED-HEAD DATA	MAIN	460
C AC (40) MODULE NAME FOR LOOKED-HEAD DATA	MAIN	470
C JP (1) SUPPLIED 'PARN' FIELD STRING LENGTH (SYSTEM ACCESS)	MAIN	480
C JD (1) A CHARACTER 'DDNAME' TABLE LENGTH (SYSTEM ACCESS)	MAIN	490
C	MAIN	500
C INPUT UNIT 05, CONDENSED OUTPUT UNIT 09	MAIN	510
C	MAIN	520

(CGWT)

C	INTEGER*2 IC,JP,JD	MAIN 530
	REAL*8 AC	MAIN 540
C		MAIN 550
	COMMON /CDATA/ AC(40), IC(80), JP(72), JD(48)	MAIN 560
C		MAIN 570
C		MAIN 580
C	PROTECT DRIVER INSTRUCTIONS AGAINST CONTROL MODULE FAILURE	MAIN 590
	IC1 = IC(1)	MAIN 600
	IC(1) = -1	MAIN 610
C	IF (IC1.LT.1) - SOMETHING IS WRONG.	MAIN 620
	IF (IC1.LT.1) IC1 = -1	MAIN 630
	IF (IC1.LT.1) GO TO 103	MAIN 640
	IF (IC1.EQ.1) GO TO 100	MAIN 650
	IF (IC(7).GT.0) GO TO 102	MAIN 660
	IF (IC(8).GT.0) GO TO 102	MAIN 670
	IF (IC1.GT.1) GO TO 101	MAIN 680
	100 CONTINUE	MAIN 690
C		MAIN 700
C	IMPLEMENT INITIALIZATION PROCEDURE HERE	MAIN 710
C	IF ERROR GO TO 102	MAIN 720
C		MAIN 730
	101 CONTINUE	MAIN 740
C		MAIN 750
C	IMPLEMENT NORMAL PROCEDURE HERE	MAIN 760
C	IF ERROR GO TO 102	MAIN 770
C		MAIN 780
	GO TO 103	MAIN 790
	102 CONTINUE	MAIN 800
C		MAIN 810
C	IMPLEMENT WRAP-UP PROCEDURE HERE	MAIN 820
C		MAIN 830
	103 CONTINUE	MAIN 840
C		MAIN 850
C	RETURN TO DRIVER.	MAIN 860
	IC(1) = IC1	MAIN 870
	RETURN	MAIN 880
	END	MAIN 890
		MAIN 900
C	CADY CONTROL MODULE ENTRANCE ROUTINE, TYPICAL	CADY 10
C		CADY 20
	SUBROUTINE CADY (IP, ID, IB, AB)	CADY 30
C		CADY 40
	INTEGER*2 IC,JP,JD,IB,IP,ID	CADY 50
	REAL*8 AC,AB	CADY 60
C		CADY 70
	COMMON /CDATA/ AC(40), IC(80), JP(72), JD(48)	CADY 80
C		CADY 90
	DIMENSION IP(1), ID(1), IB(1), AB(1)	CADY 100
C		CADY 110
C		CADY 120
	DO 100 I=1,48	CADY 130
	AC(I) = AB(I)	CADY 140

(CONT)

IC(I) = IB(I)	GABY 150
IC(I+40) = IB(I+40)	GABY 160
100 CONTINUE	GABY 170
DO 101 I=1,72	GABY 180
JP(I) = IP(I)	GABY 190
101 CONTINUE	GABY 200
DO 102 I=1,48	GABY 210
JB(I) = IB(I)	GABY 220
102 CONTINUE	GABY 230
C	GABY 240
C GO TO THE MAIN PROGRAM OF THE CONTROL MODULE	GABY 250
CALL MAIN	GABY 260
C	GABY 270
C RETURNS ROUTE FROM THE MAIN PROGRAM	GABY 280
DO 103 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
103 CONTINUE	GABY 330
C	GABY 340
C RETURNS TO THE DRIVER	GABY 350
RETURN	GABY 360
END	GABY 370

Table 4.2. Use of Data in the Communication Block
Between the Driver and the Control Module

Item	Use ^a	Control Module Action	Driver Action
IC(1)	Control module access count, termination flag.	If 1, read data from unit 05, save on unit 98; set 0 for subsequent case, < 0 for termination; < 1 upon entry indicates an error condition.	Initialize as 0 whenever user input instructs access of a control module, add 1 prior to each access of a control module, seek new user input instructions if 0, terminate if < 0.
IC(2)- IC(6)	Allowed STOP return code numbers corresponding to the code module name list AC(2)-AC(6) below.	Set here.	Used here.
IC(7)	Task completion flag, 0 - completed successfully, ≠0 - failure	If ≠0, execute wrap-up procedures	Store STOP return number from code module here if it exceeds that allowed.
IC(8)	User input data transfer flag (0 - no; 1 - transfer data; 2 - transfer data and access the special processor named on the first card (A0); > 2 - transfer data only)	Set to anticipate needs.	If > 0, move a block to unit 05, no other tasks if > 2; set to zero on successful data transfer.
IC(9)	Individual code module task completions	Probably set 0 at the start of each separate case; non-zero upon first access indicates previous calculations have been done, perhaps under control of this or another control module.	Add 1 after each acceptable return from a code module excluding the control module.
IC(10)	Reserved for future driver control.		

Table 4.2 (Cont'd.)

Item	Use ^a	Control Module Action	Driver Action
IC(11)	Control module mode of operation.	Uses: 0 - normal, 1 - initialization, 2 - wrap-up mode, 3 - special processor in use, 4 - data transfer mode only	(None)
IC(12)	Processor time left (reference).	Set and use.	(None)
IC(13)	1000 I/O requests left (reference).	Set and use.	(None)
IC(14)	Reserved for control module use.		
IC(20)	Access count of the input processor.		
IC(21)	If > 0, driver transferring data for and will call the input processor		(None)
IC(75)	If 1- write CONTROL file with IX(51) and IX(53) and on next access write IX(16). If 2- do not alter CONTROL file again.		
IC(76)	Access count of a nontronics code.		
IC(77)	If 1, transfer data and call the input or special processor. If -1, transfer data only and save header name in AC(30) and module name in AC(40).		
IC(78)	Number of modules left for the driver to call after the current one.		
IC(79)	Module number (IM(I), page 4-8) currently in use.		
IC(80)	Total number of entries on card 4 (pages 4-8).		
AC(1)	Control module name.	(None)	Set here.
AC(2)- AC(6)	Code module name.	Set here for accesses in order, end with a blank if < 5 entries.	Initialize blank.
AC(7)	Reserved for future driver control.		Initialize blank.
AC(8)	Latest input data header name	Used here, then set blank.	Initialize blank, set here unless first record read contains end-of-file.
AC(9)- AC(10)	Reserved for future driver control.		
AC(11)- AC(30)	Reserved for control module use.		

Table 4.2 (Cont'd.)

<u>Item</u>	<u>Type^a</u>	<u>Control Module Action</u>	<u>Driver Action</u>
AC(39)	Header name for data that was transferred for future use.		
AC(40)	Module name corresponding to header name AC(39).		
JP(1)	PART field, string length starts on a half-word boundary followed by a character string.	Needed for system routine use.	(None)
JD(1)	DSNAME table, length of table starts on a half-word boundary, then 8 character names follow, binary zeros nullify.	Needed for system routine use.	(None)

^aA control module will not be summoned under the apparent error condition of unsuccessful transfer of data for it to unit 0); a terminator flag in the input data is required. A request from the control module to take no action must not be allowed; the driver processes the user input file seeking a subsequent request for access to a control module. A stand-alone module not under constraints of this system can be executed on demand by this driver, and indeed several sequentially, and each may require input data, but this capability cannot be used within the constraints of this system requiring full control by a control module. The elements in the IC array are set zero and in array AC are set 'BLANK' initially by the driver, and prior to a subsequent access of any control module, IC(1) - IC(8) and IC(10) are set zero and AC(1) - AC(10) set 'BLANK'. Upon entry to a control module, AC(1) will contain its access name, IC(1) will be 1, IC(8) is a task completion flag set for transfer of the control module input data, IC(9) remains the code module access count.

Table 4.3. User Input Instructions to Control Module 'CONTROL1'

Card 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	090000	Primary memory allocation for data (040000) ^a
3	7-12	IP2	0	Reserved
3	13-18	IP3	0	Reserved
3	19-24	IP4	0	Override data block size for direct access files (4-byte words)
3	25-30	IP5	10	Processor time allowed to be used before wrap-up procedures are executed by the control module (min); must be less than allowed in the job control instructions, if > 0
3	31-36	IP6	50	Ditto 1000 Input/Output requests, if > 0
3	37-42	IP7		Additional input is included to present interface data file information, if > 0
3	43-48	IP8		Unit number containing stacked interface data files saved from a previous run, usually tape, unit 97, unformatted
3	49-54	IP9		Unit number on which interface data files are to be stacked for future use or archives, usually tape, unit 96 or 97, unformatted
3	55-57	IP10	0	Neutronics initialization option ^b
3	58-60	IP11	0	Mode of computation system operation 0 - Usual modular operation 1 - Pseudo stand-alone mode

^aNon-zero default values shown for entry of zero.

^bIf > 0, data from files RTFLUX and RZFLUX are used beginning with access count IP10 + 1; done on second access when exposure is to be treated if 0, not if < 0.

Table 4.3 (Cont'd.)

<u>Card Number</u>	<u>Columns (Format)</u>	<u>Reference Name</u>	<u>Typical Entry</u>	<u>Use</u>
3	61-63	IP12	0	Initial interface data file table initialization option 0 - Use SEEK default table (sets up 15 including CONTROL as the first) >0 - Set up this many from the list of which CONTROL is the first (see Table 5.2)
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 CONTROL)
3	67-69	IP14	0	Edit level for control module, primarily for debugging if >0
3	70-72	IP15 ^a	0	Primary termination options 0 - No action 1 - Save data files only upon successful completion before termination, per instructions 2 - Save data files as possible before termination only if the first code module access completed successfully, per instructions
4	(2413)	IM(I)		String of integers presenting the code module numbers to define the calculational path, terminating with a blank entry, limited to 48 entries. These numbers and the corresponding modules are shown in Fig. 2.1. Any number special processor is accessed by the number 2 (see Sec. 11), or others added in local application by 99. A loop over a substring of two or more code modules is defined by a negative entry which is the number of previous entries included in the loop and the next entry is the number of passes through the loop; additional accesses may be specified

^a Not implemented in the early version.

Table 4.3 (Cont'd.)

<u>Card Number</u>	<u>Columns (Format)</u>	<u>Reference Name</u>	<u>Typical Entry</u>	<u>Use</u>
				in the path after these two entries which define the loop. Note that user input data must normally be processed requiring that the input processor or a special processor or both be accessed before a code module. A loop may contain as the first module the input processor or a special processor to describe a basic multiple case run. Care must be taken in the arrangement of the user input data to satisfy input processor and special processor requirements consistent with these instructions and under restrictions defined elsewhere.

(NOTE: Input to the control module ends here unless IP7 > 0)

5+ (9(2x,A6)) H(I)

This is a string of interface data file names presented in sets (sub-strings) of names for four different purposes. Termination of each set of names is with a blank 8 column field or an entry 'X' in the last column of a separating 8 column field, limited to 72 entries (8 cards). Purposes of each of the sets is as follows:

<u>Set^a</u>	<u>Use</u>
1	Names of files made available to be used as supplied from a previous run (normally on disc).

^aNote: For the first two sets of file names, IP12 should usually be 1; assignment of unit numbers is in increasing order starting with the first set beginning with unit 11 (unit 11-20, then 30-39, then 70+). If a file name from a second set has already been assigned a unit number, it will not be used.

Table 4.3 (Cont'd.)

<u>Card Number</u>	<u>Columns (Format)</u>	<u>Reference Name</u>	<u>Typical Entry</u>	<u>Use</u>
				2 ^a Names of files to be selected from the files made available as a stack to be recovered from unit IP8.
				3 ^a Names of files to be saved by transfer to disk for unit numbers descending from unit logical number 95 (requires job control data for these units)
				4 ^a Names of files to be stacked on unit IP9 per completion instructions
(next)	(A3)	-	END	Normal data block terminator card required

(The user input data stream continues with data as required in order of code block access; generally each block of data requires a header card and a terminator END card.)

^a Not implemented in the early version.

END OF SECTION

5. DATA FILES

Certain reference information is presented in Table 5.1 about the data files. A data file may fall into one out of fourteen different classifications due to its contents or use, or may be a scratch file.

A scratch file is used locally by a code module; system referencing data for it is destroyed upon return from the code module where it is written, preventing its subsequent use in this system. A scratch file could reside on a tape, but quite generally the space is allocated for it on a system disc unit and the data is of binary type, unconverted, unformatted, in records which are accessed sequentially or directly as programmed in a specific code module. Where practical, scratch files are defaulted to reside in the computer memory.

All interface data files contain data in records accessed sequentially. The interface data files are special in that the contents are formally documented to allow use by any module. Thus they serve effectively to communicate data between code modules along a calculational path, and admit parallel code modules to be used to perform the same task of computation and data processing by different formulations or procedures. If data must be saved by a code module for its future use, it must be done as an interface file under governing rules.

Many of the different data files serve special purposes as indicated in Table 5.1. Thus a neutronics code may write a BCD formatted file of data on a tape for transmittal to another installation. Such special uses are discussed elsewhere.

All user input data is transferred from the user input data file to logical unit 05 in blocks as needed, generally in a look-ahead sense, under instructions from the control module. The control module stores away the input instructions it receives on unit 98, along with additional data it needs for recovery.

BLANK PAGE

Table 5.1. Data Files

Data Files	Committed FORTRAN Logical Unit Number ^a	Data Type	Primary Use	Produced By	Used By
<u>External</u>					
1. User Input Data	-	Card image	User input!	-	Driver
2. Interface Data Files	*	Binary	Same, reuse	Previous run ^c	
3. Stacked Interface Files	*	Binary	Same, reuse	Control module	Control Module
4. Formatted Interface Files	7,*	Card image	Outside world link	File editor	Input processor
5. Special Data Supplied	8	Binary	Link with the past	-	Special processor
6. Special Data Produced	4	Card image	Outside world link	Code module	-
7. Code Module Edit	6	-	Hard copy product	Code module	-
8. Condensed Edit (e.g., Control Module)	99	-	Condensed product	Control module	-
9. Driver Edit	-	-	Input data list	Driver	-
<u>Additional Internal</u>					
10. Interface Data Files	*	Binary	Data communication	Code module	Code module
11. Control Module Instructions	98	Binary	Primary control	Control module	Control Module
12. User Input Data Block	5	Card image	Blocked user input	Driver	{ Control module, Input processor, Special processors
13. Tables for File Management	9	Binary	Reference information	Modules	Modules
14. Plotting Data	2	Binary	Save data	Modules	-
15. Scratch Data Files	*	Binary ^b	Scratch data	Modules	Modules

^aA * indicates optional assignment within the scheme of file management under restrictions.

^bScratch files may be sequential, direct access by record, or defaulted to memory.

^cOnline storage capacity is limited, restricting use.

Space on each auxiliary unit to be used in a run must be allocated in the job control instructions, along with blocking information for sequential files. Many units are automatically allocated by use of the catalog procedure recommended for general use, along with default blocking data. The major data storage requirements in this system are normally certain scratch data files used by a neutronics code module. For a large neutronics problem, one disc volume may not be adequate to store a file, requiring that more than one be assigned and instructions for spanning be presented. These major file requirements are discussed in the documentation for the code modules. Generally edits are provided within the major code modules to show these requirements for a specific problem.

Unit numbers are associated with certain of the individual interface data file names when initialization procedures are followed, and thereafter automatically. The routine SEEK has been programmed to make the initial assignments given by the list in Table 5.2. Instructions to a control module allow use of this list as a default option, often useful, or cause the selection of a subset from the top of this list, or use of a list of names supplied by a user. Note that units 9 and 10 are always required and committed, and generation of a new file may extend the list. Unit numbers 21-29 and 40-69 are reserved for scratch use and simple control over space allocation and blocking by the user of these units for predictable requirements. Interface units then have numbers 10-20, 30-39, and 70 up, with the list automatically extended beyond the active list, as needed. The top of the active list can be determined by a code and units above this can be used as scratch if needed. If two or more code modules have large scratch data file requirements, an attempt is made to allocate the largest files to the same units with a primary blocking which is often dictated by one of the uses. For direct access by record, the blocking from the job control instructions is not used, but rather is tailored for the local needs. The catalog procedure allows simple allocation of space to the primary scratch units.

Table 5.2. Initial Interface File Assignments

Logical Unit Number ^a	File Name	Data Contents
9	(Active file catalogue)	
10	CONTRL	Control
11	GRUPXS	Cross sections
12	GEODST	Geometry description
13	NDXSRF	Nuclide referencing
14	ZNATDN	Nuclide concentrations
15	SEARCH	Criticality search data
16	RSTRTR	Restart data (neutronics)
17	RTFLUX	Regular flux
18	ATFLUX	Adjoint flux
19	RZFLUX	Zone flux
20	PWDINT	Power densities
30	CYSPRR	Cross section processor data
31	ISOTXS	Cross sections
32	ISOTXS, Version 2	Cross sections
33	FIXSRC	Fixed source

^aScratch files are assigned logical units 21-29 and 40-69; special files are in the range 1-8 and 98-99; the addition of additional files extends this list, e.g. 34, 35, etc.

If the user desires to produce a permanent file directly during a job run, he must predetermine the unit number which will be assigned to it. Examples are the generation of a group-ordered cross-section file or the generation of a regular flux file for future use.

Operation of the local computers is tailored somewhat to the job load. Under multi-tasking, not only is memory shared by the active jobs but also the online auxiliary storage, part of which is committed to the system. It is not simple to get effective separation of units on devices and data channels for highly efficient data transfer. The use of tape units must be minimized because of the small number available which must be shared by the active jobs. A long run which ties up a tape to store a file of results at the end, is not efficient use of the facility, although it may be the simplest way to run a job.

Print-out of the three output units is in a prescribed order. Some of the code modules also edit a very small amount of information on logical unit 99; thus this unit contains a basic history of a calculation and key, but minimum results. Note that the printing of the results transmitted to a remote terminal can be stopped as desired; usual practice is for a complete edit to be done locally for transmission. It is generally the most practical to use microfiche for large amounts of printed output. The trend should be away from tape use toward disc use. It is often practical to store one or more files on disc(s) for a short period and later transfer these to tape for long-time storage in a run separate from and after that of a calculation.

The formal specifications for the interface data files have gone through revisions in the inter-installation effort. Basic documentation of those in current use, version III, is provided in a LASL report.²

END OF SECTION

6. CODE MODULE CONTROL AND ESSENTIAL DATA

For each member code module in this system with a reference number higher than three, there is a record of instructions in the file CONTRL. These instructions include the parameters for control of the calculational path through the module, edit options, and options regarding the use and generation of interface data files. This information is available to the control module and may be altered by it to exercise control. The scheme implemented involves maintaining a reference copy of this file (on unit 98), and upgrading from it to an active version (on unit 10). Such trivial matters as informing the neutronics code to produce a zone, group flux file when exposure is to be calculated, can be taken care of automatically by a sophisticated control module even if not requested by the user. Also, the existence of files required for a calculation in accordance with instructions can be established before computation time is invested.

In implementing such control, it has generally been assumed that most input data will be processed prior to any calculation. Thus those files which are required should be (and should have been) generated before usual calculations start. Input data processors should generally be accessed to generate interface files at the start. In typical application, the first code block accessed is the input processor code block member, although prior use of one or more special processors may be necessary or desirable, to generate interface data files from user input data.

Sophisticated control modules will be developed and made available to select specific calculational paths through the modules, as for usual depletion problems involving exposure calculations and fuel management, with simple control over this path, over file generation, and over edit levels by the user.

The user analyst carries a rather severe burden in application of this or any sophisticated system for computation: interface data file requirements must be satisfied which depend on the calculational path through the modules and often on the path through one or more of the code

BLANK PAGE

modules. Thus if a code is to solve a neutron flux problem, it requires sufficient information to fully define the problem to be solved and other instructions which depend on the specific code. For a simple exposure calculation, a zone-average neutron flux file must be generated by a neutronics code, and compatible use of nuclide concentrations and referencing data and cross sections is essential for consistency. Interface data file requirements of the code modules in this system are documented elsewhere. An example is given here.

A depletion calculation is to be done by following the history between fuelings with alternate use of a neutronics code (VENTURE) and a code which calculates the effects of exposure (BURNER). Often the only data file available is a nuclide-ordered cross-section file ISOTXS. The following files must be made available for a basic calculation:

<u>Code</u>	<u>Files Required</u>	<u>Contents</u>
VENTURE	GEODST	Geometry description
	NDXSRF	Nuclide referencing
	ZNATDN	Nuclide concentrations
	CONTRL	Problem instructions
	GRUPXS	Group-ordered cross sections
BURNER	EXPOSE	Data for exposure calculation
	CONTROL	Problem instructions

Of course, BURNER also uses the files NDXSRF, ZNATDN, and GRUPXS, and, in addition, a zone, group flux file RZFLUX which must be generated by the neutronics code. The GRUPXS file must be generated by access of the cross-section processor. The CONTRL file must contain a record of instructions for each of these three codes, and an additional one for any other code module member of the system to be used (excluding the input processor which expects to find a block of user input data whenever it is accessed), as for reaction rate calculations. The user must supply the data to generate the files GEODST, NDXSRF, ZNATDN, and EXPOSE. Additional files may also be required, SEARCH to provide the necessary data for criticality searches and RODSET if control rods are to be

positioned by access of the code module available for this purpose. For perturbation calculations, regular and adjoint solutions must be obtained and regular, adjoint flux integrals, and files ATFLUX and PERTUB generated as needed. Certain results are placed on interface files by request, as of the power density map in file PWDINT by the neutronics code. With the capability to generate a file many times in a calculation comes the need for careful control, selection from the possibilities. Limited data storage capacity on line, especially for the kind which can be used for storing data for extended periods, is a real constraint on practical use of the local computer facilities.

A computer is used effectively and economically only if the neutronics problems are resolved efficiently. This aspect of application requires careful consideration. Reasonable selections must be made from the alternatives and automated procedures available for initialization and applying a procedure of iteration and balancing memory use and data transfer, and for identifying acceptable convergence criteria adequate to satisfy reliability requirements. It also generally requires that information obtained from the solution of one neutronics problem be made available for the next one within the interface data constraints, including the flux solution in the file RTFLUX.

END OF SECTION

7. DATA FILE MANAGEMENT AND DATA TRANSFER

The management of data files and data transfer are addressed here. Effective use of the computation system in sophisticated application is likely impossible unless the user understands how the interface data files are managed. The programmer adding capability to this system must understand the scheme for logical unit use and data transfer.

Data File Management

Each interface data file which is active at any time during a run exists on an auxiliary device, has a name, a version number, and a unique, associated logical unit number. Two files may have the same identifying name but not also the same version number; thus two criteria define uniqueness. Quite generally, code modules add files in a run which have unique identification; the version number is automatically indexed upward from the largest version number of existing files having the same name when one is added. Thus of those files having the same name, the one with the largest version number is interpreted as the latest version. Two files having the same name do not necessarily have the same data blocking nor consistent data. For example, there could be two files which have the name GEODST but contain different problem geometric descriptions (but different version numbers). Control of the generation of and often the use of data from interface data files is exercised in the instructions to the code modules (in records in file CONTRL). When an interface file is to be written, typically an existing file is replaced with the new data, or a new one is created if one does not exist. Without explicit instructions generally new files replace old ones as the default option, and explicit control is available to generate new files. In those situations where the data from two files having the same name are used at the same time, it is generally assumed that they are compatible and have the same blocking and apply to the same problem. Typically, the assumption is made that the next-to-latest and latest version numbers identify forward movement from some point, as in the time domain: the latest being the perturbed state

BLANK PAGE

or the condition after some time interval has elapsed from the state of the next-to-latest version number. Quite generally management functions such as deletion of interface files are not vested in the code modules.

Management of the interface data files is vested in the control module. Some of the tasks it may perform are:

1. Selectively initialize the file management tables (logical unit 09) based on user input instructions,
2. Selectively recover files which were stacked on a unit in a previous run,
3. Test for the presence of readable files which are reportedly supplied,
4. Protect the file management tables from the destructive capability of special processor modules,
5. Delete files and repack the file management tables,
6. Exercise validation tests on user identification names on the files,
7. Perform clean up of reference information for a subsequent case in the same run, and
8. Carry out a wrap-up procedure prior to termination of a run including saving individual files by transfer of the data to other units and stacking selected files for archives or future use, in accordance with user input instructions.

This role is performed in the programmed procedure for selecting a calculational path in accordance with instructions and imposing those conditions available for user control.

File management is accomplished by the exclusive use of one version of the routine SEEK by all member code modules and the control module in this system. This routine is used in a code to associate a logical unit number with a file name. The interface data file tables on unit 09 are accessed in an initialization step and updated with proper termination in the member code modules; the file reference data in these tables are

never used directly or manipulated within these modules. Certain data transfer information is included which reflects current use, information about FORTRAN access for units numbered 01 through 99, which may be of interest at the control module level.

Data Transfer

Data transfer within the member code modules is done by a package of routines which contains the following routines or ENTRY points:^a

<u>Name</u>	<u>Function</u>
DOPC	Basic input/output management
RITE	Data transfer from memory
REED	Data transfer to memory
CRIT	Data transfer, memory to memory (simulates fast to slow memory)
CRED	Data transfer (slow to fast memory)
SEEK	Interface file management

This package of routines provides for binary data transfer, unformatted (without type change), without indexing, using the operating system FORTRAN routines, or a special implementation of the system input/output routines which could be used for both interface and scratch data files.³ Records may be of mixed type. Access of a data file may be either in the sequential mode or direct access by record, the latter allowed only for scratch files.

The rather simple rules for the use of these routines must be followed in developing compatible codes. In addition, there are problem areas which must be avoided. Care must be taken to conform with data type requirements. Efficient memory use presents a challenge, especially since variable dimensioning is usually necessary which rules out the use of EQUIVALENCE to maintain data type identification. An optimizer is apt to remove data transfer steps from within nested loops in its ignorance about what data is actually being changed under this scheme.

^aThese may eventually be replaced by a set which is acceptable to the cooperating installations collectively.

Printing, reading user input data, punching, and generation of special BCD formatted files are done with usual FORTRAN statements, but restricted to specific FORTRAN unit numbers. Indeed, only limited input reading or generation of formatted files is allowed, and is generally done only by the input processor, the file editor, the control module, and the special processors.

Primary documentation of these data input/output routines is given in Table 7.1. Additional information has been published.⁴ A matter of some concern involves the use of special routines which implement the use of the operating system input/output routines outside of the system FORTRAN package; these must be obtained locally if used. If not used, references to special subroutines must be eliminated when a code module is loaded. Array communication calls to the routines (ROXX,ROXY) and special accesses of DOPC are not required if data is to be transferred only sequentially with the system FORTRAN package, but initialization and termination calls to DOPC are always required. When files are defaulted to reside in memory, a program must account for and protect the memory allocated to hold the files.

TABLE 7.1. DATA INPUT/OUTPUT SERVICE ROUTINES

(THESE COMMENT CARDS ARE PRIMARY DOCUMENTATION AT TIME OF PUBLICATION,
SEE THE FORTRAN LISTING OF THE PROGRAM TRANSMITTED FOR DETAILS)

CDOPC ROUTINE FOR OPENING AND CLOSING AUXILIARY FILES	
C MODIFIED 4-15-76 ORNL VERSION	DOPC 20
C	DOPC 30
C SUBROUTINE DOPC(ITASK,LU,RR,LR,ILR,IPSOR)	DOPC 40
C	DOPC 50
C COMMON/BCNTIO/MONOR(100),NVERS(100),NEDUN(100),INODD(100),NDF,	DOPC 60
C * BIO,INPT,IOUT,JOUT,IAV(11),IENA	DOPC 70
C COMMON/CONSAM/LTRACK,NOTEP,KKSTA,KBER	DOPC 75
C SEE ROUTINE SEEK FOR CONTENTS OF COMMON/BCNTIO/	DOPC 80
C	DOPC 90
C DIMENSION KK(1), IPSOR(1)	DOPC 100
C	DOPC 110
C REAL*8 DD,MONOR	DOPC 120
C	DOPC 130
C THE FOLLOWING ARE REQUIRED FOR A MODULE IN THE LOCAL SYSTEM	DOPC 140
C 1- INITIALIZE WITH A CALL TO DOPC	DOPC 150
C 2- ALLOCATE STORAGE	DOPC 160
C 3- COMMUNICATE STORAGE LOCATIONS WITH CALLS TO ROXY AND ROXY	DOPC 170
C 4- OPEN NECESSARY FILES (NOT REQUIRED FOR SEQUENTIAL FORTRAN)	DOPC 180
C WITH CALLS TO DOPC	DOPC 190
C 5- CLOSE FILES AS DESIRED WITH CALLS TO DOPC	DOPC 200
C 6- DETERMINE INTERFACE FILE UNITS WITH CALLS TO SEEK	DOPC 210
C 7- WRAP UP WITH A CLOSE CALL TO DOPC	DOPC 220
C (STAND ALONE OPERATION MAY USE SEEK, REED, AND RITE WITH	DOPC 230
C SEQUENTIAL OPERATION WITHOUT THESE OTHER REQUIREMENTS)	DOPC 240
C	DOPC 250
C DOPC DOES NOT ALTER THE ARGUMENTS COMMUNICATED, SO SUBSEQUENT USE	DOPC 260
C IS POSSIBLE WITH ONLY THE DESIRED CHANGES	DOPC 270
C NOTE THAT ARRAY KK IS RESERVED TO STORE CONTROL INFORMATION, SO	DOPC 280
C IT MUST BE PROPERLY COMMUNICATED AND NOT TAMPERED WITH	DOPC 290
C	DOPC 300
C ARGUMENT USE	DOPC 310
C -----	DOPC 320
C ROLE OF ARGUMENTS	DOPC 330
C -----	DOPC 340
C ITASK OPTION ON JOB TO BE DONE	DOPC 350
C 0 -INITIALIZATION, USUALLY REQUIRED, STORAGE ARRAYS	DOPC 360
C NOT YET AVAILABLE	DOPC 370
C RETURNS IPSOR(1) = INPT INPUT UNIT	DOPC 380
C IPSOR(2) = IOUT OUTPUT UNIT	DOPC 390
C IPSOR(3) = JOUT CONDENSED OUTPUT UNIT	DOPC 400
C OTHER VALUES OF IPSOR NOT USED OR SET	DOPC 410
C 1- OPEN SCRATCH DATA FILE	DOPC 420
C 2 -CLOSE SCRATCH DATA FILE	DOPC 430
C (CLOSING FILES STORED IN EXTENDED CORE MAY BEST	DOPC 440
C BE DONE IN REVERSE ORDER FROM OPENING)	DOPC 450
C 3 -WRAP UP, CLOSE ALL FILES BEFORE LEAVING A MODULE	DOPC 460
C LU LOGICAL UNIT NUMBER	DOPC 470
C RR NUMBER OF LOGICAL RECORDS IN THE FILE	DOPC 480

(CONT)

C	LR	LOGICAL RECORD LENGTH (REQUIRED FOR FIXED LENGTH	DOPC 490
C		RECORDS, OTHERWISE MAXIMUM REQUIRED), WORDS	DOPC 500
C	ILR	LENGTH OF THE DATA FILE, WORDS	DOPC 510
C	IPSOK(1)	MODE OF DATA ACCESS	DOPC 520
C		0- SEQUENTIAL	DOPC 530
C		1- SEQUENTIAL, RECORD LENGTHS STORED	DOPC 540
C		2- RANDOM ACCESS BY RECORD, FIXED LENGTH RECORDS	DOPC 550
C		3- RANDOM ACCESS BY RECORD, FIXED LENGTH RECORDS	DOPC 560
C		RANDOM WRITE REQUIRED WHICH MAY REQUIRE PREWRITES	DOPC 570
C	IPSOK(2)	REFERENCE TECHNIQUE OF DATA TRANSFER	DOPC 580
C		0- MANUFACTURER'S PORTMAN	DOPC 590
C		IPSOK(1) OPTION 1 = OPTION 0, 3 = 2 (IBM)	DOPC 595
C		1- LOCAL ADAPTATION OF THE SYSTEM ROUTINES	DOPC 600
C		2- DEFAULT THE FILE TO EXTENDED CORE	DOPC 610
C	IPSOK(3)	DEBUG MESSAGE EDIT LEVEL, EDITS DONE IF .GT. 0	DOPC 620
C	IPSOK(4)	RETURN ERROR FLAG, TASK COMPLETED ONLY IF 0	DOPC 630
C	IPSOK(5)	RELATIVE POSITION IN EXTENDED CORE DATA CONTAINER	DOPC 640
C		ARRAY X WHERE THE FILE STARTS (IPSOK(2) = 2 ONLY)	DOPC 650
C	IPSOK(6)	RESERVED FOR SPECIAL OPTIONS IN LOCAL TRANSFER SCHEME	DOPC 660
C		.LT.0 IS USED FOR TRACK OVERFLOW ON THE DISC	DOPC 665
C	IPSOK(7)	MAXIMUM BLOCK SIZE OF LOCAL TRANSFER SCHEME	DOPC 670
C		ZERO RECOMMENDED FOR LOCAL TRANSFER SCHEME	DOPC 675
C	IPSOK(8-10)	RESERVED	DOPC 680
C		PRIMARY INPUT/OUTPUT CONTROL PARAMETERS	DOPC 690
C		-----	DOPC 700
C	(VALUES USED HERE FOR DESCRIPTION ONLY, NOT PROGRAM VARIABLES)		DOPC 710
C	L	IS THE LOGICAL UNIT NUMBER	DOPC 720
C	N	IS THE LOCATION OF THE CONTROL DATA IN THE KK ARRAY (IF NEEDED)	DOPC 730
C	J	IS THE FILE LOCATION WITHIN THE EXTENDED CORE DATA ARRAY X	DOPC 740
C	NP	IS THE NUMBER OF PHYSICAL RECORDS PER LOGICAL RECORD	DOPC 750
C	NR	IS THE NEXT POSITION IN THE FILE BY RECORD COUNT	DOPC 760
C	NW	IS THE NEXT POSITION IN THE FILE BY WORD COUNT	DOPC 770
C	N	IS THE MAXIMUM RECORD COUNT WRITTEN	DOPC 780
C	IP	IS A FLAG INDICATING POSITION FOR MULTIPLE PHYSICAL RECORDS	DOPC 790
C	IL	IS THE LENGTH OF THE PREVIOUS RECORD WRITTEN	DOPC 800
C	NL	IS NORMALLY ONE TRACK LENGTH (LOCAL ACCESS SCHEME)	DOPC 805
C			DOPC 810
C	IPSOK		DOPC 820
C	(1) (2) INODD(L) NEDUN(L) KK(N) KK(N+1) KK(N+2) KK(N+3) KK(N+4) MORE		DOPC 830
C		-----	DOPC 840
C	0,1 0 0	NR NONE	DOPC 850
C	2,3 0 1	NP NONE	DOPC 860
C			
C	1	NR 2 IP N NR IPSOK(1) YES	DOPC 870
C	0 2	NW 5 NR J NR IL	DOPC 880
C	1 2	NW 4 NR J NR IL YES	DOPC 890
C	2,3 2	LR 3 NR J NR -	DOPC 900
C			DOPC 910
C		KK CONTROL ARRAY X DATA CONTAINER	DOPC 920
C		-----	DOPC 930
C	0	-	DOPC 940
C	0,2,3 1	$50 + 5 * ((LP-1)/NL)$	DOPC 950
C	1 1	$50 + 5 * ((LR-1)/NL) + NR$	DOPC 960
C	0,2,3 2	5	ILR DOPC 970
C	1 2	5 + NR	ILR DOPC 980

(CONT)

C	SYSTEMS, THIS WON'T WORK - A FIXED COMMON BLOCK CAN BE USED)	RITE 350
C	ENTRY BOXI (N , M , X , L , KK)	RITE 360
C	N LENGTH OF THE EXTENDED CORE DATA CONTAINER ARRAY X	RITE 380
C	X EXTENDED CORE DATA COLTAINER ARRAY	RITE 390
C	L LENGTH OF THE KK CONTROL ARRAY	RITE 400
C	KK CONTROL ARRAY	RITE 410
C	KK (1) LENGTH OF THE CONTROL ARRAY	RITE 420
C	KK (2) SIZE OF THE EXTENDED CORE DATA CONTAINER ARRAY	RITE 430
C	KK (3) USE OF THE SPACE FOR CONTROL DATA	RITE 440
C	KK (4) AMOUNT OF THE EXTENDED CORE DATA ARRAY USED	RITE 450
C	KK (5) LARGEST USE OF THE EXTENDED CORE DATA ARRAY	RITE 460

C	CSEEK UTILITY ROUTINE FOR MANAGING INTERFACE DATA FILES	SEEK 10
C	MODIFIED 3-1-75 ORNL VERSION	SEEK 20
C		SEEK 30
C	SUBROUTINE SEEK (HNAME, IVERS, HREP, HOP)	SEEK 40
C		SEEK 50
C	COMMON/RIGHTIO/HONOR(100), NVERS(100), NEDUH(100), INODD(100), NDP,	SEEK 60
C	* BIO, INPT, IOPT, JOUT, IAV(10), IENA	SEEK 70
C		SEEK 80
C	DIMENSION FNAME(15)	SEEK 90
C		SEEK 100
C	REAL*8 HNAME, HONOR, BLANK, CHANGE, PURGE, CONTRL, FNAME	SEEK 110
C		SEEK 120
C	DATA FNAME/6HCONTRL, 6HGRUPIS, 6HGEODST, 6HNDKSRP, 6HZNATDN, 6HSEARCH, SEEK 130	
C	* 6HSTRSTR, 6HRTPLUX, 6HATPLUX, 6HRZPLUX, 6HPUDINT, 6HCISPRR, 6HISOTXS, SEEK 140	
C	* 6HISOTXS, 6HPIISRC/	SEEK 150
C	DATA CHANGE/6HCHANGE/	SEEK 160
C	DATA BLANK/6H /, PURGE/6HPURGE /, CONTRL/6HCONTRL/	SEEK 170
C		SEEK 180
C	FILE MANAGEMENT SUBROUTINE FOR STANDARD INTERFACE FILES	SEEK 190
C	ALLOWING ONLY ONE FILE TO A LOGICAL UNIT	SEEK 200
C		SEEK 210
C	HNAME FILE NAME (EXCEPTIONS NOTED BELOW)	SEEK 220
C	IVERS FILE VERSION NUMBER (EXCEPTIONS NOTED BELOW)	SEEK 230
C	HREP LOGICAL UNIT NUMBER OR ERROR RETURN CONDITION	SEEK 240
C	(EXCEPTIONS NOTED BELOW)	SEEK 250
C	HOP OPTION NUMBER (EXCEPTIONS NOTED BELOW)	SEEK 260
C	HONOR(I) FILE NAME IN CATALOG, I IS LOGICAL UNIT NUMBER	SEEK 270
C	FNAME CATALOG FILE NAMES FOR INITIALIZATION	SEEK 280
C	NVERS(I) FILE VERSION IN CATALOG	SEEK 290
C	NEDUH(I) DATA FOR I/O TASK (RECORD POSITION FOR SEQUENTIAL)	SEEK 300
C	INODD(I) TECHNIQUE OF DATA TRANSFER INDICATOR (SEQUENTIAL, 0)	SEEK 310
C	IAV STORAGE ALLOWED FOR DIRECT ACCESS PARAMETERS	SEEK 320
C	NDP NUMBER OF FILES IN CATALOG	SEEK 330
C	BIO LAST LOGICAL UNIT NUMBER ASSIGNED	SEEK 340
C	INPT USER DATA INPUT LOGICAL UNIT	SEEK 350
C	IOPT USER OUTPUT DATA LOGICAL UNIT	SEEK 360
C	JOUT USER CONDENSED (TERMINAL) OUTPUT LOGICAL UNIT	SEEK 370
C	IENA USED FOR ENTRY PROTECTION IN DOPC	SEEK 380
C		SEEK 390
C	PRIMARY CONTROL OPTIONS	SEEK 400
C		SEEK 410
C	HNAME = 'CHANGE' INTERCHANGE FILE NAMES AND VERSION NUMBERS	SEEK 420
C	INPUT - HNAME, IVERS, HREP, HOP	SEEK 430

(CONT)

C			DOPC 990
C		SERVICE ROUTINES USED AND ENTRY POINTS	DOPC1000
C		-----	DOPC1005
C	DOPC	HANDLE FILE OPENING, ETC	DOPC1010
C	ROXY		DOPC1020
C	RITE	DATA TRANSFER	DOPC1030
C	REED ROXY		DOPC1040
C	CRIT	CORE, EXTENDED CORE DATA TRANSFER	DOPC1050
C	CRCD		DOPC1060
C	SECK	HANAGE INTERFACE DATA FILES	DOPC1070
C	PERR	DATA TRANSFER ERROR TERMINATION AFTER WRAP-UP	DOPC1075
C	PBSAN	LOCAL DATA TRANSFER SCHEME USING SYSTEM ROUTINES	DOPC1080
C		PREN PDISP PREAD PWRITE PPOINT FORSPC PDKSPC PPATR PCHECK	DOPC1090
C		(PLANS ARE TO SPLIT UP THIS LARGE ROUTINE)	DOPC1095
C	DEFILE	IBM FORTRAN DIRECT ACCESS FILE OPENING	DOPC1100
C	CLOSDA	CLOSE FORTRAN DIRECT ACCESS FILES	DOPC1110
C			DOPC1120

C	CRITE DATA TRANSFER ROUTINE, ORNL VERSION	
C	MODIFIED 4-15-76 ORNL VERSION	RITE 20
C	SUBROUTINE RITE(LUN,IREC,ARRAY,NUDS,BCDE)	RITE 30
C		RITE 40
C	COMMON/BCDIO/HONOR(100),NVERS(100),NEDUH(100),INODD(100),NDP,	RITE 50
C	* BIO,INPT,IOUT,JOUT,IAY(11),IEMA	RITE 60
C	COMMON/CONSAR/LTRACK,NOTEP,KKSTA,KBER	RITE 65
C		RITE 70
C	DIMENSION ARRAY(NUDS), I(1), KK(1)	RITE 80
C		RITE 90
C	REAL*8 HONOR	RITE 100
C		RITE 110
C	THIS ROUTINE PACKAGES THE PROCEDURES FOR TRANSFER OF DATA BETWEEN	RITE 120
C	MEMORY AND DISK AND MEMORY - MEMORY (PSEUDO EXTENDED CORE)	RITE 130
C		
C	ON SOME COMPUTERS IT WILL BE NECESSARY TO COMMUNICATE THE	RITE 150
C	STARTING ADDRESSES OF ARRAYS KK AND I IN SOME OTHER MANNER	RITE 160
C		RITE 170
C	LUN LOGICAL UNIT NUMBER OF THE FILE	RITE 180
C	IREC RECORD NUMBER TO BE ADDRESSED	RITE 190
C	ARRAY REFERENCE ARRAY IN CORE TO PROVIDE LOCATION FOR TRANSFER	RITE 200
C	NUDS LENGTH OF THE ARRAY THE BE TRANSFERRED	RITE 210
C	IF 0, REQIND ONLY	RITE 220
C	IF .LT. 0, END FILE AND REQIND ONLY	RITE 230
C	MODE USED FOR ASYNCHRONOUS OPERATIONS	RITE 240
C	0- NO	RITE 250
C	1- TRANSFER OPERATION	RITE 260
C	2- CHECK PREVIOUS TRANSFER COMPLETION, REQUIRED AS A	RITE 270
C	SPECIAL OR EXTRA CALL PRIOR TO USE OF THE DATA,	RITE 280
C	EXCEPT WHEN PROTECTION IS PROVIDED BY A SUBSEQUENT	RITE 290
C	CALL FOR ADDITIONAL DATA TRANSFER PRIOR TO USE	RITE 300
C		RITE 310
C	IO = 0	RITE 320
C	GO TO 101	RITE 330
C		RITE 340
C	VARIABLE LENGTH DATA BLOCKS COMMUNICATED HERE (ON SOME OF THE	

(CONT)

```

C      IF IVERS EQ 0, THE NAMES AND VERSION NUMBERS ASSOCIATED WITH CATALOG POSITIONS NREP AND NOP WILL BE
C      INTERCHANGED
C      IF IVERS GT 0, THE NAMES AND VERSION NUMBERS ASSOCIATED
C      LOGICAL UNIT NUMBERS NREP AND NOP WILL BE
C      INTERCHANGED
C      NREP WILL BE RETURNED ZERO FOR ERROR CONDITION
C      NOP = 0, LOCATE FILE PRIOR TO READING, WRITING
C      INPUT - HNAME, IVERS, NOP
C      IF IVERS EQ 0, RETURNS VERSION NUMBER AND LOGICAL
C      UNIT NUMBER OF THE LATEST VERSION
C      IF IVERS EQ N, RETURNS LOGICAL UNIT NUMBER FOR
C      VERSION N
C      NREP WILL BE RETURNED ZERO FOR ERROR CONDITION
C      NOP = 2 CREATE A NEW FILE VERSION PRIOR TO WRITING
C      WILL INITIALIZE CONTRL IF NOT IN TABLE POSITION 1
C      INPUT - HNAME, NOP
C      RETURNS NEW VERSION NUMBER AND LOGICAL UNIT NUMBER
C      AND ADDS NEW DATA TO CATALOG
C      NREP WILL BE RETURNED ZERO FOR ERROR CONDITION
C      NOP = 3 CATALOG INITIALIZATION, PUTS DATA ON UNIT ICAT
C      IF NREP .GT. 0, IT IS THE NUMBER OF FILES FROM THE LIST
C      PURGING OPTION
C      NOP = 4 INPUT - HNAME, IVERS, NOP
C      IF HNAME EQ 'PURGE ', ALL FILE REFERENCES WILL BE
C      DELETED EXCEPT FOR CONTRL
C      IF HNAME NE 'PURGE ' AND IVERS EQ 0, ALL VERSIONS
C      OF THE FILE WILL BE DELETED
C      IF HNAME NE 'PURGE ' AND IVERS EQ N, ONLY VERSION N
C      OF THE FILE WILL BE DELETED
C      NOP = 5 EXTRACT CONTENTS OF CATALOG
C      INPUT - IVERS, NREP, NOP
C      IF IVERS EQ 0, NREP WILL BE TREATED AS THE POSITION
C      IN THE CATALOG AND THE ASSOCIATED FILE NAME,
C      VERSION NUMBER, AND LOGICAL UNIT NUMBER
C      WILL BE RETURNED
C      IF IVERS GT 0, NREP WILL BE TREATED AS THE LOGICAL
C      UNIT NUMBER AND THE ASSOCIATED FILE NAME,
C      VERSION NUMBER, AND CATALOG POSITION
C      WILL BE RETURNED
C      NREP WILL BE RETURNED ZERO FOR ERROR CONDITION
C      NOP = 6,7 NOT USED
C      NOP = 8 READ CONTENTS OF CATALOG (SAVED IN LAST MODULE
C      EXECUTED)
C      NOP = 9 WRITE CONTENTS OF CATALOG (FOR USE BY NEXT MODULE
C      TO BE EXECUTED)
C      NOP = 10 EDIT CONTENTS OF CATALOG
C      IF NREP.GT.0, WRITES JOUT INSTEAD OF IOUT
C      NOP = 11 RETURN UTILITY UNIT NUMBERS
C
C      ***** REQUEST ***** ***** RESPONSE *****
C      HNAME IVERS NREP NOP HNAME IVERS NREP NOP
C      CHANGE 0 IT1 IT2 CHANGE 0 IT1 IT2
C      CHANGE 0 0-E IT2

```

(CONT)

C	CHANGE	H	LU1	LU2	CHANGE	H	LU1	LU2	SEEK 900
C					CHANGE	H	0=E	LU2	SEEK 990
C	H	0	-	0	H	HL	LU	0	SEEK1000
C					H	0	0=E	0	SEEK1010
C	H	H	-	0	H	H	LU	0	SEEK1020
C					H	H	0=E	0	SEEK1030
C	H	0	-	1	H	HL	LU	1	SEEK1040
C					H	0	0=E	1	SEEK1050
C	H	H	-	1	H	H	LU	1	SEEK1060
C					H	H	0=E	1	SEEK1070
C	H	-	-	2	H	HH	LU	2	SEEK1080
C					H	-	0=E	2	SEEK1090
C	-	-	-	3	-	-	INIT	3	SEEK1100
C	PURGE	-	-	4	PURGE	-	-	4	SEEK1110
C	H	0	-	4	H	0	-	4	SEEK1120
C	H	H	-	4	H	H	-	4	SEEK1130
C	-	0	IT	5	H	H	LU	5	SEEK1140
C					-	0	0=E	5	SEEK1150
C	-	H	LU	5	H	H	IT	5	SEEK1160
C					-	H	0=E	5	SEEK1170
C	H	H	LU	6	H	H	LU	6	SEEK1180
C	-	-	-	7	-	-	-	7	SEEK1190
C	-	-	-	8	-	-	-	8	SEEK1200
C	-	-	-	9	-	-	-	9	SEEK1210
C	-	-	-	10	-	-	-	10	SEEK1220
C	-	-	-	11	-	-	-	10	SEEK1230
C					INPT	IOBT	JOBT	SEEK1230	
C								SEEK1240	
C	H		FILE NAME					SEEK1250	
C	H		VERSION NUMBER					SEEK1260	
C	HL		LAST VERSION NUMBER					SEEK1270	
C	HH		NEW VERSION NUMBER					SEEK1280	
C	LU,LU1,LU2		LOGICAL UNIT NUMBER					SEEK1290	
C	IT,IT1,IT2		TABLE POSITION					SEEK1300	
C	H		GREATER THAN ZERO					SEEK1310	
C	0=E		ERROR					SEEK1320	
C	-		NOT USED (BUT NAME2 MUST NOT BE 'CHANGE')					SEEK1330	
C								*****SEEK1340	

```

CCBIT BASIC ROUTINE FOR MEMORY, EXTENDED CORE DATA TRANSFERS
C      A MUCH FASTER ASSEMBLER LANGUAGE VERSION OF THIS IS IN USE
C
C      SUBROUTINE CRIT(X,Y,H,J)
C
C      DIMENSION X(1), Y(1)
C
C      USE OF ARGUMENTS
C      ---
C      X      DATA ARRAY IN MEMORY
C      Y      DATA ARRAY IN EXTENDED (SLOW OR PARTITIONED) CORE
C      H      LENGTH OF DATA STRING TO BE MOVED, SHORT WORDS, MIXED TYPE
C              ALLOWED
C      J      SYSTEM FLAG ON SOME SYSTEMS, NOT USED LOCALLY
C

```

(CONT)

```
C      ENTRY FOR EXTENDED CORE TO MEMORY TRANSFER
C
C      ENTRY CRED (Y,I,N,J)
C
      DO W1 I=1,N
        Y(I)=X(I)
101 CONTINUE
      RETURN
      END
```

END OF SECTION

8. JOB CONTROL INSTRUCTIONS

To run a job on the local IBM computers (model 91 at X-10 and 195 at K-25), the computer operating system must have available the job control instructions for that job. Most of the job control instructions for executing the code modules in this computational system are cataloged and stored in a library which is accessed at run time. This catalog is shown in Table 8.1 for the 195 (slightly different from the 91).

The information in the cataloged job control instructions which is problem dependent is parameterized and actual values of these parameters must be supplied by the user. These parameters include core region size, CPU time, and space, buffer size, and blocking factors for each auxiliary storage I/O data file.

Table 8.2 shows a typical set of job control instructions which the user supplies to supplement or override the instructions in the catalog. Only a brief description of these instructions will be given here. For information about the use of the local computers, refer to the local report "Programmers Notebook" issued by the Computer Sciences Division of the Nuclear Division of Union Carbide.

User Supplied Job Control Instructions

Referring to Table 8.2, the parameters NB1, NB2, B1, F2, NX, NS, and N1 through N16 are described under Problem Dependent Data File Parameters below. The core storage region size (in k=1024 bytes), specified on the // GOSIZE= card must be approximately 300 plus the primary data allocation size (IP1, Table 4.3) divided by 250. Any cards present between the // GOSIZE= card and the //GO.SYSIN DD * card override the corresponding cards in the catalog, and must be given in the same order as those in the catalog - unit numbers increasing. Note that when overriding cards in the catalog, a GO. should be punched following the // in column 1-2: //GO.

The catalog identifies three individual data files for printed output. The file PRINT receives output from the driver and consists of all user

BLANK PAGE

```

//          DCB=(RECFM=FBS,LRECL=80,BLKSIZE=3200)
//PT05F001 DD UNIT=SYSDA,SPACE=(80,(4000,1000)),
//          DCB=(RECFM=F,BUFNO=1,LRECL=90,BLKSIZE=80)
//FT06F001 DD SYSOUT=A,DCB=(RECFM=VBA,LRECL=137,BLKSIZE=1100)
//FT07F001 DD SYSOUT=B,DCB=(RECFM=F,BLKSIZE=80)
//FT08F001 DD UNIT=SYSDA,SPACE=(8B1,(6N1,2)),
//          DCB=(RECFM=VBST,LRECL=X,BUFNO=6NB1,BLKSIZE=6B1)
//FT09F001 DD UNIT=SYSDA,SPACE=(6B1,(6NS,1)),
//          DCB=(RECFM=VBST,LRECL=X,BUFNO=6NB1,BLKSIZE=6B1)
//FT10F001 DD UNIT=SYSDA,SPACE=(6B1,(6NS,1)),
//          DCB=(RECFM=VBST,LRECL=X,BUFNO=6NB1,BLKSIZE=6B1)
//FT11F001 DD UNIT=SYSDA,SPACE=(8B1,(6N1,6NX)),
//          DCB=(RECFM=VBST,LRECL=X,BUFNO=6NB1,BLKSIZE=6B1)
//FT12F001 DD UNIT=SYSDA,SPACE=(6B1,(6N1,6NX)),
//          DCB=(RECFM=VBST,LRECL=X,BUFNO=6NB1,BLKSIZE=6B1)
//FT13F001 DD UNIT=SYSDA,SPACE=(6B1,(6N1,2)),
//          DCB=(RECFM=VBST,LRECL=X,BUFNO=6NB1,BLKSIZE=6B1)
//FT14F001 DD UNIT=SYSDA,SPACE=(6B1,(6NS,1)),
//          DCB=(RECFM=VBST,LRECL=X,BUFNO=6NB1,BLKSIZE=6B1)
//FT15F001 DD UNIT=SYSDA,SPACE=(6B1,(6NS,1)),
//          DCB=(RECFM=VBST,LRECL=X,BUFNO=6NB1,BLKSIZE=6B1)
//FT16F001 DD UNIT=SYSDA,SPACE=(6B1,(6N1,6NX)),
//          DCB=(RECFM=VBST,LRECL=X,BUFNO=6NB1,BLKSIZE=6B1)
//FT17F001 DD UNIT=SYSDA,SPACE=(6B1,(6N16)),
//          DCB=(RECFM=VBST,LRECL=X,BUFNO=6NB1,BLKSIZE=6B1)
//FT18F001 DD UNIT=SYSDA,SPACE=(6B1,(6N16)),
//          DCB=(RECFM=VBST,LRECL=X,BUFNO=6NB1,BLKSIZE=6B1)
//FT19F001 DD UNIT=SYSDA,SPACE=(6B1,(6NS,1)),
//          DCB=(RECFM=VBST,LRECL=X,BUFNO=6NB1,BLKSIZE=6B1)
//FT20F001 DD UNIT=SYSDA,SPACE=(6B1,(6N13)),
//          DCB=(RECFM=VBST,LRECL=X,BUFNO=6NB1,BLKSIZE=6B1)
//FT21F001 DD UNIT=SYSDA,SPACE=(6B1,(6N1,2)),
//          DCB=(RECFM=VBST,LRECL=X,BUFNO=6NB1,BLKSIZE=6B1)
//FT22F001 DD UNIT=SYSDA,SPACE=(6B1,(6NS,1)),
//          DCB=(RECFM=VBST,LRECL=X,BUFNO=6NB1,BLKSIZE=6B1)

```

(CONT)

assigned a specific record type. The record types are numbered consecutively beginning with the first record following the file identification record. Data for the file identification record are supplied on the file identifier card, identifying the record type of the data following. If the list for record type *m* is repeated in the file, *mD* must precede each repetition of the list. In overlay applications the defined ordering of records is followed; records to be left intact may be skipped as desired. Skipped records are copied unmodified from the reference file onto the new file. For the overlay option, any parameter that controls the length or number of records in the file may not be changed.

3. STOP Card. Input processing is terminated when a card containing STOP as the first entry is encountered.

4. FLAG Card. This is a new option card that can be used to change the normal input unit number and/or control the card image printout. Cards of the form

FLAG IUNIT IPRINT ISTART IEND

can be inserted anywhere in the input stream (preferably before OV cards). FLAG is a keyword identifying the special instructions to the processor. IUNIT, IPRINT, ISTART, and IEND are integers. IUNIT identifies a secondary unit from which the input processor will immediately begin processing. Processing on the secondary unit continues until a STOP card (no additional cards will be processed) or end-of-file (processing will then revert to the normal input unit) is encountered. A zero value for IUNIT instructs the processor to continue on the normal input unit. IPRINT controls card image printing. If IPRINT is zero, all card images will be printed. If

```

//          DCB=(RECFM=VBST,LRECL=X,BUPHC=6NB1,BLKSIZE=6B1)
//PT33P001 DD UNIT=SYSDA,SPACE=(6B1,(6N4)),
//          DCB=(RECFM=VBST,LRECL=X,BUPHC=6NB1,BLKSIZE=6B1)
//PT34P001 DD UNIT=SYSDA,SPACE=(6B1,(6N1,6N1)),
//          DCB=(RECFM=VBST,LRECL=X,BUPHC=6NB1,BLKSIZE=6B1)
//PT35P001 DD UNIT=SYSDA,SPACE=(6B1,(6N1,2)),
//          DCB=(RECFM=VBST,LRECL=X,BUPHC=6NB1,BLKSIZE=6B1)
//PT36P001 DD UNIT=SYSDA,SPACE=(6B1,(6N6,2)),
//          DCB=(RECFM=VBST,LRECL=X,BUPHC=6NB1,BLKSIZE=6B1)
//PT37P001 DD UNIT=SYSDA,SPACE=(6B1,(6N15,2)),
//          DCB=(RECFM=VBST,LRECL=X,BUPHC=6NB1,BLKSIZE=6B1)
//PT38P001 DD UNIT=SYSDA,SPACE=(6B1,(6N15,2)),
//          DCB=(RECFM=VBST,LRECL=X,BUPHC=6NB1,BLKSIZE=6B1)
//PT39P001 DD UNIT=SYSDA,SPACE=(6B1,(6N15,2)),
//          DCB=(RECFM=VBST,LRECL=X,BUPHC=6NB1,BLKSIZE=6B1)
//PT40P001 DD UNIT=SYSDA,SPACE=(6B2,(6N5)),
//          DCB=(RECFM=PT,BUPHC=6NB2)
//PT41P001 DD UNIT=SYSDA,SPACE=(6B1,(6N4)),
//          DCB=(RECFM=VBST,LRECL=X,BUPHC=6NB1,BLKSIZE=6B1)
//PT42P001 DD UNIT=SYSDA,SPACE=(6B2,(6N7)),
//          DCB=(RECFM=VBST,LRECL=X,BUPHC=6NB2,BLKSIZE=6B2)
//PT43P001 DD UNIT=SYSDA,SPACE=(6B2,(6N8)),
//          DCB=(RECFM=VBST,LRECL=X,BUPHC=6NB2,BLKSIZE=6B2)
//PT44P001 DD UNIT=SYSDA,SPACE=(6B1,(6N5,1)),
//          DCB=(RECFM=VBST,LRECL=X,BUPHC=6NB1,BLKSIZE=6B1)
//PT45PJ01 DD UNIT=SYSDA,SPACE=(6B1,(6N9)),
//          DCB=(RECFM=VBST,LRECL=X,BUPHC=6NB1,BLKSIZE=6B1)
//PT46P001 DD UNIT=SYSDA,SPACE=(6B1,(6N10)),
//          DCB=(RECFM=VBST,LRECL=X,BUPHC=6NB1,BLKSIZE=6B1)
//PT47P001 DD UNIT=SYSDA,SPACE=(6B1,(6N5,1)),
//          DCB=(RECFM=VBST,LRECL=X,BUPHC=6NB1,BLKSIZE=6B1)
//PT48P001 DD UNIT=SYSDA,SPACE=(6B1,(6N1,2)),
//          DCB=(RECFM=VBST,LRECL=X,BUPHC=6NB1,BLKSIZE=6B1)
//PT49P001 DD UNIT=SYSDA,SPACE=(6B1,(6N17)),
//          DCB=(RECFM=VBST,LRECL=X,BUPHC=6NB1,BLKSIZE=6B1)

```

(CONT)

A Data Field

The data field contains words which may be data along with control parameters and allowed comments. This section describes the allowable data forms, the structure of data entries and the various control options available to simplify data entries.

1. Allowable Data Forms. The allowable data forms are integers, single precision floating point numbers, and Hollerith words. Any combination of these forms is allowed within the data field. For files containing double precision numbers, each number is first constructed in single precision on a short word machine and then converted to a double word for the file. When the overlay option is used, a double precision number is converted to single precision before overlay occurs. After the overlay, reconversion is done to double precision for the file. Card input specifications determine the allowable combinations of data forms.

a. Numeric Words. Numeric entries are integers if they contain no decimal point or exponent. Otherwise, numerics are taken to be real (floating-point) numbers. An exponent is an expression containing an optionally signed integer preceded optionally by the letter E. The sign or the E may be absent but not both. An exponent in a numeric word must always be preceded by at least one digit, otherwise, forms En or E+n delimited by blanks or commas are interpreted as Hollerith words. To allow data written by the computer to be read, a blank after the E is treated as a plus sign. Also, a D (not the same as mD) will be treated as if it were an E. Examples of allowed forms are:

b. Hollerith Words. Hollerith data are specified in three different ways:

1. by enclosing the data in asterisks,
2. by using the nH specifications, or
3. by beginning a single word entry, containing no more than six characters, with a letter.

In all cases, Hollerith data are stored at the rate of six characters per word. On IBM machines each six-character word is stored in a double precision word. Input words of less than six characters are stored left adjusted with blank fill. Imbedded blanks in words are allowed only on options (1) and (2); in option (3) blanks delimit words.

For example, the following are all equivalent:

```
*U238A * *PU239B* *IRON *
*U238A PU239BIRON *
16HU238A PU239BIRON
U238A PU239B IFON
```

The requirements for a 12-word (72-character) title may easily be satisfied by putting an asterisk (*) in columns 1 and 72 and limiting the actual title to columns 2 through 71.

It is recommended that Hollerith always be delimited using the asterisk notation. In any event, if the Hollerith includes the character D in any position of the word or the character E in the last position of a word,

END

INPUT PROCESSOR

COMMENT*****DATA FOR THE INPUT PROCESSOR GOES HERE

END

DCRSPR

COMMENT*****DATA FOR THE SPECIAL INPUT PROCESSOR DCRSPR GOES HERE

END

DVENTR

COMMENT*****DATA FOR THE SPECIAL INPUT PROCESSOR DVENTR GOES HERE

END

DUTLIN

COMMENT*****DATA FOR THE SPECIAL INPUT PROCESSOR DUTLIN GOES HERE

END

/*

//

9-9

field on a card may be terminated by a slash(/) after which comments may be supplied on that card. The trailing comments, however, will appear in the listing, since all card images are printed, but will not be included in the data file.

Comments enclosed in dollar signs (\$)

\$ COMMENT \$

may also be inserted freely between datum entries. Except for listing the card images, such comments are ignored by the processor. The initial \$ terminates an immediately preceding data entry.

Card fields for data entries are limited to 72 columns.

3. Control Options. Control options available in the data field are repeat, skip, interpolation, and section repeat.

a. Repeat Option. The repeat option is an unsigned integer immediately followed by the letter "R". Use of this option permits repeating the previously entered data item an integer number of times. The specification nR means n total entries of the previous entry, for example

1 2R = 1 1, and

2.0 3R = 2.0 2.0 2.0

b. Skip Option. The skip option is an unsigned integer immediately followed by the letter "S". This means that the integer number of loca-

the catalog specification. It identifies logical unit 32 as a 9 track labeled tape. The parameter NNNNNN should be replaced by the actual tape number for the 360/195 or the logical unit number (32) for the 360/91. DDD should be replaced by NEW if the tape is to be written, or OLD if the tape is to be protected from being written, and SOME.NAME should be replaced by an identifying name. Labeled tapes are required on the 360/195 and are recommended in general. If an overriding data set identifies a disk other than a scratch disk (SYSDA), the DSN parameter must also be included.

All cards between //GO.SYSIN DD * card and the /* card are data cards for the run. The first four are the control module instructions described in Table 4.3. The fourth card describes the calculational path as follows:

- 1) The Input Processor module is accessed, for example, to read cards and write the nuclide ordered cross section file ISOTXS on the tape unit 32,
- 2) The special input processor DCRSPR (see Sec. 11) is accessed to read input data and write the necessary interface files for the Cross Section Processor module,
- 3) Then the module for processing cross sections is accessed, to convert the nuclide ordered file ISOTXS to the group ordered file GRUPXS,
- 4) The special input processor DVENTR is then accessed to read input data and write the necessary interface files for the VENTURE Neutronics module,
- 5) The VENTURE Neutronics module is then accessed,
- 6) The special processor DUTLIN is accessed to read data and write the

^a This is normal order on the 360/195. On the 360/91, FT06F001 is normally printed last.

9-10

c. Interpolation Option. The interpolate (nI) specification provides for linear interpolation between two REAL constants. The previously processed word is the initial end-point, n is the number of equally spaced points to be inserted between the initial and terminal end-points, and the terminal end-point follows the nI specification. Integer interpolation is not available. For example:

1.0 3I 5.0 = 1.0 2.0 3.0 4.0 5.0

d. Section Repeat Option. A section is a group of entries enclosed in parentheses. The entries may consist of numeric or Hollerith data; repeat, interpolate, and skip specifications; or section repeats. The section is always used in conjunction with the repeat option in the form

(entries) nR .

The entries are expanded to obtain n identical data sets.

The end of an interpolation section entry must lie within the section. The initial point may immediately precede the section, although this is seldom useful. The skip option as a section entry must be used cautiously since items skipped are taken from the array being overlayed and the overlayed array is the entity repeated in the section repeat operation. Sections may be continued over more than one card, and section repeats may be nested within section repeats up to 10 levels of nesting. An example of nesting is:

(1 (5.0 2I 20.0) 2R WORD1) 2R

which is equivalent to

in the system. The user should refer to specific documentation about these requirements. Information about specifying these parameters is given here and Table 8.3 presents reference information.

When a file has special requirements, such as for an unusually large amount of disc space, it may be most practical to include override instructions for this unit rather than use the parameters.

B1 and B2 are the sizes in bytes (4 bytes per short word) of core memory blocks into which data are read, limited to 32000. The larger this block size the fewer are the I/O requests required to read a particular data set. B2 defines the block size of the data sets which are read most often during the iterative part of the calculation. NB1 and NB2 (permissible values of 1 or 2) specify the number of buffers (number of blocks of size B1 or B2) of core memory into which data will be stored.

We recommend specifying NB1 = 1, NB2 = 1, B1 = 3520, and B2 = 32000, NX = 2, NS = 50, and N1 = 100. These values will be adequate for most practical problems. We find advantage in using only one buffer which minimizes core memory requirements. For small problems, it may be desirable to set B2 to something smaller than 32000 to minimize core memory. It is recommended that B1 and B2 not be set smaller than 3520. Values for parameters N2 through N16 are calculated as follows where reference is to the multi-dimensional problem to be solved by the neutronics code:^a

^aSpace requirements are shown for "track overflow"; they increase by a factor equal to the reciprocal of the fraction utilization without this.

9-11

Comments

The following comments should be heeded:

1. A header card is required.
2. The standard input processor will only accept cards or card-images written in the EBCDIC character set. The following characters are different from the BCD characters:

(is 12-5-8 punch
) is 11-5-8 punch
+ is 12-6-8 punch
= is 6-8 punch
3. Interface file CONTRL is written initially by the driver. Local special processors which use data from the input stream add or replace records in this file (unless in a stand-alone mode). The standard processor will write a completely new file. All records on this file are designated 1D. The records may be in any order except as noted below:

PROINS	always first
DVRINS	always present
(others)	optional
* * or BLANK	always last, more than four characters required.
4. The block of data for the input processor is terminated with a card containing STOP and another card containing END.

For a non-hexagonal problem running in the "plane stored" mode, with 48 columns, 22 rows, 16 planes, and 6 groups, N2=257, N3=1, N4=1, N5=113, N6=10, N7=5, N8=5, N9=43, N10=43, N11=54, N12=1, N13=1, N14=1, N15=10, and N16=1. Larger values for these parameters waste computer resources and likely increase cost.

Major space requirements for data files are edited by the more sophisticated codes such as VENTURE in this system as a user aid.

Regarding Very Large Space Requirements

When there is inadequate space on a disc (approximately 14,000,000 bytes on the IBM 360/91 at X-10, approximately 40,000,000 bytes on the IBM 160/195 at K-25) to contain a file, special action must be taken. Available space is impacted by dedication to system use and by other jobs in execution and the state of allocated storage due to the accumulated history of use. Major requirements can best be satisfied on the weekend.

If a file requires more than one disc, spanning is required and clean discs should be used. The user may contact "Programming Assistance" at the appropriate site (X-10 or K-25) for information about using dedicated discs and job control instructions for use with provision for spanning.

10-1

10. FILE EDITOR

The file editor was developed at LASL and modified extensively for local use. It may be used to print in card-image form the contents of interface data files, or to generate a file containing stacked interface data files, formatted for direct recovery by the input processor. The file editor requires instructions from the interface data file FPRINT when summoned in the calculational path, which must have previously been generated by the input processor. The contents of FPRINT describe the action to be taken by the file editor and are documented in Table 10.1. A unit which is to be written for future reading by the input processor must have compatible job control instructions including formatted records of 80 byte length. Files which can be edited were shown in the previous section, Table 9.1.

The maximum record length which can be processed by the file editor is 30,000 short words (15,000 long words). Also, only 10 different record types are now allowed, limiting use in a few instances.

The instruction file FPRINT is deleted from the file management table after use. Therefore a new file must be generated by use of the input processor and user input data for each access of the file editor. Of course, once a file is generated, it is available for use until it has been used and deleted.

When data is written on a file for recovery, writing begins at the

	4Z	G		Fixed source by zone and group.
	---			Set N5=1 for the "row-stored mode only.
N5	4(8CR+C+R) 4(9CR+2C+2R)	PG	B2	Non-hexagonal geometry. Hexagonal geometry.
N6				Set N6=10
N7	8CR	P	B2	
N8	8CR	P	B2	
N9	8CR	P	B1	
N10	8CR	P PG	B1	Normally. If initial flux guess is to be read from RTFLUX interface and to be expanded.
N11	8C	RP	B1	
N12				Set N12=1.
N13	---			Set N13=1 unless the power density interface PWDINT will be written.
N13	8CR	P	B1	For writing the interface PWDINT.
N14	---			Set N14=1 unless the P ₁ calculation is to be done.
N14	8CR	PG	B1	For the P ₁ calculation.
N15				Set N15=10.
N16	---			Normally N16=1.
N16	8CR	PG	B1	If the standard flux interfaces RTFLUX and/or ATFLUX are to be written on scratch disc.

11-1

11. ABOUT SPECIAL PROCESSORS

Special processors are allowed in this computational system. All true system member processors are assigned the path access number 2 and each has a unique eight-character name associated with it (six characters ending with two blank characters allowing identification at the user level with six characters). Other special processors may be added which are assigned the path access number 99.

The following four special processors are presently in the system:

DCRSPR - reads fixed format input data and writes interface files for the Cross Section Processor module.

DVENTR - reads fixed format input data and writes interface files for the VENTURE and VANCER neutronics modules.

DUTLIN - reads fixed format input data and adds or replaces records on the interface file CONTROL.

DCMACR - reads fixed format input data and writes a CITATION format pseudo microscopic cross section file.

The fixed format input data read by each of these processors is described elsewhere.⁵ Additional special processors will be added with development effort and reliability testing to ensure proper performance in this system. The special processors named above are frozen members of the system and cannot be overridden by the user supplying his own modules with the names indicated.

The catalog contains no provision to identify external files, ones supplied or those to be saved. This permits simple use when all of the data needed is in the user input data stream and no files are to be saved. In typical application, data files are both supplied and saved; therefore, appropriate information about these units must be supplied with the job control instructions. Indeed, considerable care must be taken to present instructions which are compatible with this code system, with its scheme of managing data files, and with the control module input instructions which also play a role.

The use of a single buffer is generally recommended for most purposes.

END OF SECTION

11-2

For each special processor module referenced by path access number 2 or 99 in the computational path (IM(I), Table 4.3), the reference module name followed by any required input data must appear in the input stream, with an END card termination.

When all of the calculations specified on the computational path card have been completed, the user may execute any other modules previously loaded as desired by including cards with =NAME punched beginning in column 1, where NAME is the name of the module to be accessed. Care must be taken to access the necessary data files on proper logical units, which may require that special action be taken for the run. Likely these logical units must be specified as input to the module, and input data requirements must be satisfied.

One or more special processors can be added to the system for local use. The module must be put in a data set with the name to be used for access (path access number 99), eight characters (six characters plus two blanks recommended).

Special care should be taken if one or more data sets are concatenated onto the set containing codes used routinely: do not destroy the production codes, as by improper disposition. All the data sets used should be considered to be permanent (don't DELETE anything, avoiding accidental destruction of production codes). Avoid action which prevents quality assurance of the production codes. Make load modules in computer runs separate from runs using the production code.

The following general comments are offered on the user input data:

1. Input for the control module must come first,
2. Requirements of the special processors must be met within the constraints of their actions and the actions of the input processor when it is used,
3. Care must be exercised to satisfy the requirements of the computational code modules, so that desired actions are taken regarding the reference tables of the interface data files and assigned logical unit numbers,
4. A header card must proceed the block of input data for the input processor; an identifying note may be put on this card since the contents are not used for control.

Quite generally it is assumed that special processors use the file management tables to determine the logical unit numbers associated with the file names and generate the files without altering these tables, except to add new files. Thus they play the game fairly. However, it is probable that some which don't will come into use, and their use must be carefully restricted. Any specific special processor can be used for some function, but a code module may carry out its assigned task after this point only if the actions taken allow continued communication of consistent data about the interface data files. Generally it is possible to use a special processor at the start of a run or at the end of a run successfully. Special processors are identified by the header record of a block of data in the user input data file and access of a special processor is indicated by

12-1

12. SAMPLE USER INPUT AND EDIT

Information about four cases, separate computer runs, is presented here to illustrate the available capability and its use. Certain aspects are addressed in the following discussion.

Case 1

Presented in Table 12.1 is a set of user input data as listed from cards. The input data to the local IBM-360/195 computer was identical to this listing. This run used the input processor which treats free-form input data.

The job control instructions are quite simple for this run since the catalog procedure described earlier provides sufficient logical units and few of the data blocking parameters need to be changed. Entrance to the IBM-360/195 computer located at the K-25 site was from a remote terminal connected to the IBM-360/91 computer located at the X-10 site some six miles away. The /91 computer has indirect access to the /195 through a PDP-10 computer identified as "REMOTES" also located at X-10. The first card instructs the /91 to relay the card images to "REMOTES", and the second card instructs "REMOTES" to send them on to the /195. The job control instructions are followed by the user input data to the codes beginning with a card identifying the control module to be used, here CONTROL1 preceded by *. The calculational path through the code modules and certain reference informat

2. If the file name is not in the table, it is added and associated with the next higher unit number in the allowed set, and the new file is written.
3. If a file by this name was already generated in this access to the input processor, a new one and an associated next higher version number are added to the table, and a new file is written.

A 99V flag in front of the file name forces a new file to be created as a special option implemented locally.

The input processor has the capability to update an existing file, useful capability not extensively tested locally. Care must be taken to count REAL*8 words as double entries, as when A6 names are involved, on short word machines. We consider that simple changes are possible, but complicated changes, as for data records in a loop structure, require special consideration: there are rather severe limitations such as the fact that record lengths cannot be changed. Thus we generally recommend against updating with new data lacking explicit description of the constraints, or only with careful testing prior to costly calculations.

A general discussion about the user input data requirements for the input processor is presented below, a modification of the instructions from LASL to emphasize user considerations. Example use of the input processor is shown in Sec. 12. A list of the files which can now be generated is given in Table 9.1.

12-2

TABLE 12.1. INPUT FOR CASE 1.

```

/*RELAY      REMOTES
95
//USERID JOB (CHARG), 'USER ADDRESS - NAME',
// MSGLEVEL=(1,1)
// *CLASS CPU95=60S, REGION=300K, IO=10, LINES=5, CARDS=0
//STEP EXEC BOLDVENT,
// NB1=1, NB2=1, B1=3520, B2=15360, B1=2, B5=50, B1=100,
// B2=8, B3=10, B4=1, B5=8, B6=10, B7=1, B8=1, B9=2, B10=2, B11=2, B12=1, B13=1,
// B14=1, B15=10, B16=8,
// GOSIZE=300K
//GO.SYSIN DD *
=CONTROL1
THIS TITLE WILL BE REPLACED BY THE TITLE READ BY THE INPUT PROCESSOR.
15000
1 6 7 0
END
THE INPUT PROCESSOR FREE FORM DATA.
OV CONTRL
1D PROINS /
* PERIODIC BOUNDARY IN THETA-R GEOMETRY, CASE A4 WITH BLACK ABSORBER *
* 24X20X3 GROUP, 1440 POINTS PWRD. * ADJ. *
*GUCRUN* *GUCRUN* 0.0 74R 0 100R
1D DVRINS 0 0 100R 15000 0 99R
1D XCPINS 0.0 100R 0 0 0 1 0 18R 1 1 0 76R
1D DTHINS 0.0 3R 1.0 3R 0.0 1.6 0.0 0.0 1.0-6 1.0-5 0.0 88R
0 3R 1 0 1 0 1 0 11R 4 0 6R 1 1 1 0 1 1 0 1 1 1 1 1 0 0 5 1
0 0 0 0 1 0 1 0 0 0 0 1 0 42R
1D * * 0.0 100R 0 100R
OV ISOTIS
1D 3 3 0 1 0 1 1 1
2D /
*CITATION MACROS TO MICROS
*ZONE1* *ZONE2* *ZONE3* 0.6 0.4 0.0 0.0 3R 0.0 4R 0 3 6
4D (*ZONE1*) 3R 0.0 1.0 0.0 4R 0 1 1 0 5R 1 1 0
0 1 1 2 2 1 1 1
5D 1.987677-1 3.833180-1 9.962149-1 0.0 3R
0.0 2.864-2 1.132-1 0.0 8.0-14 1.0-12
0.0 9.4275+10 1.088+11 0.6 0.4 0.0
7D 0.0 0.0 4.378-2 0.0 4.295-2
4D (*ZONE2*) 3R 0.0 6R 0 8R 1 1 0
0 1 1 2 2 1 1 1
5D 1.8205-1 2.745745-1 3.333333-1 0.0 3R
1.3-2 8.0-3 0.0
7D 0.0 0.0 4.3-1 0.0 2.5-3
4D (*ZONE3*) 3R 0.0 6R 0 8R 1 1 0

```

NDXSRF	Nuclide referencing	x	x
ZMATDN	Nuclide concentrations	x	x
SEARCH	Criticality search data	x	x
SNCOMS	S _n cosines, quadratures		x
FLXSRC	Fixed source	x	x
RTFLUX	Regular total point flux	x	x
ATFLUX	Adjoint total point flux	x	x
RAFLUX	Regular angular flux		x
AAFLUX	Adjoint angular flux		x
RZFLUX	Regular total zone flux	x	x
RCURNT	Regular current		x
ACURNT	Adjoint current		x
PWDINT	Point power density		x
CONTRL	Computation module control records	x	x
CXSPRR	Cross section processing data	x	x
PERTUB	Forward, adjoint integrals	x	x
FISSOR	Fission source	x	x
EXPOSE	Exposure data	x	Mo
ZCONCA	Local nuclide concentrations	x	Mc
RODSET	Control rod positioning data	x	Mo
FPRINT	File editor instructions	x	Mo

12-3

```

1 6 3 1 0 1 0 5R /
3D /
    0.0 1.707963-1 1.570796 1.741593 3.141593
    0.0 30.0 35.0 45.0 70.0
4 8 4 8 /
6 4 4 6 /
5D /
7.685832+1 6.3+2 7.685832+1 6.3+2 2.775439+1 2.275+2
2.775439+1 2.275+2 6.831851+1 5.6+2 6.831851+1 5.6+2
2.455197+2 2.0125+3 2.455197+2 2.0125+3
1.0-3 /
0.0 3R 0.4692 0.0 2R 0.0 2R 0.4692 /
2 1 1 1 /
1 4R 2 1 2 1 5R 3 4R
7D /
(1 4R 2 8R 3 4R 4 8R) 6R /
(5 4R 6 8R 7 4R 8 8R) 4R /
(9 4R 10 8R 11 4R 12 8R) 4R /
(13 4R 14 8R 15 4R 16 8R) 6R /
0V NDXSRF
1D /
3 1 3 3 3 0
2D /
(*ZONE1* *ZONE2* *ZONE3*) 2R 0.0 6R 9 3R 3 0 0 0
(1 2 3) 2R
3D /
3.125352+3 5.550879+1 4.516035+3 1.0 3R 1 3R
0V ZMATDN
1D /
0.0 0 3 3 1
2D /
1.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 1.0
STOP
END
/*
//

```

The card format has two distinct categories for entries on a card. These are (a) the file and record control words, and (b) the datum field for the actual data items to be loaded into a given array. The rules governing usage of these entries are given below.

File and Record Control Cards

There are four types of control cards governing the input of data which are addressed in the following discussion.

TABLE 12.2. CONDENSED EDIT FOR CASE 1.

-25 ON 300/1.0. CRITICAL MONITORING. JATT-01-02-77. TIME-13.01.00. JOURNALISTERN									
-REMAINING 1/20 0. CPU MIN. 3.00									
DATA									
BY THE FILE -040- BY THE INPUT -040-040-									
C 0 0 1 3 3 3									
END									
INITIAL 1/20 FILE MANAGEMENT TABLES									
BY SUPPLIED BY									

-040- 040-040- 040-040- 040-040- 040-040- 040-040- 040-040-									
1 - REMAINING 1/20 0. CPU MIN. 0.00									
2 - REMAINING 1/20 0. CPU MIN. 0.00									
3 - REMAINING 1/20 0. CPU MIN. 0.00									
4 - REMAINING 1/20 0. CPU MIN. 0.00									
5 - REMAINING 1/20 0. CPU MIN. 0.00									
6 - REMAINING 1/20 0. CPU MIN. 0.00									
7 - REMAINING 1/20 0. CPU MIN. 0.00									
8 - REMAINING 1/20 0. CPU MIN. 0.00									
9 - REMAINING 1/20 0. CPU MIN. 0.00									
10 - REMAINING 1/20 0. CPU MIN. 0.00									
11 - REMAINING 1/20 0. CPU MIN. 0.00									
12 - REMAINING 1/20 0. CPU MIN. 0.00									
13 - REMAINING 1/20 0. CPU MIN. 0.00									
14 - REMAINING 1/20 0. CPU MIN. 0.00									
15 - REMAINING 1/20 0. CPU MIN. 0.00									
16 - REMAINING 1/20 0. CPU MIN. 0.00									
17 - REMAINING 1/20 0. CPU MIN. 0.00									
18 - REMAINING 1/20 0. CPU MIN. 0.00									
19 - REMAINING 1/20 0. CPU MIN. 0.00									
20 - REMAINING 1/20 0. CPU MIN. 0.00									
21 - REMAINING 1/20 0. CPU MIN. 0.00									
22 - REMAINING 1/20 0. CPU MIN. 0.00									
23 - REMAINING 1/20 0. CPU MIN. 0.00									
24 - REMAINING 1/20 0. CPU MIN. 0.00									
25 - REMAINING 1/20 0. CPU MIN. 0.00									
26 - REMAINING 1/20 0. CPU MIN. 0.00									
27 - REMAINING 1/20 0. CPU MIN. 0.00									
28 - REMAINING 1/20 0. CPU MIN. 0.00									
29 - REMAINING 1/20 0. CPU MIN. 0.00									
30 - REMAINING 1/20 0. CPU MIN. 0.00									
31 - REMAINING 1/20 0. CPU MIN. 0.00									
32 - REMAINING 1/20 0. CPU MIN. 0.00									
33 - REMAINING 1/20 0. CPU MIN. 0.00									
34 - REMAINING 1/20 0. CPU MIN. 0.00									
35 - REMAINING 1/20 0. CPU MIN. 0.00									
36 - REMAINING 1/20 0. CPU MIN. 0.00									
37 - REMAINING 1/20 0. CPU MIN. 0.00									
38 - REMAINING 1/20 0. CPU MIN. 0.00									
39 - REMAINING 1/20 0. CPU MIN. 0.00									
40 - REMAINING 1/20 0. CPU MIN. 0.00									
41 - REMAINING 1/20 0. CPU MIN. 0.00									
42 - REMAINING 1/20 0. CPU MIN. 0.00									
43 - REMAINING 1/20 0. CPU MIN. 0.00									
44 - REMAINING 1/20 0. CPU MIN. 0.00									
45 - REMAINING 1/20 0. CPU MIN. 0.00									
46 - REMAINING 1/20 0. CPU MIN. 0.00									
47 - REMAINING 1/20 0. CPU MIN. 0.00									
48 - REMAINING 1/20 0. CPU MIN. 0.00									
49 - REMAINING 1/20 0. CPU MIN. 0.00									
50 - REMAINING 1/20 0. CPU MIN. 0.00									
51 - REMAINING 1/20 0. CPU MIN. 0.00									
52 - REMAINING 1/20 0. CPU MIN. 0.00									
53 - REMAINING 1/20 0. CPU MIN. 0.00									
54 - REMAINING 1/20 0. CPU MIN. 0.00									
55 - REMAINING 1/20 0. CPU MIN. 0.00									
56 - REMAINING 1/20 0. CPU MIN. 0.00									
57 - REMAINING 1/20 0. CPU MIN. 0.00									
58 - REMAINING 1/20 0. CPU MIN. 0.00									
59 - REMAINING 1/20 0. CPU MIN. 0.00									
60 - REMAINING 1/20 0. CPU MIN. 0.00									
61 - REMAINING 1/20 0. CPU MIN. 0.00									
62 - REMAINING 1/20 0. CPU MIN. 0.00									
63 - REMAINING 1/20 0. CPU MIN. 0.00									
64 - REMAINING 1/20 0. CPU MIN. 0.00									
65 - REMAINING 1/20 0. CPU MIN. 0.00									
66 - REMAINING 1/20 0. CPU MIN. 0.00									
67 - REMAINING 1/20 0. CPU MIN. 0.00									
68 - REMAINING 1/20 0. CPU MIN. 0.00									
69 - REMAINING 1/20 0. CPU MIN. 0.00									
70 - REMAINING 1/20 0. CPU MIN. 0.00									
71 - REMAINING 1/20 0. CPU MIN. 0.00									
72 - REMAINING 1/20 0. CPU MIN. 0.00									
73 - REMAINING 1/20 0. CPU MIN. 0.00									
74 - REMAINING 1/20 0. CPU MIN. 0.00									
75 - REMAINING 1/20 0. CPU MIN. 0.00									
76 - REMAINING 1/20 0. CPU MIN. 0.00									
77 - REMAINING 1/20 0. CPU MIN. 0.00									
78 - REMAINING 1/20 0. CPU MIN. 0.00									
79 - REMAINING 1/20 0. CPU MIN. 0.00									
80 - REMAINING 1/20 0. CPU MIN. 0.00									
81 - REMAINING 1/20 0. CPU MIN. 0.00									
82 - REMAINING 1/20 0. CPU MIN. 0.00									
83 - REMAINING 1/20 0. CPU MIN. 0.00									
84 - REMAINING 1/20 0. CPU MIN. 0.00									
85 - REMAINING 1/20 0. CPU MIN. 0.00									
86 - REMAINING 1/20 0. CPU MIN. 0.00									
87 - REMAINING 1/20 0. CPU MIN. 0.00									
88 - REMAINING 1/20 0. CPU MIN. 0.00									
89 - REMAINING 1/20 0. CPU MIN. 0.00									
90 - REMAINING 1/20 0. CPU MIN. 0.00									
91 - REMAINING 1/20 0. CPU MIN. 0.00									
92 - REMAINING 1/20 0. CPU MIN. 0.00									
93 - REMAINING 1/20 0. CPU MIN. 0.00									
94 - REMAINING 1/20 0. CPU MIN. 0.00									
95 - REMAINING 1/20 0. CPU MIN. 0.00									
96 - REMAINING 1/20 0. CPU MIN. 0.00									
97 - REMAINING 1/20 0. CPU MIN. 0.00									

A file control card is the first card read by the processor. Processing of the named file continues until a new file control card or a STOP card (see below) is encountered. Any number of files may be processed in any order.

As currently implemented, when $n=0$, the latest version of the file, if known by the file management system to exist, will be rewritten; otherwise a new file will be created with a version number one higher than the highest number found, version one if there is no entry of that name in the tables. When the overlay option (n greater than 0) is used, the old file is purged and is not available for further use, and the new file has the version number of the old one, but it is placed on a new logical unit.

2. Record Type Control Card - mD. An unsigned integer immediately followed by the alphabetic character "D" denotes a "Data" card. The first word on a card following a file control card must be an mD. Thereafter, if a card is encountered without the first word being an identifier, then the last read identifier is used. The data on this card may be integer, floating point, Hollerith, etc., with any mixture being possible, but the actual requirements set out in the data specifications for the specific file must be satisfied.

The mD are associated with the record types appearing in the interface file specifications. Each unique data list in the specifications is

12-5

Case 2

This case involves solution of a reference two-dimensional neutronics problem for an HTGR reactor. Information from the computer run is shown in Table 12.3 which is a listing of the input data cards, and the condensed results from the run in Table 12.4. The detailed edit from the codes is not shown.

Case 3

This case is an example of the use of special job control instructions to illustrate the processing of data (nuclide-ordered cross section file ISOTXS) which exists in a file separate from the user input data stream, by the input processor. The input instructions shown in Table 12.5 effect access to the control module, and supply data for the input processor as shown in the listing. The product from this run is a binary ISOTXS file, saved on tape.

Case 4

This example treats a reactor history through two cycles with refueling at the end of the first cycle. The reactor description and its history are somewhat idealized for simplicity. The capability for refueling reactor zones and to apply a prescribed control rod insertion schedule (which may differ from one cycle to another), is illustrated here by specifications for the desired nuclide concentrations at specific

TABLE 12.3. INPUT FOR CASE 2.

```

//YOURNAME JOB ('7777'), 'ADDRESS AND NAME',
// NSLEVEL=(1,1)
// *CLASS CPU95=0405, IO=007, REGION=0380K, LINES=005, CARDS=010
//XWHY EXAC BOLDVENT,
// N81=1, N82=1, B1=3520, B2=7200, N1=10, N2=10, N3=20, N4=10, N5=20, N6=10,
// N7=10, N9=10, N9=10, N10=10, N11=10,
// N12=10, N13=10, N14=10, N15=10, N16=10, N17=20, N18=30, N19=10,
// NOSIZE=390K
//GO,SYSLIN DD *
=CONTROL1
HTGR BENCHMARK
32000 0
1 6 7 1 0
END
LASLIN
OV CONTRL
1D PROINS /
*HTGR BENCHMARK 2-D TRIANGULAR GEOMETRY & GROUPS 60 DEG ROTATIONAL SYM *
* USING LASL INPUT PROCESSOR *
*GMCRUN* *GMCRUN* 0.0 76R 0 10GR
1D DVINS 0.0 10GR 32000 0 99R
1D XCPINS 0.0 10GR 0 0 0 1 0 19R 1 0 76R
1D DTNINS 0.0 3R 1.0 1.0 0.166667 0.0 0.0 0.0 0.0
1.0-6 1.0-5 0.0 4R
0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 80
0 0 0 0 0 0 0 0 2 1 1 1 1 0 1 0 1 0 0 0
0 0 0 0 0 0 0 0 0 0 0 1 1 1 0 0 0 0 0 0
0 50R /
1D * 0.0 10GR 0 10GR
OV ISOTXS
1D 8 8 0 1 0 1 1 1
2D /
* 2-D HEXAGONAL HTGR BENCHMARK CROSS SECTIONS (MACROSCOPIC) *
*CGMPSA * *CGMPSB * *CGMPC * *CGMPD * *CGMPE * *CGMPF * *CGMPG * *CGMPZ *
0.9675 0.0325 0.0 0.0
0.0 4R
1.49+7 1.43+5 1.76+1 2.3R 1.0-5
0 1 6 9 12 15 18 21
4D (*MACRO *) 3R 0.0 6R 0 0 1 0 5R 1 1 0
0 1 1 2 2 2 1 1 1 1
5D /
0.1257862 0.2433070 0.2487562 0.2544529
0.0 4R
1.99-4 4.55-3 9.12-3 8.93-3
2.25-5 1.49-4 4.09-4 1.97-3
2.4R 2.422819 2.421569 2.416244
7D 0.0 0.0 1.09-2 0.0 3.2R-3 0.0 2.04-2
4D (*MACRO *) 3R 0.0 6R 0 0 1 0 5R 1 1 0
0 1 1 2 2 2 1 1 1 1

```

DRIVER1
CONTRLT
CONTRLN
CONTRLP
DRIVER2
DRIVERS

CHI
VEL
ENERGY

SIGTR
SIGT
SIGC
SIGP
MU
SIGS

(CONT)

SD /	0.1257862 0.2433090 0.2487562 0.2544529	SIGTR
0.0 4R		SIGT
3.61-5 7.69-4 4.28-4 8.70-4		SIGC
2.25-5 1.49-4 4.08-4 1.97-3		SIGP
2.48 2.422819 2.421569 2.416244		NU
7D 0.0 0.0 1.09-2 0.0 3.28-3 0.0 2.04-2		SIGS
4D (*HACRO *) 3R 0.0 6R 0 0 1 0 5R 1 1 0		
0 1 1 2 2 2 1 1 1 1		
SD /	0.1418440 0.2754821 0.2801120 0.2873563	SIGTR
0.0 4R		SIGT
5.82-5 1.36-3 9.41-4 1.89-3		SIGC
3.95-5 2.62-4 7.18-4 3.46-3		SIGP
2.488608 2.427481 2.423398 2.424855		NU
7D 0.0 0.0 1.23-2 0.0 3.67-3 0.0 2.28-2		SIGS
4D (*HACRO *) 3R 0.0 6R 0 0 1 0 5R 1 1 0		
0 1 1 2 2 2 1 1 1 1		
SD /	0.1418440 0.2754821 0.2801120 0.2873563	SIGTR
0.0 4R		SIGT
5.08-5 1.43-3 7.77-4 1.87-3		SIGC
3.76-5 2.36-4 6.47-4 3.12-3		SIGP
2.476064 2.427966 2.426584 2.423077		NU
7D 0.0 0.0 1.23-2 0.0 3.67-3 0.0 2.28-2		SIGS
4D (*HACRO *) 3R 0.0 6R 0 0 1 0 5R 1 1 0		
0 1 1 2 2 2 1 1 1 1		
SD /	0.1418440 0.2754821 0.2801120 0.2873563	SIGTR
0.0 4R		SIGT
6.23-5 1.46-3 7.38-4 1.86-3		SIGC
3.63-5 2.20-4 6.03-4 2.91-3		SIGP
2.471074 2.427273 2.421227 2.422680		NU
7D 0.0 0.0 1.23-2 0.0 3.67-3 0.0 2.28-2		SIGS
4D (*HACRO *) 3R 0.0 6R 0 0 1 0 5R 1 1 0		
0 1 1 2 2 2 1 1 1 1		
SD /	0.1418440 0.2754821 0.2801120 0.2873563	SIGTR
0.0 4R		SIGT
6.40-5 1.50-3 6.99-4 1.85-3		SIGC
3.51-5 2.04-4 5.59-4 2.70-3		SIGP
2.444387 2.426471 2.432916 2.422222		NU
7D 0.0 0.0 1.23-2 0.0 3.67-3 0.0 2.28-2		SIGS
4D (*HACRO *) 3R 0.0 6R 0 0 1 0 5R 1 1 0		
0 1 1 2 2 2 1 1 1 1		
SD /	0.2032520 0.3921569 0.4006410 0.4069089	SIGTR
0.0 4R		SIGT
1.39-5 2.18-6 1.97-5 1.06-4		SIGC
0.0 4R		SIGP
0.0 4R		NU

(CONT)

7D 0.0 0.0 1.77-2 0.0 5.33-3 0.0 3.31-2
 4D (*MACRO *) 3R 0.0 0 0 1 0 5R 1 1 0
 0 1 1 1 1 1 1 1

SIGS

5D /

0.0 4R
 0.0 4R
 0.0 4R
 0.0 4R
 0.0 4R

SIGR
 SIGT
 SIGC
 SIGP
 RU

7D 0.0 0.0 0.0 0.0

0V GEODS?

1D 9 8 8 1 1 1 1 38 19 1
 2 4 2 4 0 0 1 0 1 1
 1 1 0 0 0 0 0

3D 0.0 397.1 0.0 397.1 38 19

5D /

0.0 8R
 0.0 0.4692 8 1 8R 1 2 3 4 5 6 7 8

7D /

8 31R 7 3R 8 3R 7
 8 27R 7 11R
 8 26R 7 12P
 8 25R 7 13R
 8 24R 7 14R
 8 23R 7 12R 5 3R
 8 19R 7 12R 2 3R 5 3R 3
 8 18R 7 9R 6 3R 2 3R 3 5R
 8 17R 7 9R 6 3R 3 9R
 8 16R 7 9R 4 3R 3 10R
 8 15R 7 6R 6 3R 4 3R 3 6R 2 3R 3 2R
 8 11R 7 9R 6 3R 2 3R 3 6R 2 3R 3 3R
 8 10R 7 9R 5 3R 2 3R 3 13R
 8 9R 7 9R 5 3R 3 17R
 8 8R 7 9R 4 3R 3 15R 2 3R
 8 7R 7 6R 6 3R 4 3R 3 6R 1 3R 3 6R 2 3R 3
 8 3R 7 9R 6 3R 2 3R 3 6R 1 3R 3 11R
 8 2R 7 9R 5 3R 2 3R 3 21R
 8 7 9R 5 3R 3 24R 1

0V HDISBP

1D 8 1 8 8 8 0

2D /

*CONPA * *CONPB * *CONPC * *CONPD * *CONPE * *CONPF * *CONPG * *CONPZ *
 (*MACRO *) 8R 0.0 8R 0.0 8R 0 8R 8 0 0 0
 (1 2 3 4 5 6 7 8) 2R

3D /

0.0 8R /
 1.0 8R 1 8R

0V ZWATDN

1D 0.0 0 8 8 1

(CONT)

```

2D /
  1.0 0.0 7R
  0.0 1.0 0.0 6R
  0.0 2R 1.0 0.0 5R
  0.0 3R 1.0 0.0 4R
  0.0 4R 1.0 0.0 3R
  0.0 5R 1.0 0.0 2R
  0.0 6R 1.0 0.0
  0.0 7R 1.0
OV FPRINT
ID 4 0 0 0
2D *RTFLUX* 0 0 0
2D *FISSON* 0 0 0
2D *RIFLUX* 0 2 0
2D *FWDINT* 0 0 0
STOP
END
/*
//

```

DRIVER2

TABLE 12.4. CONDENSED EDIT FOR CASE 2.

BOLD VENTURE VERS-2 RUN AT K-25 ON 360/195. CONTROL MOD=CONTROL1, DATE=03-02-77, TIME=10.04.25, JOBNAME=TYPE

INITIALIZATION. DIFFERENCE - REMAINING I/O= 6. CPU MIN= 0.66

RUN TITLE AND CONTROL MODULE DATA

HTER BENCHMARK

32000 0 0 0 0 0 0 0 0 0 0 0 0 0 0
1 6 7 3

MODULES TO BE ACCESSED IN ORDER

1 6 7 3

INITIAL I/O FILE MANAGEMENT TABLES
FILE NAME SUPPLIED BY

FILE NUMBER	SECT	USER-EXIST	USER-STACK	VERSION	WRITTEN
10	CONTRL			1	1
11	GROUPS			1	0
12	GRDOST			1	0
13	MODSRP			1	0
14	JNATON			1	0
15	SEARCH			1	0
16	RSTRTN			1	0
17	RTFLUX			1	0
18	ATFLUX			1	0
19	RZFLUX			1	0
20	PRODINT			1	0
30	CRSPRN			1	0
31	ISOTAS			1	0
32	ISOTAS			2	0
33	FLISNC			1	0

READ DATA AND ACCESS MODULE 1 - REMAINING I/O= 6. CPU MIN= 0.66

ACCESS MODULES 6 7 3 0 0 - REMAINING I/O= 9. CPU MIN= 0.83

ITERATIONS, CONVERGENCE, SEARCH, PEAK PRODUCTION. K - 23 0.995260-06 0.0 3.995400-05 1.1206013

FINAL I/O FILE MANAGEMENT TABLES
FILE NAME SUPPLIED BY

FILE NUMBER	SECT	USER-EXIST	USER-STACK	VERSION	WRITTEN
10	CONTRL			1	1
11	GROUPS			1	1
12	GRDOST			1	1
13	MODSRP			1	1
14	JNATON			1	1
15	SEARCH			1	0
16	RSTRTN			1	0
17	RTFLUX			1	1
18	ATFLUX			1	0
19	RZFLUX			1	1
20	PRODINT			1	1
30	CRSPRN			1	0
31	ISOTAS			1	0
32	ISOTAS			2	1
33	FLISNC			1	0
34	FLISOR			1	1

WRAP-UP NOW COMPLETE. -- DRIVER TO SEEK NEW CASE - REMAINING I/O= 3. CPU MIN= 0.13 - TIME=10.11.25

TABLE 12.5. INPUT FOR CASE 3.

```

//TOURNAME JOB (77777), 'ADDRESS AND NAME',
// MSGLEVEL=(1,1)
//%CLASS CPU95=003H,IO=01%,REGION=0270K,LINES=000
//ARDI EXEC BOLDVENT,
// 92=3520,
// GOSIZE=270K
//GO.FT32P001 DD UNIT=TAPE9,VOL=SER=BBBBBB,LABEL=(1,SL),
// DISP=(NEW,KEEP),DCB=(RECFM=VBS,LRECL=X,BLKSIZE=3520),
// DSN=X.YOJD7777.ISOTIS
//GO.FT90P001 DD UNIT=TAPE9,VOL=SER=AAAAA,LABEL=(,ML),DISP=OLD,
// DCB=(RECFM=PB,LRECL=80,BLKSIZE=5120)
//GO.SYSIN DD *
=CONTROL1
EASTINGHORE CROSS SECTIONS - CARDS TO BINARY
10000 0
1 0
END
LASLIN
FLAG 90 2 1 4000
STOP NORMAL INPUT STREAMS
END
/*
//

```

DRIVER1
CONTROL
CONTROL
CONTROL
DRIVER2
DRIVERS

DRIVER2

Information about the problem is summarized below.

Neutronics

One-half core in R-Z geometry
 19 x 30 meshpoints, 3 groups (1710 space-energy points)
 20 exposure zones (core, axial and radial blankets)
 480 nuclide concentrations, cross sections for 45
 Control rods represented by a uniform distribution

History

First cycle then second cycle after refueling core and axial blanket
 Each cycle treated in two exposure steps
 Neutronics problems solved start, midpoint, and end each cycle
 Control rod concentrations prescribed

The path of the calculation is presented in Table 12.6. The individual system modules are accessed as shown. Information about the calculation is shown in Table 12.7. Table 12.8 presents a summary of the calculated results; it may be noted that the input data specifications have fortuitously caused near critical conditions to exist through the reactor history.

Table 12.9 is a listing of the user input data which includes the job control instructions; note that an available file of microscopic cross sections was used. Data files are generated for the initial neutronics problem with the special processor DVENTR (input data requirements documented elsewhere). The input processor is used to generate the file EXPOSE from data supplied for the exposure calculation. The input processor is used to change the nuclide concentrations as desired. To add instructions to the module control file CONTRL for the exposure and reaction rate modules, and to change the instructions to the neutronics code prior to its final access to request solutions for both the regular and adjoint problems, the special processor DUTLIN was used. This processor allows addition or replacement of one or more records without destroying other contents, without change in the version number of this file or its assigned unit number (input data requirements are documented elsewhere). Quite generally, the instructions to the modules were to

replace an old file when a new one is written, which minimizes file usage but prevents saving intermediate results. The exception is the instruction to the exposure module regarding the new concentrations, and the action of the input processor discussed below.

Each time of exposure is divided into two equal time steps. The neutron flux level is adjusted after the first step, as predicted from the results for the first step, to effect nearly the desired power level over the exposure period. This refinement is obtained at the cost of solving the chain equations twice. Following exposure, nuclide concentrations are obtained at two points in time under shutdown conditions. In accordance with instructions, the new nuclide concentrations to be placed on file are those following exposure, before shutdown, and a new file is to be created. The first nuclide concentration file (ZNATDN) is created from input data by the special processor DVENTR, version 1 on unit 16. Subsequent action taken by the input processor causes a new version 1 file to be written on unit 18 and the one on unit 16 is deleted. The exposure module then places a version 2 file on unit 16. The input processor is instructed to upgrade version 2 and generates a new version 2 file on unit 19, deleting the one on unit 16. The exposure module then places a version 3 file on unit 16. At the end of the calculation, versions 1, 2, 3, 4 and 5 exist on units 16, 19, 18, 30, and 31 respectively. Considerable care must be exercised to provide control over the generation, use and saving of files when there is more than one version in existence. Multiple versions should be avoided unless actually needed. Note that special job control instructions may be required to allocate space and assign logical units for files. The condensed printout for the case is presented as Table 12.10, the results dependent on the code version in use.

It should be noted that when a calculational module is accessed, it follows the instructions then available in its dedicated record in the file CONTRL. These instructions remain unchanged unless altered with special input data at some point in a calculation, after which change they remain as altered. These control instructions can be supplied or changed with the input processor, but there are severe constraints on its use. The position of a record to be changed must be known by the user, and the existing contents will be destroyed if it is simply generating a CONTRL file.

TABLE 12.6. PATH OF A DEPLETION CALCULATION

PROBLEM SETUP	(Group-ordered microscopic cross sections supplied)
	Control module accessed (reads reference data)
	Special processor (neutronics problem data)
	Special processor (control instructions for certain modules)
	Input processor (exposure data, revised nuclide concentrations)

FIRST CYCLE	Neutronics (T = 0 days)
	Exposure (150 days in 2 steps, also shutdown)
	Input processor (control rods half inserted)
	Neutronics (T = 150 days)
	Exposure
	Input processor (control rods removed)
	Neutronics (T = 300 days)
	Reaction rates

SECOND CYCLE	Input processor (refuel core, rod insertion)
	Neutronics (T = 300 days)
	Exposure
	Input processor (control rods half inserted)
	Neutronics (T = 450 days)
	Exposure
	Input processor (control rods removed)
	Special processor (control instructions for neutronics)
Neutronics (T = 600 days, regular, adjoint)	
Reaction rates	

Table 12.7. INFORMATION ABOUT A DEPLETION CALCULATION

Module access count	45
Neutron flux-eigenvalue problems solved	7
Exposure steps	4
Point nuclide concentrations are obtained	16
Processor time (min)	1.55
Clock time (min)	20.
Data transfer operations	11,300
Lines printed	4,500
Memory required (short words)	75 K

Neutronics module processor time (min)

Initialization	0.04
First problem, total	0.21
Subsequent problems, typical total	0.18
Total	1.34

Table 12.8. SUMMARY RESULTS FOR A DEPLETION CALCULATION

Time (Days)	Control Rod Position ^a	k_{eff} ^b	Peak Power Density (W/cc)	Blanket Power (Fraction)	Fissile Inventory (Kg)	Primitive Reaction Rate Breeding Ratio
<u>First Cycle</u>						
0	In	0.99813	696		2693	1.264
				0.0197		1.252
150	Half In	0.99802	666		2794	1.256
				0.0382		1.244
300	Out	0.99918	637	0.0508	2892	1.246
<u>Second Cycle After Refueling</u>						
300	In	1.00047	738		2889	1.244
				0.0312		1.237
450	Half In	0.99881	686		2982	1.247
				0.0507		1.239
600	Out	0.99956	643	0.0642	3077	1.246

^aControl rod insertion not the same for first and second cycles.

^bThe results depend on the level of convergence satisfied in terminating the iterative procedures, and also on the specific procedures applied; data shown here only approximately agree with those in Table 12.10 reflecting such differences.

TABLE 12.9. INPUT FOR CASE 4.

```

//USERID JOB (CHARG), 'USER ADDRESS - NAME',
// MSGLEVEL=(0,0)
//CLASS CP001=002H, IO=012, REGION=0320K, LINES=005, CARDS=0
//BURN EXEC BOLDVENT,
// BB1=1, BB2=1, B1=3520, B2=7200, B1=10, B2=60, B3=20, B4=10, B5=30, B6=10,
// B7=10, B8=10, B9=10, B10=10, B11=10,
// B12=10, B13=10, B14=10, B15=10, B16=20, B17=30, B18=10,
// COSIZE=320K
//GO. PT11P001 DB UNIT=3330, VOL=SER=2X0000, DISP=SHR,
// 95H=TDF-GRUPKS.LMPDR.G3.S2
//GO.SYSIN DD *
=CONTROL1
LMPDR RZ 3 GROUP, 1/2 CORE (NO REAL DESIGN) 30 X 19 POINTS
010000 1
2 2 1 7 13 -3 2 1 7 9 1 7 13 -3 2 1 2 7 9
GRUPKS
END
DVIEW?R
001
001 1-1
- .15 2500.0+6 0.97 0.5 1-2
1. -5 8. -5 1-3
1 0 0 0 44 1-4
1 0 3 2 0 1 1 0 1 1-5
003 1-6
7 1 2 2 1 0 0 1 0 0 0 0 0 3-1
0.0 3-2
004 3-3
10 91.2 5 37.9 4 29. 3 13.5 3 12.5 5 30. 4-1
0 4-2
5 30. 3 18. 3 18. 4 22. 4 22. 4-2
005 5-1
22 22 22 22 22 22 5-2
8 10 12 16 20 21 5-2
7 9 11 15 19 21 5-2
2 4 6 14 18 21 5-2
1 3 5 13 17 21 5-2
012 12-1
0 12-2
1 2 1 1 1.0 12-3
3 4 1 2 1.0 12-3
5 6 1 3 1.0 12-3
7 8 2 4 1.0 12-3
9 10 2 5 1.0 12-3
11 12 2 6 1.0 12-3
13 16 2 7 1.0 12-3
17 20 2 8 1.0 12-3
21 21 2 9 1.0 12-3
2. 22 2 10 1.0 12-3
0 12-3
013 13-1
24 13-2

```

(CONT)

24
 U235A U236A U238A PU239APU240APU241APU242A016A HA23A CR-A HU55A PE-A 13-3
 HI-A IE135APH147APU148APH149ASH149ASPPA SSPPA TA101ACB-BA PE-BA HI-BA 13-4
 21
 U235B U236B U238B PU239BPU240BPU241BPU242B016B HA23B CR-B HU55B PE-B 13-4
 HI-B IE135BPH147BPU148BPH149BSH149BSPPB SSPPB TA101B 13-4
 020 20-1
 1 2 20-2
 U235A 1.2000-5U238A 5.8-3PU239A 8.3000-4PU240A 3.5000-4 20-3
 PU241A 6.6000-5PU242A 2.7000-5016A 1.4000-2HA23A 9.2-3 20-3
 PE-A 1.2800-2CR-A 3.7000-3HI-A 2.5000-3HU55A 3.5 -4 20-3
 3 4 20-3
 U235A 1.2000-5U238A 5.8-3PU239A 8.7000-4PU240A 3.6000-4 20-2
 PU241A 6.9000-5PU242A 2.8000-5016A 1.4000-2HA23A 9.2-3 20-3
 PE-A 1.2900-2CR-A 3.7000-3HI-A 2.5000-3HU55A 3.5 -4 20-3
 5 6 20-3
 U235A 1.2000-5U238A 5.8-3PU239A 9.1000-4PU240A 3.7000-4 20-2
 PU241A 7.3000-5PU242A 2.9000-5016A 1.4000-2HA23A 9.2-3 20-3
 PE-A 1.2800-2CR-A 3.7000-3HI-A 2.5000-3HU55A 3.5 -4 20-3
 7 12 20-3
 U235B 1.7000-5U238B 7.5000-3HA23B 8.5000-2PE-B 1.4000-2 20-2
 CR-B 4.0000-3HI-B 3.1000-3HU55B 3.8 -4016B 1.5000-2 20-3
 13 20 20-3
 U235B 2.8000-5U238B 1.2000-2HA23B 5.8000-2PE-B 1.0000-2 20-2
 CR-B 2.9000-3HI-B 2.0000-3HU55B 2.8000-4016B 2.4000-2 20-3
 21 22 2-3
 HA23B 2.5000-2PE-B 4.9000-2CR-B 1.4000-2HI-B 9.7000-3 20-2
 HU55B 1.4000-3 20-3
 0 20-3
 END 20-2
 DUTLIN
 EXPINS 0 6 0 30
 150. 0.25 10.0 0.0 0.0
 0 0 0 0 1 2 0 0 0 0 0 0 0 1 0 0 0 0 0 2 0 0
 0 0 0 0 0 1
 RTIMS 0 0 0 24
 0 0 0 0 0 1 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0
 END
 INPUT PROCESSOR
 OV EXPOSE
 1D 0 14 0 1 0 7 5 5 0 0 0 10 3 33 0 0 0 0 0 0
 2D /
 * DEPLETION TEST CASE - SIMILAR TO A CITATION TEST CASE *
 *U235 * U236 * U238 * PU239 * PU240 * PU241 * PU242 * IE135 *
 *PH147 * SM149 * NSPP * SSPP * PH148H * PH148 *
 3D 0.0 /
 1 2 3 4 5 6 7 /

DRIVERZ

(CONT)

```

0 9 10 11 12 /
6 8 9 13 14
ND 1.689-9 2.093-5 0.289-9 1.976-7 1.688-6 / DECAY CONSTANTS
SD 0.0715 0.06 0.06 0.0715 0.06 0.0715 0.06 / XE135 YIELD
0.027 0.03 0.03 0.027 0.03 0.027 0.03 / PU147 YIELD
0.013 0.02 0.02 0.013 0.02 0.013 0.02 / SM149 YIELD
1.5011 1.5466 1.5466 1.5011 1.5466 1.5011 1.5466 / SSFP YIELD
0.0134 7R / SSFP YIELD
FD / MATRIX EXPONENTIAL OR AVERAGE GENERATION RATE
3 4 2 4 5 2 5 6 2 6 7 2 1 2 2 /
9 13 900002 13 10 2 9 14 100002 14 10 2 12 11 2 /
7D / SUPPLEMENTAL EXPLICIT CHAINS
0 0 0 /
PD / BASIC EXPLICIT CHAINS
3 2 4 2 5 2 6 2 7 0 /
1 2 2 0 /
9 900002 13 2 10 0 /
9 100002 14 2 -10 0 /
0 0 /
12 2 11 0 0 /
1V ZHATDN
1D 55 / CONTROL ROD INSERTION INITIAL CORE
2D 05 0.0167-3 115 1.4-4 1.4-4 5.0-5 2.6-5
05 0.0167-3 115 1.4-4 1.4-4 5.0-5 2.6-5
05 0.0167-3 115 1.4-4 1.4-4 5.0-5 2.6-5
05 0.0167-3 115 1.4-4 1.4-4 5.0-5 2.6-5
05 0.0167-3 115 1.4-4 1.4-4 5.0-5 2.6-5
05 0.0167-3 115 1.4-4 1.4-4 5.0-5 2.6-5 3845
STOP
END
INPUT PROCESSOR
2V ZHATDN
1D 55 / CONTROL ROD HALF INSERTION INITIAL CORE
2D 05 0.5246-3 115 0.7-4 0.9-4 2.5-5 1.3-5
05 0.5246-3 115 0.7-4 0.9-4 2.5-5 1.3-5
05 0.5246-3 115 0.9-4 0.9-4 2.5-5 1.3-5
05 0.5246-3 115 0.9-4 0.9-4 2.5-5 1.3-5
05 0.5246-3 115 0.7-4 0.9-4 2.5-5 1.3-5
05 0.5246-3 115 0.7-4 0.9-4 2.5-5 1.3-5 3845
STOP
END
INPUT PROCESSOR
3V ZHATDN
1D 55 / CONTROL ROD REMOVED
2D 05 9.0585-3 115 0. 0. 0. 0. 05 9.0585-3 115 0. 0. 0. 0.
05 9.0585-3 115 0. 0. 0. 0. 05 9.0585-3 115 0. 0. 0. 0.
05 9.0585-3 115 0. 0. 0. 0. 05 9.0585-3 115 0. 0. 0. 0. 3845
STOP
END
INPUT PROCESSOR
3V ZHATDN
1D 55 / CORE FUEL ASSEMBLIES REPLACED, SECOND CYCLE CONTROL RODS
2D 1.200-5 0. 5.8-3 8.7000-4 3.5000-4 6.9000-5 2.8000-5 1.400-2
0.0167-3 3.70-3 3.50-4 1.28-2 2.50-3 0. 7R 1.7-4 1.8-4 5.0-5 2.6-5

```

(CONT)

[illegible]

2003 /
STOP

150

INPUT PROCESSOR

4V ZWATDU

10 55 / CONTROL BOSS HALP INSERTION SECOND CYCLE

20	05	0.5246-3	115	.05-0	0.9-0	2.5-5	1.3-5	
	05	0.5246-3	115	.05-0	0.9-0	2.5-5	1.3-5	
	05	0.5246-3	115	1.2-0	0.9-0	2.5-5	1.3-5	
	05	0.5246-3	115	1.2-0	0.9-0	2.5-5	1.3-5	
	05	0.5246-3	115	1.0-0	0.9-0	2.5-5	1.3-5	
	05	0.5246-3	115	1.0-0	0.9-0	2.5-5	1.3-5	3045
STOP								

2000

INPUT PROCESSOR

SV ZNATDU

1D 55 / CONTROL ROD REMOVED

2D 05 9.0505-3 115 0. 0. 0. 0. 05 9.0505-3 115 0. 0. 0. 0.
05 9.0505-3 115 0. 0. 0. 0. 05 9.0505-3 115 0. 0. 0. 0.
05 9.0505-3 115 0. 0. 0. 0. 05 9.0505-3 115 0. 0. 0. 0. 3045
STOP

RED

DUTLIN

[illegible]

END

11

TABLE 12.10. (CONTINUED)

PRIMITIVE CONVERSION RATIO, ALSO FOR CRITICAL SYSTEM, FROM COMPRESSION (ATONR/UNIT-SEC) - 1.20730 1.20613 3.20200 10
 TIME AFTER SUPPOSED 434.000 DAYS, ESTIMATED E 0.999177, PISSELS INVESTMENT 3.063303E 03 NO, CONVERSION DATE(00) 1.22726

ACCESS INPUT MODULE 1 - NORMALISE I/O= 3, CPU RES= 1.00
 READ INPUT DATA ONLY (FROM-MEMORY) - NORMALISE I/O= 0, CPU RES= 1.00

ACCESS MODULES 7 13 0 0 0 - NORMALISE I/O= 0, CPU RES= 1.00
 ITERATIONS, CONVERSIONS, SEARCH, FROM POWER DENSITY, E - 10 0.551720-05 0.0 0.000000 02 0.000000
 PRIMITIVE CONVERSION RATIO, ALSO FOR CRITICAL SYSTEM, FROM COMPRESSION (ATONR/UNIT-SEC) - 1.20715 1.20620 3.21015 10
 TIME AFTER SUPPOSED 000.000 DAYS, ESTIMATED E 0.999030, PISSELS INVESTMENT 3.077007E 03 NO, CONVERSION DATE(00) 1.22930

ACCESS INPUT MODULE 1 - NORMALISE I/O= 3, CPU RES= 0.00
 READ DATA FOR A SPECIAL PROCESSOR - NORMALISE I/O= 3, CPU RES= 0.00
 ACCESS PROCESSOR 'DOWLIN' - NORMALISE I/O= 3, CPU RES= 0.00

ACCESS MODULES 7 9 0 0 0 - NORMALISE I/O= 3, CPU RES= 0.00
 ITERATIONS, CONVERSIONS, SEARCH, FROM POWER DENSITY, E - 10 0.544300-04 0.0 0.023070 03 0.000700
 PRIMITIVE CONVERSION RATIO, ALSO FOR CRITICAL SYSTEM, FROM COMPRESSION (ATONR/UNIT-SEC) - 1.20570 1.20520 3.23010 10
 ADJUST - ITERATIONS, CONVERSIONS, E - 15 -7.150000-04 0.000700
 SEARCH SYSTEMS LISTING, E/(ITERATION (CM, DENSITY, PZ, PZ)) - 0.035300-07 0.044700-14
 PISSELS INVESTMENT 3.077007E 03 NO, CONVERSION DATE(00) 1.20570, PISSELS CONVERSION DATE 1.205070-10 NO, W-SEC

FIGAL I/O FILE MANAGEMENT TABLE
 FILE NAME SUPPLIED BY

FILE NUMBER	NAME	USER-ENTRY	USER-STACK	VERSION	UNIT
10	CONTROL			1	1
11		000015		1	1
12	000007			1	1
13	000007			1	1
14	000007			1	1
15	000007			1	1
16	000007			1	1
17	000007			1	1
18	000007			1	1
19	000007			1	1
20	000007			1	1
21	000007			1	1
22	000007			1	1

READ-UP NOT COMPLETE. -- DRIVER TO RUN NEW CASE - NORMALISE I/O= 1, CPU RES= 0.07 - TIME=23.03.27

REFERENCES

D. R. Vondy, T. B. Fowler, G. W. Cunningham, "Venture: A Code Block for Solving Multigroup Neutronics Problems Applying the Finite-Difference Diffusion-Theory Approximation to Neutron Transport, Version II," ORNL-5062, Issued: October 1975.

G. E. Bosler, R. D. O'Dell, W. M. Resnik, "LASIP-III, A Generalized Processor for Standard Interface Files," LA-6280-MS, Issued: April 1976.

W. A. Rhoades, "The FBSAM Data Transmission Package for IBM 360/370 Computers," ORNL-TM-5199, Issued: January 1976.

D. R. Vondy, "Programming Practices and Computer Code Development," ORNL-TM-5065, Issued: December 1975.

D. R. Vondy, T. B. Fowler, G. W. Cunningham, Input Data Requirements for Special Processors in the Computation System Containing the Venture Neutronics Code," ORNL-5229, Issued: November 1976.

END OF SECTION

APPENDIX A. LOCAL COMPUTER FACILITIES

The hardware at the ORNL (X-10) and ORGDP (K-25) sites are described here. Details about the local operation and use of the computers are covered in the local report "Programmer's Notebook," issued by the Computer Sciences Division of the Nuclear Division of Union Carbide Corporation.

Approximate Local Computer Charge

A fixed rate schedule is now in effect, subject to change, as shown in the following list. These rates exclude capital cost recovery charged users when appropriate.

	<u>Basic Rate</u>
IBM 360/195 CPU Time (per hr.)	\$250.00
IBM 370/155 CPU Time (per hr.)	50.00
IBM 360/75 CPU Time (per hr.)	30.00
IBM 360/91 CPU Time (per hr.)	150.00
IBM 360/91, 360/195, 360/75, 370/155	
Input/Output (per unit)	.0004
IBM 360/91, 360/195, 360/75, 370/155	
(per line printed, per card punched)	.0008
IBM 1401 CPU Time (per hr.)	25.00
IBM 360/30 CPU Time (per hr.)	30.00
PDP-10 Misc. Service (per hr.)	40.00
PDP-10 Kilo Core (per hr.)	15.00
PDP-10 Connect Time (per hr.)	6.00
PDP-10 Disk Space (per block per month)	.075
Teleprocessing CPU (per hr.)	110.00
Teleprocessing Connect Time (per hr.)	6.00
TSO CPU Time (per hr.)	30.00
TSO Disk Space (per track per month)	.75
TSO Connect Time (per hr.)	6.00

BLANK PAGE

Hardware

Two Computing Center sites are described here. The X-10 site is located in 4500W and the K-25 site is in K-1007. Five major computer systems, the IBM 360/75, IBM 360/91, IBM 370/155, IBM 360/195, and the DEC PDP-10, are operated along with plotting and other peripheral devices. This section contains a brief description of the hardware for each system. Some of the terms used are as follows:

b² - one binary digit
 by - 8 binary digits - the smallest addressable memory unit in IBM computers
 K - 1,024
 ns - nanosecond = 1/1,000,000,000 second
 μs - microsecond = 1/1,000,000 second
 msec - millisecond = 1/1,000 second
 bpi - bits per inch
 cps - characters per second
 CPU - Central Processing Unit

IBM 360/91 and IBM 360/75 (X-10 Site)

The 360/91 Computer System is made up of the following components:

<u>Quantity</u>	<u>Type</u>	<u>Description</u>
1	2091	<u>Central Processing Unit</u> - basic cycle time of 60 ns.
1	2395	<u>Processor Storage</u> -capacity of 2,048K bytes of which approximately 1500K is available for user programs.
2	2301	<u>Drum Direct Access Storage</u> - used for the operating system; not available to users. One drum can store 4,096,000 bytes.
2	2318	<u>Disk Direct Access Storage</u> - auxiliary storage device with a total capacity of 233,408,000 bytes spread over eight removable disk packs. One pack can store 29,176,000 bytes.
4	2401	<u>Magnetic Tape Units</u> - two are 9-track units recording data at 800 bpi. Two are 7-track units recording data at 200 bpi, 556 bpi, or 800 bpi.
2	2402	<u>Magnetic Tape Units</u> - 9-track (800 bpi).
1	2540	<u>Card Read/Punch</u> - 1,000 cards/minute reading; 300 cards/minute punching.
3	1403	<u>Printers</u> - 132 characters/line; up to 1,100 lines/minute.
1	2501	<u>Card Reader</u> .
-	-	Various control units for the above auxiliary storage and input/output devices.

The 360/75 Computer System is made up of the following components:

<u>Quantity</u>	<u>Type</u>	<u>Description</u>
1	2075	<u>Central Processing Unit</u> - basic cycle time of 200 ns.
3	2365	<u>Processor Storage</u> - total capacity of 768K bytes with 750 ns access time.
2	2361	<u>Large Core Storage</u> - total capacity of 2,048K bytes with average of $4\mu s$ access time.
2	2318	<u>Disk Direct Access Storage</u> - auxiliary storage of up to 233,408,000 bytes spread over eight removable disk packs. One pack can store 29,176,000 bytes.
	2311	<u>Disk Units</u> - auxiliary storage of 7,250,000 bytes on each unit.

4	2401 2402 2403	<u>Magnetic Tape Units</u> - two are 9-track units recording data at 800 bpi, and two are 7-track units recording data at 200 bpi, 556 bpi, or 800 bpi.
2	1403	<u>Printers</u> - 132 characters/line, up to 1,100 lines/minute. One printer has the universal character set.
1	2540	<u>Card/Read Punch</u>
1	23122	<u>CDC Disk Unit</u> - 8 disk packs with auxiliary storage of 50,352,000 bytes on each unit.
4	2321	<u>Data Cell Drives</u> - auxiliary storage of up to 400,000,000 bytes spread over ten rewritable data cells. One data cell can store 40,000,000 bytes.
1	2914	<u>Switching Unit</u> .
1		<u>Holt-Spiess-Hewman Data Set</u>
1		<u>Control Unit (Culler-Fried System)</u> .
1	CC70	<u>CCI Communication Interface</u> .
-	-	Various control units for auxiliary storage devices and input/output devices.

The following units may be directly shared by the 360/91 and 360/75:

Quantity	Type	Description
3	2314	Direct Access Storage Facility
1	2321	Data Cell Drive
4	2311	Disk Units
1	23122	CDC Disk Unit

The interconnection of the equipment is illustrated in Fig. A-1.

Fig. A-1. The X-10 Site Computer Facilities

PDP-10

The PDP-10 computer system is made up of the following major components:

Quantity	Type	Description
1	KA10	Central Processor - basic cycle time is 200 ns.
6	HA10	16K bank core memory, 1/2s (36 bit word)
1	HE10	16K bank core memory
1	HD10A	Ampex memory (64K)
3	TU55	DEC tape drives
1	CR10E	Card reader, 1000 cards/minute
1	CP10A	Card punch, 200-365 cards/minute
1	LP10C	Line printer, 1000 lines/minute
2	TU30A	IBM - 9-track compatible tape
1	BD10	Swapping Disk; 2,200,000 bytes (1 byte = 8 bits)
1	RH10B	Swapping Drive
17	RF02	Disk Pack Drive 23,385,600 bytes per drive
1	RF03	Disk Pack Drive 66,771,200 bytes per drive
1	DC10	(5) 8-line Communication (three with dial-up feature)
1	PDP-15/10	Computer for communications control (16K 18-bit words, basic cycle time of 800ns)
4	PDP-15/10	Computers for graphical CRT interaction
4	VT15	CRT Display Unit
1	R5000	Paper tape reader

The interconnection of the equipment is illustrated in Fig. A-2.

Fig. A-2. The X-10 Site PDP-10 Facility.

IBM 360/195 and IBM 370/155 (K-25 Site)Central Processing Units

- 360/195 - Main Processor (2,097,152 bytes)
 370/155 - Support and Main Processor (1,048,576 bytes)

Channels

- 360/195 - Three Selector Channels
 One Block Multiplexor Channel
 One Multiplexor Channel
 370/155 - Four Block Multiplexor Channels
 Two Byte Multiplexor Channels

I/O Devices

- Shared - Four IBM 3330 Disk Drives (8 Spindles)
 Seven CDC 33302 Disk Drives
 Two IBM 2314 Disk Drives (8 Spindles each)
 Eight 3420 Tape Drives (Dual 800-1600 bpi)
 360/195 - One 2408 9 track Tape Unit (1 drive)
 Two 2402 9 track Tape Units (4 drives)
 One 2402 7 track Tape Unit (2 drives)
 Two 2301 Drums
 370/155 - One 2401 9 track Tape Drive
 Two 2401 7 track Tape Drives
 One 2403 7 track Tape Drive
 One 3211 Printer
 Four 1403-M1 Printers
 One 3505 Card Reader
 One 2501 Card Reader
 Two 2540 Card Reader/Punch
 One 3525 Card Punch
 One 2303 Drum

Remote Job Entry

- 370/155 - IBM 1130 Computer (Paducah)
 IBM 3780 Terminal (Paducah)
 IBM DATA 100 Terminal (ERDA-ORO)
 IBM 360/50 - 370/165 (ERDA-HDQ)
 PDP-11 (Goodyear Atomic Corp.)
 IBM 360/50 (Y-12)
 IBM 360/30 (A⁴Research)
 360/75, 360/91, PDP-Complex (ORNL Computing Center)
 IBM 2780 Terminal (National Lead, Cincinnati)

Teleprocessing Terminals

- 360/195 - Fifteen 2740 type Terminals
 Three 1050 Terminals
 Five 2260 type Terminals

Operator Consoles

- 360/195 - One 3060 Console (CRT)
 370/155 - Four 3277 Consoles (CRT)
 One 3286 Printer

One 3215 Console
One 1443 Printer
One 2740 Console

(A 2540 card reader/punch and one 1403 printer which are normally on-line to the IBM 360/30 can be switched to the IBM 370/155 and are included in the configuration diagrams.)

The interconnection of the equipment is illustrated in Fig. A-3. Doubling of the 360/195 memory and addition of more disc units is expected in the near future. At present, about 60 percent of the 3330 disc unit space is dedicated to system use, leaving 40×10^6 bytes or 10^7 short words available on each unit (special units can be mounted when more space is needed).

Fig. A-3. The K-25 Site Computer Facilities.

Disk and Tape Drive Availability

There are disks available at each site for scratch storage space (SYSDA). There are also entire disk packs (YPRISC and ZPRISC) available at X-10 only to be mounted and used as a 'private' disk pack for the duration of a job. This 'private' pack will be released at the end of a job.

At K-25, it is recommended that jobs mount no more than two private disk packs concurrently. At X-10, jobs may mount three disk packs.

Jobs should use as few tapes as possible. The following limitations are on the number of drives actually used concurrently by a job, not necessarily the number of tapes used. At K-25, there are two 7-track, five 9-track (800 bpi) and five 9-track (1600 bpi) drives available. A job's turnaround time will be adversely affected by using half or more of an available mountable resource. At X-10, there are two 7-track and three 9-track drives available for a job.

System Software

The operating system used on the IBM 360/91 and 360/75 is HASP-MVT. The operating system used on the intercoupled IBM 370/155 - 360/195 is ASP-MVT. These systems are described below.

MVT

The MVT (Multiprogramming with a Variable Number of Tasks) Operating System was designed to efficiently make use of the total resources of the system. Resources include I/O devices and channels, the CPU, core storage, and existing data sets. A task is defined as any unit of work contending for system resources. A task may be (and generally is) a job, but a job may consist of more than one task.

MVT has the ability to have a maximum of fifteen tasks resident in core at any given time. A task has control of the CPU on a priority basis, relinquishing control 1) whenever it is necessary to wait for an I/O operation to be performed or 2) whenever a higher priority task requires the CPU. By a proper mixture of tasks, it is possible to get several tasks (jobs) through the system concurrently in less time than the total time that would be taken if they were run separately.

ASP

ASP (Attached Support Processor) is a program designed to permit configurations with one or more CPUs to function with simultaneous computation, job scheduling, and full I/O activities. An ASP system usually consists of two or more processors interconnected by high speed data paths called channel-to-channel adapters. The processor on which the ASP software runs is usually referred to as the SUPPORT machine, and all other interconnected processors are referred to as MAINs, even though the SUPPORT machine is often powerful enough to serve as a MAIN if necessary. At K-25, the IBM 370/155 is usually the SUPPORT machine for the interconnected system.

ASP runs as a problem program on the SUPPORT processor and acts as a machine operator for both MAINs. A maximum of eight user tasks may be resident in core at one time, and the maximum core available to any one job is approximately 1250K. The major functions of ASP are described below.

Input data streams are read with JCL and data being separated and stored in different direct access queues. JCL statements for DD * and DD DATA are rewritten to point to pseudo tape drives.

Data volume setup is performed before selection of a job for execution. A device of the proper type is selected, and instructions are issued to the operator requesting the mounting of the specified data volume on the selected device. When mounting has been completed, ASP verifies the mounting with MAIN.

MAIN scheduling is invoked by a status change on MAIN or by a new job entering the system. ASP determines how much main storage is available on MAIN and then selects a job that will fit. Job selection is based on region size, priority, and class.

Print and Punch functions are invoked after a job has been terminated on MAIN. Output is routed either locally or remotely with appropriate messages indicating any special requirements.

Purge is invoked whenever a job has completed all the above processes. An accounting card is punched, and all ASP space used by the job is freed.

HASP

The HASP (Houston Automatic Spooling Priority) system is a computer program which functions as a front end processor for the IBM Operating system (OS). HASP provides an efficient means of organizing jobs, scheduling their execution on the basis of job priority and job class, collecting each job's printed and punched output, and returning that output to the user. The process of collecting the card images which constitute the job and of saving the output of the job for later

printing and punching is called spooling. NASP operates, via telephone lines, several types of remote terminals in addition to local unit record devices (card readers, printers, and punches).

The present version of NASP is V3.1 and includes the National Institutes of Health (NIH) 'SHARED SPOOL' modification which allows the sharing of the work queue between multiple computers. NASP will schedule a maximum of 12 jobs for execution at any one time. The maximum core available for any user's job is approximately 1500K.

Compilers

Of interest here from all those available are the IBM FORTRAN IV H-Level compilers. At the present time, the one at X-10 is superior to the one at K-25.

Services

Services are available at the computer sites. The following are some of the important local contact points (phone numbers):

<u>Site</u>	<u>X-10</u>	<u>K-25</u>
Management	3-0326	3-3671
Operations	3-0205	3-3625
Programming Assistance	3-0177	3-3708
Auxiliary Services (EAM)	3-1914	3-3308
Dispatcher	3-0205	3-3625
Librarian	3-1703	3-3502
Data Storage (Tapes)	3-1214	3-3625
Status	3-1817	

Use of the Facilities

Here we discuss certain aspects of the use of these facilities regarding the execution of a major (large) computer code or system of them.

Loaded Programs Accessed

Loaded machine language programs are stored on a disc. A program or a resident code system driver is called into fast core with control cards when a job is started. The job remains resident in memory once started and shares services of the central processor and memory and auxiliary hardware with other resident jobs. The operator has no way of on-line control over program instructions (other than abortion). Machine CP time used on any job is available for programmed instructions to allow some user control, as to discontinu prior to automatic abort for wrap-up or to generate a recovery data file. Programs in wide use may not generally be changed (with FORTRAN instructions) by the user.

On Precision

Although some calculations are done with 4-byte words which give nearly 7 decimal digits significance, most are done in "double precision," as frequently required to obtain significant results, using 8 bytes which gives just over 17 decimal digits significance. The significance indicated here is that carried in calculations. (There is also round-off from decimal to binary representation which may cause simple sums to indicate greater loss in significance than actually occurs in calculations.) Many calculations can be done quite adequately with less than 7 decimal digits significance, but often subtractions which cause loss of several digits significance produce poor results. One example of difficulty is that which occurs when two iterate sets of flux values are to be extrapolated, and this extrapolation is based on the change in values. Loss of significance in differences becomes serious even at relatively small loss of significance in the individual values. It is especially bad because round-off is done at the hexadecimal number level on the IBM machines rather than the binary number level.

END OF APPENDIX