AU8708947

AAEC/E645

AAEC/E645

AUSTRALIAN ATOMIC ENERGY COMMISSION

RESEARCH ESTABLISHMENT

LUCAS HEIGHTS RESEARCH LABORATORIES

A GUIDE TO THE AUS MODULAR NEUTRONICS CODE SYSTEM

by

G. S. ROBINSON

APRIL 1987

ISBN 0 642 59863 0

AUSTRALIAN ATOMIC ENERGY COMMISSION
RESEARCH ESTABLISHMENT
LUCAS HEIGHTS RESEARCH LABORATORIES

A GUIDE TO THE AUS MODULAR NEUTRONICS CODE SYSTEM

by

G.S. ROBINSON

ABSTRACT

A general description is given of the AUS modular neutronics code system, which may be used for calculations of a very wide range of fission reactors, fusion blankets and other neutron applications. The present system has cross-section libraries derived from ENDF/B-IV and includes modules which provide for lattice calculations, one-dimensional transport calculations, and one- two-, and three-dimensional diffusion calculations, burnup calculations and the flexible editing of results.

Details of all system aspects of AUS are provided but the major individual modules are only outlined. Sufficient information is given to enable other modules to be added to the system.

National Library of Australia card number and ISBN 0 642 59863 0

The following descriptors have been selected from the INIS Thesaurus to describe the subject content of this report for information retrieval purposes. For further details please refer to IAEA-INIS-12 (INIS: Manual for Indexing) and IAEA-INIS-13 (INIS: Thesaurus) published in Vienna by the International Atomic Energy Agency.

A CODES; BURNUP; COMPUTER CALCULATIONS; CROSS SECTIONS; DATA COMPILATION; DATA PROCESSING; DIFFUSION; FISSION; FORTRAN; NEUTRON FLUX; NEUTRONS; NUCLEAR DATA COLLECTIONS

CONTENTS

1. INTRODUCTION

The AUS system of neutronics computer codes was developed initially for fission reactor core calculations, and has been applied to a wide range of fast and thermal reactor types. The original system [Robinson 1975] has undergone considerable development and, in particular, has been extended to cover fusion blanket calculations [Robinson 1984]. The revised system may be used in calculations on many other neutron applications which may also involve photons.

AUS is a modular system in which the computer codes (modules) may be executed in a very flexible manner. The modules communicate through well-defined data sets on disc. The AUS modules are quite large, which leads to some duplication of function but also provides an easier system for users. Although user input has not been entirely standardised, most problem specifications have to be entered once only. The present system has cross-section libraries derived from ENDF/B-IV and includes modules which provide for lattice calculations, one-dimensional transport calculations, one-, two- and three-dimensional diffusion calculations, burnup calculations and the flexible editing of results.

The AUS system was written originally for an IBM360/50 computer and now operates on an IBM4381 model 3 computer. The code is not dependent on a particular version of the operating system, but although modules are written almost entirely in FORTRAN, extensive use of the Assembler language in the supervisor program makes it machine-dependent. This report summarises the available modules and provides details of all system aspects of AUS. This includes the supervisor program, that part of user input which controls the calculation sequence, and the interface data sets. Sufficient detail is given to enable other modules to be added to the system. Details of some of the minor modules of the system are given in **appendix G**. This report effectively replaces that of Robinson [1975].

2. BASIC CONCEPTS OF THE SYSTEM

A *data pool* is a set of data with a defined structure which is used to pass information between modules. The structure and content of the data set must also be well documented. The following data pools are currently defined:

- XSLIB for cross sections,
- GEOM for geometry description,
- FLUXA and FLUXB for fluxes, and
- STATUS for isotopic compositions and spatial smearing factors.

A *module* is any computer program which is self-contained apart from a user-supplied input stream and input/output of data *via* data pools. A module must be available in the form of an IBM operating system (OS) load module as a member of a partitioned data set. Any program could be considered an AUS module, but it is the interaction of the program with other modules of the scheme which gives meaning to the term. A module does need to indicate errors by setting a condition code. Except for a few hundred bytes, the full core region of the AUS calculation is available to each module.

A *path* (or *step*) controls the sequence in which modules are linked to perform the required calculation. It also controls the data sets to be used by the modules. A path is written in the FOREX [Robinson 1968] dialect of FORTRAN and is executed under the supervision of the AUSYS program. As it is written in FORTRAN, the path itself can perform subsidiary calculations. The sequence in which the modules are linked, and the data sets on which they operate, can thus be programmed to depend on any calculated result made available to the path.

The OS job control language (JCL) for the AUS scheme is contained in the AUS *catalogued procedure.* The procedure has DD (data definition) statements for modules, data pools, utility data sets, and data sets specific to individual modules. Except for modules, the DD names are mostly of the three- or four-character form DDn. An AUS user may specify that particular named data sets are to be used for any of the DD statements describing data pools, by using JCL symbolic parameters. Making the association of the DD names DDn (and hence the required data sets) with FORTRAN unit numbers, for each module that is linked, is a function of the path. The supervisor program AUSYS assists in making this association by retrieving the standard set of unit numbers and DD names for each module from the data set AUS.LINKLIB.

The *Input stream* for AUS consists of input to AUSYS and to each module to be linked by the path. The input stream is broken up into blocks by records of the form *DDn, beginning in column one. The set of records following each *DDn record is loaded onto the data set with the DD name DDn. Each of these data sets DDn consists of the user input data to a particular module. The only restriction on this user input to a module is that it cannot contain records with *DD in columns 1 to 3. System data must always be loaded on DD1 and consist of

- control information for AUSYS;
- a FOREX source of a non-standard path or a request to retrieve a standard path from AUS.PATHLIB; and optionally
- any data required as direct input to the path program.

The *AUSYS program* supervises and supports the execution of a path. Initially it calls the FOREX routine to compile the path source statements and then transfers control to the compiled path. Requests from the path to link a module are interpreted and the module is loaded by making use of the AELINK routines [Mason & Richardson 1969]. Before loading the module, AUSYS writes a 'snapshot' of the path on a disc data set. When the module terminates, AUSYS is reloaded by AELINK, the snapshot is retrieved and execution of the path can continue from where it left off. AUSYS contains a number of utility routines which may be called directly by the path. AUSYS is also used to maintain the system data sets AUS.PATHLIB and AUS.LINKLIB.

The only *system subroutines* available to the programmer of a module are a set of input/output routines for a cross-section data pool. The routines are obtained from the data set AUS.FORTLIB when the module is link edited. Although user input to the various modules has not been standardised, all major modules have the same style of input. These modules use the free input routine SCAN [Bennett & Pollard 1967] to read keyword-directed data and have default values for most parameters.

The sample input stream given below illustrates some of the features of the scheme. The six records following *DD1 form the path program which, in turn, links the three modules MIRANDA, ANAUSN and EDITAR with input following *DD2, *DD3 and *DD4, respectively. The three modules together form a lattice calculation. The six records following *DD1 could be replaced by the single record STEP MAE, as they constitute a standard path:

```
// EXEC AUS
//GO.SYSIN DD *
*DD1
STEP *
       LINK MIRANDA
       LINK ANAUSN(1,3)
       LINK EDITAR(1,4)
       END
STOP
*DD2
HEAD SR5
DEFN D2OM D2O .9975 H2O .0025
DEFN U U238 .9928 U235 .0072
DEFN FUEL U .047829
DEFN CAN AL .0432264
DEFN MOD D2OM .033277
REQD FUEL CAN MOD
RM 0 5*.253 .12 5*.5805686  0
REG 1(1)5 FUEL 6 CAN 7(1)11 MOD
RESREG  0 1.265 .12 2.902843 0 SMEAR 5*1 2  5*3
BUCK  3.00-4
GROUPS 26 -1 .5(0.5)2.5(1.5)10. 12.5 14.3 15.3 16.2(0.4)20.2 21. 23.
START
STOP
*DD3
*DD4
 BUCK  3.00-4 SEARCH OFF START STOP
```

3. CURRENT MODULES

3.1 Introduction

This section describes briefly the available modules, comments on their standard usage and outlines some features of the user input which are common to all modules. Further information on the accessing of data pools by each module is given in **section 6**.

3.2 MIRANDA - Cross-section Generation

The MIRANDA module [Robinson 1986c] includes a multiregion (for slabs, cylinders or clusters) resonance calculation of the subgroup type, and a cell average B_n flux solution for preliminary group condensation. The cross-section library is an AUS data pool with temperature dependence and subgroup parameters fitted to tabulated resonance integrals as a function of potential scattering. The two available libraries both have data which have been obtained mostly from ENDF/B-IV. The original library, AUS.ENDFB of 128 neutron groups, is intended for use in fission reactor core calculations. The more recent library, AUS.ENDF200G of 200 neutron groups and 37 photon groups, is intended for use in fusion blankets and other neutron applications.

In lattice calculations, data from MIRANDA for each region of a cell are passed to one of the transport theory modules to continue the calculation. A large number of groups should be retained for large cells as the MIRANDA flux solution is homogeneous.

3.3 ANAUSN - 1D, S_N Transport

The ANAUSN module [Clancy 1982] is a general purpose, one-dimensional discrete ordinate module with anisotropic scattering. Fixed source, reactivity and criticality search calculations can be performed with this module, which is the standard module for cylindrical cells and all 1D global transport calculations.

3.4 WDSN - 1D, S_N Transport

The WDSN module [Robinson 1972] is based on the WDSNST program [Brissenden & Green 1968] which includes axial leakage of the form e^{iBz}. This early module has been retained only for its axial leakage treatment.

3.5 ICPP - Isotropic Collision Probability

The ICPP module [Robinson 1985] includes both approximate and accurate numerical methods for calculating collision probabilities in slab, cylindrical, spherical and cluster geometries. The module was developed particularly for many-group, few-region calculations of group condensation spectra. For many-region cell calculations, ICPP is preferred for slabs and clusters, and ANAUSN for other geometries.

3.6 POW and POW3D - Multidimension Diffusion Including Kinetics

The POW module [Pollard 1974] is a general purpose, two-dimensional diffusion code which includes feedback-free kinetics. The module is being replaced by POW3D which can also perform three-dimensional calculations. The modules include general criticality search options and extensive editing facilities, including perturbation calculations.

3.7 AUSIDD - 1D Diffusion

The AUSIDD module (**appendix G**) is a simple one-dimensional diffusion module intended for global calculations of condensation spectra, which is complementary to the multidimensional POW modules. It differs from the POW modules in using a finite difference scheme with mesh points at the centre of the mesh intervals.

3.8 EDITAR - 1D and 2D Editing

The EDITAR module [Robinson 1986a] provides reaction rate editing and cross-section condensation facilities following a transport calculation. The module includes a B_n calculation of cell leakage, provision for bilinear weighting and perturbation calculations, and access to the STATUS data pool for nuclide concentrations and spatial smearing factors.

3.9 CHAR - Lattice and Global Burnup

The CHAR module [Robinson 1986b] uses an analytic method to solve the nuclide burnup equations and the cross-section library provides the chain mechanisms. Lattice cell or global calculations may be undertaken in as many regions as desired. The normal function of the module is simply to update the STATUS data pool of nuclide compositions following a burnup step.

3.10 BURNMAC - Global Burnup

The BURNMAC module [Robinson 1986b] provides a simple alternative method for global burnup calculations and assumes that macroscopic cross sections are a function of irradiation only. Cross sections as a function of irradiation are obtained from previous lattice burnup calculations using the CHAR module.

3.11 AUSED - Cross-section Editing

The AUSED module [Harrington 1976] loads or edits an AUS cross-section data pool. It is used mainly to maintain the basic libraries but may also be used for temporary cross-section modification on the user's cross-section data pools. The conversion from tabulated resonance integrals to subgroup parameters is also carried out by this module.

3.12 PLOTFL2 and XSPLOT - Plotting

The PLOTFL2 module [Clancy 1978] plots fluxes and reaction rates based on a one-dimensional flux data pool. The user may input a simple formula to be used in forming the reaction rate. The XSPLOT module [Harrington 1978] plots cross sections on any AUS XSLIB data pool.

3.13 ORNL - Form Card Image Cross Sections

The ORNL module (**appendix G**) produces card image cross sections in ANISN format from an AUS cross-section data pool. This provides a rather weak link between the AUS system and widely available transport codes such as DOT [Rhoades & Mynatt 1973].

3.14 Modules For Data Pool Manipulation

Three modules, MERGEL, JOINER and EXPAND (**appendix G**), are available for the manipulation of data pools. The most useful of these is MERGEL which combines two AUS cross section data pools. The other two modules are more specialised, their use being confined to some burnup applications.

3.15 Input Features Common To Most Modules

Though user-supplied input data to the modules of the AUS scheme have not been standardised, most modules have a similar style of input. The modules use the input routine SCAN [Bennett & Pollard 1967] to read free format data which is grouped into entries. Each entry consists of a keyword to indicate the data type followed by a string of data. The same entries are used in a number of modules. Default values are available for many input parameters so that data requirements are kept to a minimum for standard calculations.

Some of the standard features used with the SCAN routine are as follows :

- Data are entered in columns 1 to 72 only.
- Keywords may be up to eight alphanumeric characters.
- Floating point data may be given in abbreviated form; *e.g.* 1,1.,1.E+0, 1.-0,1.E 0 are equivalent.
- Repeat notation is, for example, 4*0. = 0. 0. 0. 0.
- Increment notation is, for example, 1(2)7 = 1 3 5 7.
- Special characters may be used in most contexts at the discretion of the user to improve readability.
- A record with an * in column 1 is a comment record.

4. DETAILS OF THE SUPERVISOR PROGRAM AUSYS

4.1 Introduction

The AUSYS program is an AELINK control program [Mason & Richardson 1969], therefore to understand its function requires some knowledge of AELINK. AELINK consists of a number of Assembler language programs and subroutines which provide a dynamic program management facility by the use of the Assembler language macros LINK and XCTL. AELINK provides facilities for a control program to execute modules, modify DD names, pass OS standard parameter fields to modules, and retrieve condition codes. The sequence of events in an AELINK run is as follows:

(a) AELINK processes the input stream by loading the data following each *DDn record onto the data set with DD name DDn;

(b) AELINK loads the control program;

(c) the control program selects the next module and returns to AELINK;

(d) AELINK loads the selected module; and

(e) the module carries out its task and returns to AELINK.

The sequence is repeated from step (b). Only a 256-byte segment of AELINK always remains in memory and, in particular, the control program is overwritten by the loaded module.

THE AUSYS program transfers the task of module selection to a path which is a program written in the FOREX dialect of FORTRAN. The path is retrieved from AUS.PATHLIB or entered in the AUSYS input. The path is compiled and loaded by the FOREX routines and effectively becomes the AELINK control program, with AUSYS providing a suitable environment for its execution. When a path requests that a module be loaded, AUSYS writes the current state of the path program and associated variables onto disc before passing the load request to AELINK. After the module is executed, AUSYS retrieves the path which resumes execution as if the module was merely a subroutine. To the path, the only difference between calling a subroutine and linking a module is that all open data sets are closed when the module is linked.

4.2 Input Layout

Input to AUSYS is always included following a *DD1 record in the AUS input stream. The input is in the form of directives in free format and FOREX input in standard FORTRAN layout for non-standard paths. In the following description, data in upper case to be provided by the user must be reproduced exactly and entered from column 1. For normal AUS calculations there are three different forms:

A.

STEP named

where named is the name of a standard path to be obtained from AUS.PATHLIB.

B.

STEP *

Set of FOREX statements

STOP

where the set of FOREX statements is a complete non-standard path.

C.

SEEK named

STEP *

Set of FOREX statements

STOP

where named is the name of a set of standard subroutines to be retrieved from AUS.PATHLIB which when combined with the input records forms a path.

The STEP records may contain an additional integer m. If it is included, the first m-1 links to modules are ignored. This option assists in restarts of calculations provided that the required data pools have been saved.

Any of the three forms may be followed optionally by directives to initialise data sets and also by data which are direct input to the path program. The directive to initialise a data set is

$DDn DISP=NEW

where DDn is the DD name of a data set which is to be initialised with an end of file mark. Note, however, that this initialisation is automatically carried out by AUS when NEW data sets normally used as data pools are allocated (**section 5**). Further information on the editing features provided by AUSYS for STATUS data pools is given in **appendix E.**

Data required as direct input to the path program are given following the directive

$DD99

which must be the last $DD directive entered.

4.3 Coding an AUS Path

4.3.1 Summary of the FOREX language

FOREX language [Robinson 1968] is a dialect of FORTRAN, all features of which are available to the AUS user. Briefly, FOREX is single precision FORTRAN II with the following exceptions: the I/O statements are standard FORTRAN IV for sequential data sets only; and non-compounded logical IFs, which compare two arithmetic expressions, are permitted.

An important additional feature of the FOREX language is that MACRO statements may be included anywhere in the program. A MACRO statement consists of a keyword followed by a string of characters. The MACRO statement is compiled as a subroutine call with the character string as an argument. For example, the statement

```
LINK POW (1,3)
```

is a MACRO statement resulting in the execution of the POW module. The combination of such statements with normal FORTRAN statements provides flexibility and ease of path coding.

4.3.2 The LINK MACRO

The LINK MACRO sets up I/O unit assignments for a module and causes the module to be executed. AUSYS returns control to the next statement in the path when the module concludes. The path writer may regard this as a normal subroutine return except that any open data sets will have been closed. AUSYS tests the condition code returned by the module, and terminates the entire calculation if the code is not in the range 1 to 7 inclusive. This range was chosen to avoid having to find and modify all error exits of existing programs used as modules. These limits may be changed, however, by the path (**section 4.3.6**).

The form of the MACRO is

LINK named $(m_1, n_1),(m_2, n_2), \ldots$

where

named	is the DD name of a data set in the AUS catalogued procedure which contains the required module;
m_i	is a FORTRAN logical unit (or any DD name) used in the named module and may have the form of an integer ii which is interpreted as FTiiF001, or the form FTiiFjjj, or be any other DD name (*e.g.* AEPLOT) ;
n_i	is a DD name of the type DDn in the AUS catalogued procedure which is to be associated with the FORTRAN unit m_i. The form of n_i may be any of (a) a positive integer k representing the DD name DDk, (b) DDk explicitly, (c) an ALIAS for DDk (**section 4.3.4**), and (d) a negative integer -k which means integer ℓ in the kth word in COMMON (**section 4.3.6**) giving the DD name DDℓ.

The last indirect form of n_i allows dynamic specification of the data sets which are to be used by a module; for example,

```
   COMMON DUMMY (20), IUNIT
   DO 1 IUNIT = 6,10
 1 LINK POW (1, -21)
```

results in execution of the POW module using the data sets DD6 to DD10 as input in turn.

As all FORTRAN logical units used by a module must be associated with a DD name (and hence a data set), the direct specification of these in the LINK statement becomes tedious. To make path coding easier, the standard unit assignments for each module are stored on AUS.LINKLIB (**section 6**) and are recalled automatically. Only those assignments which are to be changed need to be specified in the LINK statement. There must be a one to one correspondence between FORTRAN unit numbers and DD names. Two FORTRAN units cannot be assigned to the same DD name. The use that is made of the DD names DDn is described in **section 5**.

Example

LINK MIRANDA (1,2), (FT02F001, DD12), (AEPLOT, DD13), (FT07F002, -27)

4.3.3 The ALTER MACRO

The ALTER MACRO is used to assign FORTRAN units to DD names for the path itself. ALTER has exactly the same form as the LINK statement except that *named* is not given. The standard assignments (of interest to the normal user) for a path if no ALTER is given are (FT01F001, DD1) and (FT03F001, DD13). These assignments are maintained by ALTER unless specifically overwritten. As the execution of a LINK MACRO does not affect the unit assignments for the path, an ALTER MACRO may be included at the beginning of a path and this unit assignment used throughout.

Example

ALTER (2, DD12)

4.3.4 ALIAS for DD names

To use a particular data set for any of the AUS data pools, a symbolic parameter is given the value of a data set name using JCL (**section 5**). Each of these three-character symbolic parameters is permanently associated in the catalogued procedure with a DD name. As it is much easier for a path writer to think in terms of these symbolic parameters than the DD names of form DDn, the symbolic parameter is made an alias for the DD name. The alias can then be used in all LINK and ALTER statements. The set of aliases built into AUSYS is

DD31	DD33	DD34	DD40	DD35	DD36	DD37	DD41	DD38	DD39
LIB	XS1	XS2	XS3	GM1	FL1	FL2	FL3	ST1	ST2

For coding convenience, a path may include or change aliases by using the ALIAS MACRO. The alias must be of three characters only.

Example

ALIAS (XS4, DD41), (XS1, DD27)

LINK POW (6, GM1), (8, LIB), (18, XS4), (10, XS1)

4.3.5 Multiple case facility

AUSYS provides a simple multiple case facility. To use this facility

- the MACRO statement MODIFY is included at the beginning of the path;
- a FORTRAN loop, which includes the LINK MACRO statements to be repeated is coded;
- the characters MOD (or simply M) are included anywhere in columns 73 to 80 on those input records to be replaced; and
- the replacement input records are given in the input block DD10.

Once the modify option is on, the system checks the input stream, FT01F001, of each module after it is linked and replaces any records marked as above by the next record in the DD10 input block. The option may be turned off by using the statement MODIFY OFF. To implement this option, the default unit assignment for AUSYS and the path program has been modified to include

(FT27F001,DDn),(FT28F001,DD10),(FT29F001,DD14),

where DD14 is a temporary data set and DDn is the input block of the last module linked. Words 11 and 12 of blank COMMON are used by this option.

4.3.6 Use of blank COMMON

A path and the AUSYS program share the same blank COMMON area in which the first 20 words are reserved for AUSYS use but are available to the path. A path may extend COMMON, up to a maximum of 20000 (4 byte) words, as in the example,

```
COMMON DUMMY (20), A(10000)
COMMON B (9980)
```

Use should be made of COMMON for large arrays as the sum of all non-COMMON variables and the compiled path is restricted to 6000 words.

Each routine in a path has seven variables pre-defined by a built-in COMMON of

COMMON STIME, ICOND, PROG(2), JERR, JFR, LCOND(2), ISKP

where STIME is the time available for the AUS step in minutes;

ICOND	is the condition code returned by the last module executed;
PROG(2)	is the name (2A4) of the last module executed;
JERR	is the error level of the FOREX compiler;
JFR	is used in association with the SCAN free input routines;
LCOND(2)	is the permissible range of condition codes (1 and 7 are standard); and
ISKP	is one more than the number of further LINK statements which are to be ignored.

4.3.7 Routines available to a path

The standard FORTRAN functions which are available are

EXP, ALOG, ALOG10, ATAN, SIN, COS, SQRT, TANH.

Standard subroutine sets which may be called are the I/O routines for an AUS cross-section data pool (**appendix B**) and the SCAN free input routines of Bennett & Pollard [1967].

Additional subroutine calls which may be made are

CALL CLOCK (T)
: which returns the time in minutes from the start of the AUS calculation.

CALL SEND

or

CALL EXIT
: which return control to OS. (The END statement in FOREX is executable and serves the same purpose.)

CALL PSTAT (IU, IOUT)
: which lists on unit IOUT the STATUS data pool that is on FORTRAN unit IU. Card image formats are used if IOUT = 2.

CALL PMATS (IU,RAD)
: which produces card images on FORTRAN unit 2 giving material definitions from the STATUS data pool on FORTRAN unit IU for the irradiation value RAD for those materials which have undergone burnup. RAD has the same units as the CHAR module, which are normally watt-days. Linear interpolation in irradiation is used.

CALL LIST (IU, MNXST, MNSP, MNSCAT, MNSCSP, MNP)
: which lists the AUS cross-section data pool that is on FORTRAN unit IU. The remaining

arguments are for the ARDT subroutine (**appendix B**) and are all equal to 2 for a complete listing.

CALL LISTEN (MSHORT, MGAP, MGA, IFNUC, IFN, IFGA)

which is called before a call to the LIST subroutine if a partial list of an AUS cross-section data pool is required. The additional arguments take the following values :

- MSHORT is zero if only the beginning of each record, *i.e.* one printed line, is to be printed, otherwise the full record is printed.
- MGAP is the number of terms ℓ in a $P_{\ell-1}$ expansion of photon production which are to be printed and zero gives no photon production.
- MGA is the number of terms in the expansion of photon interaction as for MGAP.
- IFNUC is a vector such that if IFNUC(I)=0, the I^{th} nuclide is not printed.
- IFN is a vector of length equal to the number of neutron groups which takes the following values for each group:

  0 no print,
  1 print everything,
  2 print cross sections only,
  3 print scattering matrices only, and
  4 print photon production only.
- IFGA is a vector of length equal to the number of photon groups which serves the same function as IFN.

4.4 Maintenance of the System Data Sets AUS.LINKLIB and AUS.PATHLIB

The system data sets AUS.LINKLIB and AUS.PATHLIB are sequential FORTRAN data sets which are maintained using the AUSYS program. AUS.LINKLIB contains the standard unit assignments used in linking each module. AUS.PATHLIB contains the standard paths of the scheme.

Input records which update, load or print the data sets are included in the system input immediately following the *DD1 card. The directives UPDATE, LOAD, STOP and PRINT control the available functions. PRINT causes the contents of both path and link libraries to be listed. The input required to update or add the named path is

UPDATE

STEP named

A set of records containing the path program

STOP

A set of standard subroutines (to be retrieved using the SEEK directive) is handled in the same way.

To update or add a LINK, the input required is

UPDATE

LINK named k $(m_1, n_1), \ldots(m_k, n_k)$

where the form and meaning are the same as for the LINK MACRO except that the number of unit specifications k must be given, an ALIAS may not be used, and FORTRAN layout (*i.e.* column 6 continuation) may not be used.

The LOAD directive is similar to UPDATE except that a new library is created. Any number of STEPs (each delimited by a STOP) and LINKs may be included between the LOAD directive and a final STOP record.

Functions to delete or reorganise the data sets have not been included in AUSYS. A file containing the data to be permanently included in the libraries is maintained and used to reload the libraries from time to time.

5. JOB CONTROL LANGUAGE REQUIREMENTS

5.1 Introduction

The OS job control language is used by AUS for all allocation and retrieval of data sets. The large number of DD statements for all possible data sets to be used by AUS are contained in the AUS catalogued procedure and are of only minor concern to the AUS user. The JCL statements entered by the user for a simple AUS run are

```
//     EXEC  AUS
//GO.SYSIN  DD *
Data for the calculation
/*
```

which cause the AUS procedure to be executed. In many cases, however, either some of the data pools created will be required for later calculations, or some data pools will already exist. This situation is handled by the specification of symbolic parameters on the EXEC statement.

5.2 Conventions for the Use of DD Names

The current version of the AUS catalogued procedure is listed in **appendix A**. The use made of the data set on the DD statement with the name DDn is mainly governed by a set of conventions.

The data sets on DD1 to DD10 are temporary card image data sets which are used for user input to modules. The data in the main input stream following an *DDn card is automatically loaded onto the DDn data set at the start of a calculation. The DD names DD11, DD12 and DD13 are reserved for plotter, card image and print output, respectively. The data sets on DD14 to DD16 are further temporary card image data sets which may be used as scratch data sets by a module or path. The data sets on DD17 to DD30 are a corresponding set of unformatted data sets. At present, DD14 and DD21 to DD25 are the only scratch data sets used by modules, except for POW3D which uses DD17 to DD30.

The data sets on DD31 and DD33 to DD41 are used as the data pools of the AUS scheme. Except for DD39, they may all be easily saved and retrieved by the use of symbolic parameters. DD31 is a standard AUS cross-section library, and DD33, DD34 and DD40 are used as cross-section data pools. DD35 is used as a geometry data pool, and DD36 and DD37 are used for the flux data pools FLUXA and FLUXB, respectively. DD38 and DD39 form a pair of STATUS data pools; DD39 is not normally saved as it is merely a pointer data set for DD38. DD41 is used as the flux dump for the POW3D module.

The data sets on DD53 to DD57 are special data sets which are each used by one module only. DD98 and DD99 are dummy data sets which are used if a set of output from a module is not required. The remaining data sets of the procedure are system data sets.

5.3 Use of Symbolic Parameters

All except one of the DD names which are normally used for AUS data pools have been given symbolic data set names and symbolic data set disposition fields. The default values for these parameters result in temporary data sets which are not saved by OS. To save or re-use any of the data pools, a data set name is assigned to one of the symbolic parameters XS1, XS2, XS3, GM1, FL1, FL2, FL3 or ST1, together with an appropriate disposition.

The correspondence of symbolic parameters and DD names is (XS1, DD33), (XS2, DD34), (XS3, DD40), (GM1, DD35), (FL1, DD36), (FL2, DD37), (FL3, DD41) and (ST1, DD38). The default disposition, DISP=NEW, causes a new data set to be allocated and an end of file to be written onto it. For example, to save XS1 in one job and to use it in the next,

```
// EXEC AUS,XS1='AUS.GSRXSANY'
```

could be specified in the first job and

```
// EXEC AUS,XS1='AUS.GSRXSANY',DISPXS1=OLD
```

in the second job.

Any data set name which has a node (*e.g.* AUS) present in the OS data set catalogue may be used. If the AUS node is used, the user's initials (*e.g.* GSR) should also be included, as in the example above.

The other symbolic parameter of general interest is LIB (DD31) to specify the main cross-section library. The default value for LIB is ENDFB, which specifies that the 128-group AUS.ENDFB library is to be used. The other library which may be used is AUS.ENDF200G which has 200 neutron groups and 37 photon groups.

5.4 Use of Other JCL Parameters

The default value of the region size for the AUS scheme is 768 kbytes which is adequate for most calculations. The modules of the scheme generally make use of whatever memory is available but may require more than 768 kbytes for some large calculations. To increase the region to 1000 kbytes, for example, use

```
// EXEC AUS, any symbolic parameters,
// REGION.GO=1000K
```

The normal listing of the complete input stream may be prevented by using PARM.GO= where nothing follows the = sign.

6. STANDARD LINKAGES

The standard association of FORTRAN logical unit numbers with DD names for each AUS module is stored on AUS.LINKLIB. Only the required changes then need be given in the LINK MACRO of a path. The standard assignments for each module are tabulated below:

ANAUSN					
FT01 DD2	FT02 DD12	FT03 DD13	FT08 DD21	FT09 XS1	FT10 GM1
FT11 FL2	FT12 DD22	FT22 XS2	FT23 XS3		
AUSED					
FT01 DD2	FT02 DD12	FT03 DD13	FT08 LIB	FT10 XS1	FT11 XS2
PLOT DD11	FT04 DD56	FT07 DD21	FT12 XS3		
AUSIDD					
FT01 DD2	FT03 DD13	FT07 GM1	FT09 FL2	FT10 XS1	
BURNMAC					
FT01 DD2	FT03 DD13	FT04 GM1	FT05 FL1	FT09 ST1	FT10 XS1
FT11 XS2					
CHAR					
FT01 DD2	FT03 DD13	FT04 GM1	FT05 FL1	FT06 FL2	FT07 XS1
FT08 XS2	FT09 ST1	FT10 ST2	FT11 DD21	FT12 DD22	FT13 DD23
FT14 XS3					
EDITAR					
FT01 DD2	FT03 DD13	FT04 ST1	FT05 ST2	FT07 GM1	FT09 FL2
FT10 XS1	FT11 XS2	FT12 XS3			
EXPAND					
FT01 DD2	FT03 DD13	FT04 ST1	FT05 ST2	FT10 XS1	FT11 XS2
FT06 DD21					
ICPP					
FT01 DD2	FT03 DD13	FT11 FL2	FT13 GM1	FT14 XS1	FT16 XS2
JOINER					
FT03 DD13	FT10 XS1	FT11 XS2	FT12 XS3		
MERGEL					
FT01 DD2	FT03 DD13	FT04 ST2	FT10 XS1	FT11 XS2	FT29 DD21
FT12 XS3					
MIRANDA					
FT01 DD2	FT03 DD13	FT04 ST1	FT05 ST2	FT06 DD21	FT07 GM1
FT08 LIB	FT09 FL2	FT10 XS1	FT11 XS2	FT12 XS3	FT13 DD22
FT14 DD23	FT15 DD24				
ORNL					
FT01 DD2	FT02 DD12	FT03 DD13	FT10 XS1		

```
PLOTFL2
 FT01 DD2     FT03 DD13    FT09 XS1     FT10 GM1     FT11 FL2     PLOT DD11
POW
 FT01 DD2     FT02 DD12    FT03 DD13    FT04 DD57    FT05 DD21    FT06 GM1
 FT07 DD14    FT08 LIB     FT09 FL1     FT10 XS1     FT11 XS2     PLOT DD11
 FT12 DD22    FT13 DD23    FT14 DD24    FT15 DD25    FT16 DD98    FT17 ST1
 FT18 XS3
POW3D
 FT01 DD2     FT02 DD12    FT03 DD13    PLOT DD11    FT04 DD53    FT06 GM1
 FT07 DD14    FT08 LIB     FT09 DD21    FT10 XS1     FT11 XS2     FT14 DD28
 FT18 XS3     FT19 DD29    FT20 DD30    FT21 DD23    FT22 DD24    FT23 DD25
 FT24 FL1     FT25 DD22    DR01 FL3     DR02 DD17    DR03 DD18    DR04 DD19
 DR05 DD20    DR06 DD26    DR07 DD27
WDSN
 FT01 DD2     FT03 DD13    FT09 DD21    FT11 FL2     FT13 GM1     FT14 XS1
 FT15 DD98
XSPLOT
 FT01 DD2     FT02 DD12    FT03 DD13    FT08 LIB     FT10 XS1     FT11 XS2
 FT12 XS3     PLOT DD11    FT04 DD53
```

7. STANDARD PATHS

Standard FOREX path programs may be retrieved from the system data set AUS.PATHLIB by entering the following directive in the AUSYS input, *i.e.* after the *DD1 record:

```
STEP named
```

where named is the name of a path.

Because the modules of AUS are so large, and paths are normally so simple to write, this feature has proved to be relatively unimportant. The only paths currently available which are of general interest are the simple module sequences detailed below. A user's own standard paths may be added to the library and this has been of value.

Single module paths are available for the modules MIRANDA, ANAUSN, ICPP, POW3D and POW. For example

```
STEP POW
```

is equivalent to

```
STEP *
       LINK POW
       END
STOP
```

There are four paths for cell calculations; MIE and MAE are for one-stage calculations, and MIEAE and MIEIE are for two-stage calculations. A two-stage calculation is many-group, few-region followed by few-group, many-region. The paths are

```
STEP MIE                              STEP MAE
  LINK MIRANDA (1,2)                    LINK MIRANDA (1,2)
  LINK ICPP (1,3)                       LINK ANAUSN (1,3)
  LINK EDITAR (1,4)                     LINK EDITAR (1,4)
  END                                   END
STOP                                  STOP
```

```
STEP MIEAE
  LINK MIRANDA (1,2)
  LINK ICPP (1,3)
  LINK EDITAR (1,4)
  LINK ANAUSN (1,5), (9, XS3)
  LINK EDITAR (1,6), (10, XS3), (12, XS1)
  END
STOP
```

```
STEP MIEIE
  LINK MIRANDA (1,2)
  LINK ICPP (1,3)
  LINK EDITAR (1,4)
  LINK ICPP (1,5), (14, XS3)
  LINK EDITAR (1,6), (10, XS3), (12, XS1)
  END
STOP
```

8. CODING AN AUS MODULE

8.1 Introduction

An AUS module is similar to a stand-alone FORTRAN program in most respects. The distinguishing feature is the use of AUS data pools which are described in **appendices B to E**. In coding a module, maximum use should be made of these data pools so that user-supplied input is kept to an absolute minimum. As new data sets to be used as data pools are initialised with end of file marks, all data pools can be checked to ascertain whether they contain pertinent information. The AUS scheme also makes use of the OS condition code to pass information between the module and the supervisor program. Other desirable features of an AUS module are a standard style of input data and efficient use of the available memory. Descriptions of some standard AAEC FORTRAN library routines which are useful in coding a module are included in the following subsections.

8.2 Use of the Condition Code

The condition code is set by a module to indicate to the supervisor whether the computation was successfully completed. The supervisor terminates the path if the condition code returned by the module is not in the range 1 to 7 inclusive. A zero condition code has been considered as an error in order to avoid the necessity of finding all error exits in existing stand-alone codes which are converted to AUS modules. To set the condition code and exit (to the supervisor), the calling sequence is

```
CALL CEXIT(N)
```

This is equivalent to the FORTRAN statement

```
STOP N
```

except that the STOP statement types on the computer console and should be avoided.

8.3 Use of Computer Memory

All computer memory available for an AUS calculation is available to a module apart from a few hundred bytes. Modules are coded to have flexible dimensions for stored arrays and to make automatic use of the available storage. Thus, in large problems the user need only increase the region size of the AUS job step.

Variable dimensions are achieved either by using the FORTRAN feature, which allows variable dimensions of subroutine arguments, or by explicitly calculating all addresses within one singly subscripted variable. The latter method allows complete flexibility of use of memory but considerably increases the time required to code a module.

The VARRAY routine of Cox and Pollard, [Pollard 1978, appendix] is used for the dynamic procurement of memory. The four calling sequences are as follows:

(a) CALL NARRAY(N)
which returns the number of four-byte words of free memory (and from which the program must deduct storage required by buffers).

(b) CALL VARRAY(AV, IARD, N)
which procures N four-byte words of memory and returns IARD as the address of the first word of memory made available relative to the *four-vector AV and which is aligned to an *eight-word boundary. AV may be any dimensioned variable of the subroutine which calls VARRAY.

(c) CALL VARRAY(AV, IARD, -N)
which releases the memory procured with (b).

(d) CALL VARRAY
which releases all memory made available and must be called before the module terminates.

8.4 Example of an AUS Module

The sample module listed in **appendix F** has been included to illustrate some of the general features of AUS modules. The module includes sample subroutines to read the XSLIB, GEOM and FLUXB data pools as well to demonstrate most of the module requirements discussed in this section. The coding has been simplified by skipping much of the data pool information and by not making the normal tests for consistency of information.

The listing gives the complete input to compile, load and run a temporary module within the AUS scheme. For simplicity, the necessary data pools are assumed to exist already. To add a module permanently to the AUS scheme, a DD statement for the module is added to the AUS catalogued procedure, and the standard unit assignments for the module a.e added to the system data set AUS.LINKLIB.

9. REFERENCES

Bennett, N.W., Pollard, J.P. [1967] - SCAN - a free input subroutine for the IBM360. AAEC/TM399.

Brissenden, R.J., Green, C. [1968] - The superposition of buckling modes on to cell calculations and application in the computer program WDSN. AEEW-M809.

Clancy, B.E. [1978] - AUS module PLOTFL2. AAEC unpublished report.

Clancy, B.E. [1982] - ANAUSN — a one-dimensional multigroup SN transport theory module for the AUS reactor neutronics system. AAEC/E539.

Engle, W.W. [1967] - A user's manual for ANISN - a one dimensional discrete ordinates transport code with anisotropic scattering. K-1693.

Harrington, B.V. [1976] - AUS module AUSED — an editing program for AUS cross-section data pools. AAEC/E389.

Harrington, B.V. [1978] - AUS module XSPLOT. AAEC unpublished report.

Mason, C., Richardson, D.J. [1969] - AELINK — a facility for the dynamic linkage of independent IBM360 computer progams. *Proc. Fourth Australia Computer Conf.*, Adelaide, August, p.363.

Pollard, J.P. [1974] - AUS module POW — a general purpose 0,1 and 2D multigroup neutron diffusion code including feedback-free kinetics. AAEC/E269.

Pollard, J.P. [1978] - SKAN — a free input labelled output variable dimensioning routine for the IBM360 computer. AAEC/E431.

Rhoades, W.A., Mynatt, F.R. [1973] - The DOT III two-dimensional discrete ordinates transport code. ORNL-TM-4280.

Robinson, G.S. [1968] - FOREX — a FORTRAN compilation subroutine for the IBM360. AAEC/E190.

Robinson, G.S. [1972] - AUS module WDSN. AAEC unpublished report.

Robinson, G.S. [1975] - AUS — the Australian modular scheme for reactor neutronics computations. AAEC/E369.

Robinson, G.S. [1984] - Extension of the AUS reactor neutronics system for application to fusion blanket neutronics. AAEC/E583.

Robinson, G.S. [1985] - ICPP — a collision probability module for the AUS neutronics code system. AAEC/E620.

Robinson, G.S. [1986a] - EDITAR — a module for reaction rate editing and cross-section averaging within the AUS neutronics code system. AAEC/E621.

Robinson, G.S. [1986b] - CHAR and BURNMAC — burnup modules of the AUS neutronics code system. AAEC/E624.

Robinson, G.S. [1986c] - MIRANDA — a module based on multiregion resonance theory for generating cross sections within the AUS neutronics code system. AAEC/E626.

APPENDIX A
AUS CATALOGUED PROCEDURE

```
//*        "AUS" QUERIES: G. ROBINSON
//AUS      PROC  LIB=ENDFB,
//               VIO=VIO,PQ=10,SQ=100,P=10,S=300,SYSOUT=C,
//               DISPXS1=NEW,DISPXS2=NEW,DISPGM1=NEW,DISPFL1=NEW,
//               DISPFL2=NEW,DISPST1=NEW,XS1='&&XS1',XS2='&&XS2',
//               GM1='&&GM1',FL1='&&FL1',FL2='&&FL2',ST1='&&ST1',
//               DISPXS3=NEW,XS3='&&XS3',DISPFL3=NEW,FL3='&&FL3'
//ALLOC    EXEC  PGM=NEWEOF,REGION=10K,
//            PARM=(&DISPXS1,&DISPXS2,&DISPGM1,&DISPFL1,&DISPFL2,&DISPST1)
//STEPLIB   DD   DSN=AUS.FORTLIB,DISP=SHR
//ALL1      DD   DSN=&XS1,DISP=(&DISPXS1,CATLG),SPACE=(1000,(&PQ,&SQ)),
//               UNIT=&VIO,DCB=(RECFM=VBS,BLKSIZE=4584,LRECL=X)
//ALL2      DD   DSN=&XS2,DISP=(&DISPXS2,CATLG),SPACE=(1000,(&PQ,&SQ)),
//               UNIT=&VIO,DCB=(RECFM=VBS,BLKSIZE=4584,LRECL=X)
//ALL3      DD   DSN=&GM1,DISP=(&DISPGM1,CATLG),SPACE=(1000,(&PQ,&SQ)),
//               UNIT=&VIO,DCB=(RECFM=VBS,BLKSIZE=4584,LRECL=X)
//ALL4      DD   DSN=&FL1,DISP=(&DISPFL1,CATLG),SPACE=(1000,(&PQ,&SQ)),
//               UNIT=&VIO,DCB=(RECFM=VBS,BLKSIZE=4584,LRECL=X)
//ALL5      DD   DSN=&FL2,DISP=(&DISPFL2,CATLG),SPACE=(1000,(&PQ,&SQ)),
//               UNIT=&VIO,DCB=(RECFM=VBS,BLKSIZE=4584,LRECL=X)
//ALL6      DD   DSN=&ST1,DISP=(&DISPST1,CATLG),SPACE=(1000,(&PQ,&SQ)),
//               UNIT=&VIO,DCB=(RECFM=VBS,BLKSIZE=4584,LRECL=X)
//ALL7      DD   DSN=&XS3,DISP=(&DISPXS3,CATLG),SPACE=(1000,(&PQ,&SQ)),
//               UNIT=&VIO,DCB=(RECFM=VBS,BLKSIZE=4584,LRECL=X)
//ALL8      DD   DSN=&FL3,DISP=(&DISPFL3,CATLG),SPACE=(1000,(&PQ,&SQ)),
//               UNIT=&VIO,DCB=(RECFM=F,BLKSIZE=4584)
//GO       EXEC  PGM=AELINK,PARM=L,REGION=768K
//STEPLIB   DD   DSN=AUS.FORTLIB,DISP=SHR
//          DD   DSN=SYS1.FORTL131,DISP=SHR
//AUSYS     DD   DSN=AUS.AUSYS(AUSYS),DISP=SHR
//POW       DD   DSN=AUS.POW(PROGPGM),DISP=SHR
//WDSN      DD   DSN=AUS.WDSNST(WDSN),DISP=SHR
//ICPP      DD   DSN=GSR.ICPP(PROGPGM),DISP=SHR
//MERGEL    DD   DSN=GSR.MERGEL(MERGEL),DISP=SHR
//AUSED     DD   DSN=AUS.AUSED(PROGPGM),DISP=SHR
//CHAR      DD   DSN=GSR.CHAR82(PROGPGM),DISP=SHR
//ANAUSN    DD   DSN=BEC.ANAUSN77.PROGRAM(PROGPGM),DISP=SHR
//MIRANDA   DD   DSN=GSR.MIRANDA(PROGPGM),DISP=SHR
//*FIVE      DD   DSN=AUS.FIVE(PROGPGM),DISP=SHR
//EDIT      DD   DSN=AUS.EDITN(PROGPGM),DISP=SHR
//EDITAR    DD   DSN=GSR.EDITAR(PROGPGM),DISP=SHR
//POW3D     DD   DSN=AUS.POW3D(PROGPGM),DISP=SHR
//PLOTFL2   DD   DSN=BEC.PLOTFL2.PROGRAM(PROGPGM),DISP=SHR
//BURNMAC   DD   DSN=GSR.BURNMAC(PROGPGM),DISP=SHR
```

Appendix A (continued)

```
//AUSIDD    DD  DSN=GSR.AUSIDD(AUSIDD),DISP=SHR
//ORNL      DD  DSN=GSR.ORNL(ORNL),DISP=SHR
//JOINER    DD  DSN=GSR.JOINER(PROGPGM),DISP=SHR
//EXPAND    DD  DSN=GSR.EXPAND(PROGPGM),DISP=SHR
//SYSPRINT DD   SYSOUT=A
//DD1       DD  UNIT=&VIO,SPACE=(1000,(10,100)),
//              DCB=(RECFM=FB,BLKSIZE=880,LRECL=80)
//DD2       DD  UNIT=&VIO,SPACE=(1000,(10,100)),
//              DCB=(RECFM=FB,BLKSIZE=880,LRECL=80)
//DD3       DD  UNIT=&VIO,SPACE=(1000,(10,100)),
//              DCB=(RECFM=FB,BLKSIZE=880,LRECL=80)
//DD4       DD  UNIT=&VIO,SPACE=(1000,(10,100)),
//              DCB=(RECFM=FB,BLKSIZE=880,LRECL=80)
//DD5       DD  UNIT=&VIO,SPACE=(1000,(10,100)),
//              DCB=(RECFM=FB,BLKSIZE=880,LRECL=80)
//DD6       DD  UNIT=&VIO,SPACE=(1000,(10,100)),
//              DCB=(RECFM=FB,BLKSIZE=880,LRECL=80)
//DD7       DD  UNIT=&VIO,SPACE=(1000,(10,100)),
//              DCB=(RECFM=FB,BLKSIZE=880,LRECL=80)
//DD8       DD  UNIT=&VIO,SPACE=(1000,(10,100)),
//              DCB=(RECFM=FB,BLKSIZE=880,LRECL=80)
//DD9       DD  UNIT=&VIO,SPACE=(1000,(10,100)),
//              DCB=(RECFM=FB,BLKSIZE=880,LRECL=80)
//DD10      DD  UNIT=&VIO,SPACE=(1000,(10,100)),
//              DCB=(RECFM=FB,BLKSIZE=880,LRECL=80)
//DD11      DD  SYSOUT=&SYSOUT
//DD12      DD  UNIT=&VIO,SPACE=(1000,(10,100)),
//              DCB=(RECFM=FB,BLKSIZE=880,LRECL=80)
//DD13      DD  SYSOUT=A,
//              DCB=(RECFM=FBA,BLKSIZE=1330,LRECL=133)
//DD14      DD  UNIT=&VIO,SPACE=(1000,(10,100)),
//              DCB=(RECFM=FB,BLKSIZE=880,LRECL=80)
//DD15      DD  UNIT=&VIO,SPACE=(1000,(10,100)),
//              DCB=(RECFM=FB,BLKSIZE=880,LRECL=80)
//DD16      DD  UNIT=&VIO,SPACE=(1000,(10,100)),
//              DCB=(RECFM=FB,BLKSIZE=880,LRECL=80)
//DD17    DD  UNIT=&VIO,SPACE=(1000,(&P,&S)),
//              DCB=(RECFM=VBS,BLKSIZE=4584,LRECL=X)
//DD18    DD  UNIT=&VIO,SPACE=(1000,(&P,&S)),
//            DCB=(RECFM=VBS,BLKSIZE=4584,LRECL=X)
//DD19    DD  UNIT=&VIO,SPACE=(1000,(&P,&S)),
//            DCB=(RECFM=VBS,BLKSIZE=4584,LRECL=X)
//DD20    DD  UNIT=&VIO,SPACE=(1000,(&P,&S)),
//            DCB=(RECFM=VBS,BLKSIZE=4584,LRECL=X)
```

Appendix A (continued)

```
//DD21     DD  UNIT=&VIO,SPACE=(1000,(&P,&S)),
//             DCB=(RECFM=VBS,BLKSIZE=4584,LRECL=X)
//DD22     DD  UNIT=&VIO,SPACE=(1000,(&P,&S)),
//             DCB=(RECFM=VBS,BLKSIZE=4584,LRECL=X)
//DD23     DD  UNIT=&VIO,SPACE=(1000,(&P,&S)),
//             DCB=(RECFM=VBS,BLKSIZE=4584,LRECL=X)
//DD24     DD  UNIT=&VIO,SPACE=(1000,(&P,&S)),
//             DCB=(RECFM=VBS,BLKSIZE=4584,LRECL=X)
//DD25     DD  UNIT=&VIO,SPACE=(1000,(&P,&S)),
//             DCB=(RECFM=VBS,BLKSIZE=4584,LRECL=X)
//DD26     DD  UNIT=&VIO,SPACE=(1000,(&P,&S)),
//             DCB=(RECFM=VBS,BLKSIZE=4584,LRECL=X)
//DD27     DD  UNIT=&VIO,SPACE=(1000,(&P,&S)),
//             DCB=(RECFM=VBS,BLKSIZE=4584,LRECL=X)
//DD28     DD  UNIT=&VIO,SPACE=(1000,(&P,&S)),
//             DCB=(RECFM=VBS,BLKSIZE=4584,LRECL=X)
//DD29     DD  UNIT=&VIO,SPACE=(1000,(&P,&S)),
//             DCB=(RECFM=VBS,BLKSIZE=4584,LRECL=X)
//DD30     DD  UNIT=&VIO,SPACE=(1000,(&P,&S)),
//             DCB=(RECFM=VBS,BLKSIZE=4584,LRECL=X)
//DD31        DD  DSN=AUS.&LIB,DISP=SHR
//DD33        DD  DSN=&XS1,DISP=OLD,SPACE=(1000,(&PQ,&SQ),RLSE)
//DD34        DD  DSN=&XS2,DISP=OLD,SPACE=(1000,(&PQ,&SQ),RLSE)
//DD35        DD  DSN=&GM1,DISP=OLD,SPACE=(1000,(&PQ,&SQ),RLSE)
//DD36        DD  DSN=&FL1,DISP=OLD,SPACE=(1000,(10,1000))
//DD37        DD  DSN=&FL2,DISP=OLD,SPACE=(1000,(&PQ,&SQ),RLSE)
//DD38        DD  DSN=&ST1,DISP=OLD,SPACE=(1000,(&PQ,&SQ))
//DD39        DD  UNIT=&VIO,SPACE=(1000,(1,10)),
//                DCB=(RECFM=VBS,BLKSIZE=4584,LRECL=X)
//DD40        DD  DSN=&XS3,DISP=OLD,SPACE=(1000,(&PQ,&SQ),RLSE)
//DD41        DD  DSN=&FL3,DISP=OLD,SPACE=(1000,(10,1000))
//DD53        DD  DSN=AUS.POW3DCOM,DISP=SHR
//DD54        DD  DSN=AUS.EDITCOMN,DISP=SHR
//*DD55        DD  DSN=AUS.FIVECOM,DISP=SHR
//DD56        DD  DSN=AUS.AUSEDCOM,DISP=SHR
//DD57        DD  DSN=AUS.POWCOM,DISP=SHR
//DD98        DD  DUMMY
//DD99        DD  DUMMY
//AELINKUT DD  UNIT=&VIO,SPACE=(1000,(20))
//FT30F001 DD  UNIT=&VIO,SPACE=(1000,(10,100)),
//             DCB=(RECFM=VBS,BLKSIZE=7294,LRECL=X,BUFNO=1)
//FT31F001 DD  DSN=AUS.PATHLIB,DISP=SHR
//FT32F001 DD  DSN=AUS.LINKLIB,DISP=SHR
//*            END OF PROCEDURE "AUS"
```

APPENDIX B
AUS CROSS-SECTION DATA POOLS (XSLIB)

B1. INTRODUCTION

An XSLIB data pool is used for the storage of neutron and photon group cross-section data. The same form is used for general microscopic cross-section libraries, on which the data may be potential scattering and temperature dependent, and for macroscopic data (cm^{-1}) or microscopic data (barns) which have been derived for some particular calculation. The data pool may contain group cross sections, subgroup parameters, group scattering matrices up to any P_n order, burnup information, fission spectra, and comments on the data source. It is organised into a number of pseudo files, the first of which contains general information, and each of the rest contains information on one material. This type of structure restricts the number of groups which may reasonably be used to less than about 300. A material in the library need not be unique (*e.g.* more than one set of data may be given for an isotope) but it must be given a unique identifier. All data are single precision.

The libraries AUS.ENDFB and AUS.ENDF200G which may be input to the MIRANDA module are XSLIB data pools of the subgroup type. These libraries are normally used as the LIB data set on DD31. Other XSLIB data pools which are generated and used during a particular calculation, or a small range of calculations, are used as the XS1, XS2 or XS3 data sets. These data pools do not include subgroup data and are essentially oriented to a particular reactor or cell. Normally, they do not include a distinct treatment for photons but lump neutrons and photons together as particles. The current library structure is a modification of that originally used for neutrons only.

B2. RECORD LAYOUT

An XLSLIB data pool is a sequential unformatted data set which, for NN materials, is made up of NN + 1 pseudo files. Each pseudo file consists of a number of 228-word FORTRAN records which are blocked and unblocked by a special set of input/output subroutines (section B4). A pseudo file mark, which is the word '####', is written as the first word of the first 228-word record of each pseudo file except the first file. That is, the '####' mark functions like an end of file mark but is written as the first word of each material file for ease of file skipping. This structure is designed for fast library access on a sequential data set. The detailed description of the data pool is given below in terms of pseudo records but it must be remembered that these are not FORTRAN records and, in particular, they cannot be skipped with a short read.

B3 DETAILED STRUCTURE

B3.1 First Pseudo File

Library identification

First heading record of 20 (4 byte) words:

words 1 and 2	=	'AUSDATAb' ;
words 3 and 4	=	'NXSCATbb' ;
words 5 and 6	=	NAME where NAME = library identifier ;
words 7 to 20	=	56 characters used to describe the data pool.

Second heading record of 10 words :

word 1	=	library update identification number ;
word 2	=	number of materials (NN) ;
word 3	=	number of neutron groups (NG) + 1000 × number of photon groups (NGGA) ;

words 4 to 9 are maximum values for any material in the library of

word 4	=	number ($\leq$ 20) of reactions (NREAC) ,

word 5	=	number of neutron cross-section temperatures (NXST) ,
word 6	=	number of neutron cross-section σ_p tabulations (NSP) ,
word 7	=	number of neutron scatter matrix temperatures (NSCATT) ,
word 8	=	number of neutron scatter matrix σ_p tabulations (NSCSP) ,
word 9	=	order of scatter matrix expansion (NP), see below,
word 10	=	100 * NK + INDRES, where NK is the number of kerma factors, and INDRES takes the following values

(a) = 1, neutron cross sections are functions of potential scattering,

(b) = 2, neutron cross sections are functions of σ_0 which depends on the total cross sections,

(c) = 3, tabulated neutron 'cross sections' are actually subgroup parameters for the resonance theory of MIRANDA, used for AUS.ENDFB and AUS.ENDF200G,

(d) = 4, cross sections and scatter matrices are tabulated for all values of LPS and KPS (see below) and this is used for irradiation-dependent data in burnup calculations.

Contents

NN records of 11 words each. The Nth record relates to material N in the library (the N + 1 pseudo file) and the words are

words 1 and 2	=	A8 material name, *e.g.* 'U238bbb' ;
words 3 and 4	=	A8 data source, e.g. 'ENDFB4bb' ;
word 5	=	A4 modification, e.g. 'MOD2' ;
word 6	=	ND, number such that the highest ND for different data files for the one material is recommended data ;
word 7	=	NXST, number of neutron cross-section temperatures ;
word 8	=	NSP, number of neutron cross-section σ_p tabulations ;
word 9	=	NSCATT, number of neutron scatter matrix temperatures ;
word 10	=	NSCSP, number of neutron scatter matrix σ_p tabulations ;
word 11	=	NP which gives the order of the scatter matrix expansions and may be five hexadecimal digits each having the meaning P_n order + 1 for (in order lowest to highest digit) neutron scattering, photon production, (n,γ) photon multiplicity, fission photon multiplicity and photon interaction. Note that photon production may be split into those from (n,γ), fission and the remainder. NP may take the value zero which means no scatter matrices.

The first five words must form a unique material identifier not only on general cross-section libraries but also on a generated data pool composed of data obtained from, say, several cell calculations.

Neutron Group Information

A record of NG + 1 words
of the lethargies of the upper (energy) boundary of the first group and the lower boundaries of all groups.

A record of NG + 1 words
of the energies in eV corresponding to the above lethargies.

A record of NG words
giving the group velocities (in 10^8 cm s^{-1}).

Comments

A comment record of the form NC, (ST(K), K=1, NC)
where the words when printed 1X20A4 form lines of comments pertaining to the general library preparation.

Photon Group Information

A record of NGGA+1 words giving energies of the photon group boundaries. Given only for NGGA non-zero.

B3.2 **Material Pseudo Files**

File Mark

A one-word record of '####'

Identification

An 11-word record :
This duplicates the contents record of file 1, *e.g.* U238, ENDFB4, MOD2, ND, NXST, NSP, NSCATT, NSCSP, NP.

Burnup and Mass Information

A 20-word record :

words 1 and 2	=	A8 name of nuclide produced by decay ;
words 3 and 4	=	A8 name of nuclide produced by reaction 7 (*see cross-section records below*) ;
words 5 and 6	=	A8 name of nuclide produced by reaction 8 ;
word 7	=	0 to 6 to indicate fuel type (=0 if not fissionable) ;
word 8	=	the decay constant λ (10^{-24} s^{-1}) ;
words 9 to 14	=	fractional yield from fission of fuel of type 1 to 6 ;
word 15	=	fission energy release (joule/fission) ;
word 16	=	atomic mass (C12 scale) of an isotope ;
word 17	=	may be positive, zero or negative -

(a) > 0., atomic mass of second isotope of the nuclide which is a molecule,

(b) = 0., implies one isotope only,

(c) < 0., —D is given, where D is the average spacing between resonances (in eV) ;

word 18	=	fraction of potential scattering due to the first isotope ;
word 19	=	fraction of potential scattering due to the second isotope ;
word 20	=	NB, the burnup order indicator .

NB is used to re-order isotopes before analytic solution of the depletion equations. With few exceptions [Robinson 1986b], NB is a 7 decimal digit fixed point word PCCAAAM, where

P	=	1 for a fission product, else 0 ,
CC	=	the charge number ,
AAA	=	the mass number ,
M	=	1 for the ground state when a metastable isomer is also given, else 0.

The nuclide names given in words 1 to 6 inclusive need not be in the library. If word 1 is set to the three characters NO0, the nuclide does not burn out by decay. If both word 3 and word 5 are NO0, the nuclide does not burn out by neutron absorption.

Fission Spectrum

A record of NG words
giving the fission spectrum normalised to unit sum.

Temperature and σ_p Tables

A record of NXST words
giving the temperatures (K) at which neutron cross sections are tabulated. The temperatures are monotonic decreasing.

A record of NSP words
giving either
(a) set of σ_p (or σ_o) in decreasing order for which neutron cross sections are tabulated, or
(b) for INDRES=3, the number 1.E+20 and (NSP — 1) subgroup values of σ_{tot} in increasing order.
The subgroup theory assumes that group resonance integrals, RI, can be obtained from

$$RI/\delta u = \sum_{i=1}^{NSP-1} \frac{w_i s}{s + \sigma^i_{tot}}$$

where s is related to σ_p.

A record of NSCATT words
giving the temperatures (K) at which neutron scatter matrices are tabulated. The temperatures are monotonic decreasing.

A record of NSCSP words
giving the set of σ_p (or σ_o) in decreasing order for which neutron scatter matrices are tabulated.

Neutron Cross-section and Scatter Matrix Data

The data in this block are given for each group in turn. For any group the data are given as follows:

Cross Sections

A set of records each of the form LPS, LV, (XS(K), K=1,LV) :

LPS = -1 implies the set consists of this one record only, *i.e.* the cross sections are not tabulated ;

LPS = -2 implies NXST*NSP records are given, with records for NXST temperatures being given in turn for each σ_p value ;

LV = length of the XS vector and not all reactions need be given. The value of LV for a material is constant for all cross-section records but may be a different constant for subgroup parameter records.

The set of vectors XS is a set of cross-section records if INDRES $\neq$ 3 but, for INDRES = 3, the first NXST records are cross-section records and the remainder are subgroup parameter records.

For *cross-section records*, the data are

XS(1)	=	σ_{tr}, the transport cross section ;
XS(2)	=	σ_{abs}, the absorption cross section ;
XS(3)	=	$\nu\sigma_f$, the fission emission cross section ;
XS(4)	=	σ_s, the scattering cross section, if LPS = -1 or σ_p, the potential scattering cross section, if LPS = –2 ;
XS(5)	=	σ_{tot}, the total cross section ;
XS(6)	=	σ_f, the fission cross section ;
XS(7)	=	σ_x, the first burnup reaction, usually the (n,γ) cross section ;
XS(8)	=	σ_y, the second burnup reaction or the (n,2n) cross section ;
XS(9)	=	total kerma factor excluding following entries ;

XS(10)	=	elastic scattering kerma factor ;
XS(11)	=	(n,γ) kerma factor ;
XS(12)	=	fission kerma factor.

The kerma factor is the cross section times the energy deposited in the units of barn eV. To allow for esonance shielding, elastic scattering, capture and fission may have separate entries and the total kerma actor is obtained by summing the values given. For any given nuclide, not all NK values need be given, *i.e.* LV nay be less than NK+8.

Additional entries may be added depending on the library type:
(a) For macroscopic materials, additional parameters may be

XS(9+NK)	=	radial diffusion coefficient,
XS(10+NK)	=	axial diffusion coefficient,

or for 3D calculations XS(9+NK), XS(10+NK) and XS(11+NK) may be x,y,z diffusion coefficients.

(b) For resonance groups on an INDRES=3 library

(i)	XS(9+NK)	=	average peak resonance height σ_o ;
	XS(10+NK)	=	average value of $2E_r/\Gamma$, where E_r is resonance energy and Γ is total width;
(ii)	XS(9+NK)	=	XS(10+NK) = 0 implies narrow resonance theory;
(iii)	XS(9+NK)	=	1., XS(10+NK) = 0 implies very wide resonance extending over many groups; or
(iv)	XS(9+NK)	=	$-\sigma$ inel, XS(10+NK) = 0 implies narrow resonance theory and inelastic scattering; and
(v)	additionally		XS(11+NK) may be the ratio of isotropic to anisotropic P_0 removals.

For *subgroup parameter* records the data are

XS(1)	=	subgroup weight for absorption, w_i,
XS(2)	=	subgroup weight for resonance scattering,
XS(3)	=	subgroup weight for fission,
XS(4)	=	subgroup weight for fission emission,
XS(5)	=	subgroup weight for the ratio of group flux to asymptotic group flux, but XS(3) and XS(4) may be omitted so that XS(3) becomes the subgroup weight for flux.

Scatter Matrices

For each P_n scattering order, the following data are given in turn unless NP=0 when no data are given. A set of records each of the form KPS, KV, (SCAT(K), K=1, KV) :

KPS = position of the self-scatter term in the SCAT vector but has the additional implications

(a) KPS > 1, the set consists of NSCATT records giving temperature-dependent thermal data;

(b) KPS = 1, the set consists of this one record;

(c) KPS = 0, the set consists of NSCATT*NSP records with NSCATT records being given in turn for each σ_p value. The position of self scatter is 1.

These additional implications apply to the first record of a set only and for the following records, if any, KPS is simply the self-scatter position.

KV = length of the SCAT vector, $1 \le KV \le NG$,

SCAT = a vector of outscatters from the current group, g .

NOTES

(1)

$$\sigma^{\ell}_{g\to g'} = \int_{-1}^{1} \sigma_{g\to g'}(\mu)\, P_{\ell}(\mu)\, d\mu \quad ,$$

where μ is the cosine of the scattering angle.

(2) The self-scatter term of the P_0 vector is a 'true' self-scatter term if NP > 1, but it is transport-corrected if NP = 1. In any module using a P_0 calculation, the library self-scatter term should be ignored. Neutron balance should be conserved by using σ_{tr} (reaction 1) and the relation

$$\sigma_{g\to g} = \sigma_{tr} - \sigma_{abs} - \sum_{g'\neq g} \sigma_{g\to g'} \quad .$$

In higher order P_n calculations, the P_0 self-scatter term on the library should be used and the total cross section obtained not from reaction 5 but from

$$\sigma_{tot} = \sigma_{abs} + \sum_{g'} \sigma_{g\to g'} \quad .$$

Any group condensation must be consistent with these two conventions.

(3) The (n,2n) reaction also requires special consideration. The standard convention is adopted with $2\sigma^{(n,2n)}_{g\to g'}$ being included in the scatter vector and $\sigma_{n,2n} = \sum_{g'} \sigma^{(n,2n)}_{g\to g'}$ being subtracted from cross section 2,

i.e. $\sigma_{abs} = \sigma_{cap} + \sigma_f - \sigma_{n,2n}$

where σ_{cap} includes all reactions which do not result in neutron emission.

For (n,3n) reactions, $3\sigma^{(n,3n)}_{g\to g'}$ is included in the scatter vector and $2\sigma_{n,3n}$ is subtracted from σ_{abs}.

This is the end of the neutron group data description

Material Comments

A comment record of the form NC, (ST(K), K=1, NC) when the words are printed 1X20A4 to form lines of comments pertaining to this material.

Photon Production Matrix Data

To allow for resonance shielding, photon production data for (n,γ) and fission reactions may be given separately as multiplicity matrices. That is, the photon production matrix is obtained by multiplying the given data by the appropriate group cross section. For each neutron group in turn, data are given for

Set 1 - the P_n expansion of photon production excluding that given in Set 2 or Set 3,

Set 2 - the P_n expansion of the (n,γ) multiplicity, and

Set 3 - the P_n expansion of the fission multiplicity.

The P_n expansion order for each set is extracted from NP. For each value of n in a set, the given record has the form

MPS, MV, (VECT(K), K=1, MV)

where
MPS is 1000 $\times$ set number + first photon group in which photons are produced,
MV is the number of groups in which photons are produced, and
VECT gives the photon production (or multiplicity).

Photon Interaction Data

No data are given in this block if the photon interaction scattering order, which is taken from NP, is zero. The data given for each photon group in turn are similar to those given for neutron cross sections which are independent of σ_p and T. The cross section record is

-1, LVA, (XS(K), K=1, LVA)

where

XS(1)	=	transport cross section ,
XS(2)	=	absorption cross section ,
XS(3)	=	zero ,
XS(4)	=	Compton scattering cross section ,
XS(5)	=	total cross section ,
XS(6)	=	zero ,
XS(7)	=	photo-electric cross section ,
XS(8)	=	pair production cross section, and
XS(9)	=	kerma factor

The records in the P_n scattering expansion are

LPS,LV, (SCAT (K),K=1,LV)

where LPS is the position of self-scatter, and
LV is the number of outscatters from the photon group.

This completes a material file

B4. INPUT/OUTPUT ROUTINES

A set of subroutines is provided for blocking and unblocking the records on an XSLIB data pool. These routines should be used for all access to the data pools except for special applications requiring the simultaneous reading of more than one data set. As the input and output routines are separate and use different buffers, one data set may be written while another is being read.

B4.1 Input Routine

There are four entries to the subroutine, namely ARDP, ARDN, ARDS, and ARDT. A description of the calling sequences follows.

(a) The library positioning entry is

CALL ARDP (IT, K)

where IT is the FORTRAN unit number.
K=0 implies rewind and must be given to open a data set, and
K>0 skips K pseudo file marks.

This entry must be used to skip a pseudo file mark, even if it is the next word.

(b) The N-word read entry

CALL ARDN (ST, N)

reads | N | words and returns the words in the vector ST if N is positive. A negative value of N is used for skipping unwanted data and a nil result is obtained from N=0.

This entry is used for file 1 data and material file data except cross-section (or sub-group parameter) and scatter matrix records. Note also that CALL ARDN (ST, I) and CALL ARDN (ST(I + 1), J) are equivalent to CALL ARDN (ST, I + J).

(c) The cross-section and scatter vector read entry

CALL ARDS (LPS, LV, ST)

reads the next pseudo record and returns LPS = first word,
LV = second word and (ST(K), K=1, LV).

The cross-section (or subgroup parameter) and scatter vector pseudo records must be read with this entry and not ARDN. There is a lapse of correspondence with the actual data pool pseudo records if NP=0. In this case, a P_0 scatter vector is returned with LPS=1, LV=1, ST(1)=$\sigma_{tr} - \sigma_{abs}$.

(d) The library contraction entry is

CALL ARDT (MNXST, MNSP, MNSCAT, MNSCSP, MNP)

This entry is used if the programmer wishes to treat an XSLIB data pool as if it were a subset of the actual data set. The input parameters in the argument list correspond with the five maximum values of the words 5 to 9 of the second data pool heading reaord. If the parameter is greater than one, the corresponding set of data is all returned by the ARDS entry. The additional meanings are

MNXST=MNSP=1	the cross-section record for the highest σ_p value and the lowest temperature is the only one returned,
MNSCAT=1	the scatter records for the lowest temperature only are returned,
MNSCSP=1	the scatter records for the highest σ_p value only are returned (with KPS=1 if the library has 0),
MNP=1	only the P_0 scatter vectors are returned,
MNXST=MNSP=0	no cross-section records are returned, and

MNSCAT=MNSCSP=MNP=0 no scatter records are returned.

Library contraction may not be used if any data following the neutron cross section and scattering data are required. A library containing separate neutron and photon data may be read as if it contained only neutron data provided that the number of groups NG is used modulo 1000, and the scattering order NP is used modulo 16.

B4.2 **Output Routine**

The three entries to the routine are given below.

(a) The library positioning entry is

CALL AWRP (IT, K)

where IT is the FORTRAN unit number,
K=0 implies rewind and must be given to open the data set,
K>0 writes a pseudo file mark, and
K<0 writes an actual file mark.

(b) The N-word write entry

CALL AWRN (ST, N)

writes N words of the vector ST.

(c) The cross-section and scatter vector write entry

CALL AWRS (LPS, LV, ST)

writes the pseudo record LPS, LV, (ST(K), K=1, LV).

Example

The sample module given in **appendix F** includes a subroutine RDXS which might be used to read a simple XSLIB data pool of neutron data without tabulated cross sections. In this sample read subroutine, only the essential data have been read and the rest have been skipped. Only the data σ_{tr}, σ_a, $\nu\sigma_f$ and $\sigma_{g\rightarrow g'}$ are returned from the subroutine. Note the use of CALL ARDT to ensure that more general data pools, particularly those with higher order P_n scattering data, can be read.

APPENDIX C
AUS GEOMETRY DATA POOLS (GEOM)

C1. INTRODUCTION

A GEOM data pool contains information on the geometry of some reactor system or component. The data include geometry type, mesh intervals and material layout. The material layout is in terms of materials which are specified by number only, with the numbers corresponding to position on some associated cross-section data pool.

The GEOM data pool is normally used as the GM1 data set on DD35. It is a sequential unformatted data set containing single precision data.

The sample module of **appendix F** includes a subroutine RDGEOM for reading a GEOM data pool describing a one-dimensional geometry.

C2. DETAILED STRUCTURE

Heading record of 21 (4 byte) words:

words 1 and 2	=	NAME (A8), the geometry identifier,
words 3 to 18	=	a 64-character heading,
word 19	=	the number of dimensions described (ND),
word 20	=	the geometry type (IGEOM),
where IGEOM	=	0 for rectangular geometry xyz,
	=	1 for cylindrical geometry rz,
	=	2 for spherical geometry r, and
word 21	=	NLS is 1 normally, but 5 for cluster geometry and the number of axial mesh intervals in 3D geometry. Then 2 + ND + NLS is the number of records describing a geometry.

Mesh interval record of form NXMI, (XM(I), I=1), NXMI) is given if ND>0

where XM are the mesh interval widths in cm, and
NXMI is the number of mesh intervals for the first dimension (that is, the x or r direction).

A further mesh interval record of the same form is given for each additional dimension. Thus there is a total of ND mesh interval records.

Boundary condition record of 2*ND words :

word 1	=	left boundary condition of first dimension,
word 2	=	right boundary condition of first dimension, and
words 3 to 6,		where required, are for the second and third dimensions.

A one word dummy record is given if ND=0

The boundary condition normally has the following meanings:

<0. implies periodic,

=0. implies reflective, and

>0. implies free and is used by diffusion codes as the extrapolation distance in transport mean free paths (usual value is 0.71).

For cylindrical geometry, additional interpretations are

word 1 = number of sides of a polygonal outer boundary, and a zero value implies circular; and

word 2	=	- (number of mesh intervals to be treated as the inner region in applying a polygonal reflective boundary condition limited to two regions). Zero and positive have the normal meanings.

Both numbers are given in floating point. In cluster geometry, these boundary conditions are applied to the pin-cell boundaries and the outer boundary is taken as circular, white reflective.

Material layout record of the form NL, (LAYOUT(I), I=1, NL),

where NL	is the product of the number of intervals in each dimension, excluding any third dimension,
LAYOUT	is the array of material numbers (LAYOUT(IX,IY) in the 2D case), where the number corresponds to the material order on an XSLIB data pool, and the record is repeated for each XY plane in 3D geometry.

This completes the geometry description except for a cluster geometry which requires the following additional records.

A record of the form NL1, NPITCH, NTYPE, (NROD(I), MROD(I), PROD(I), QROD(I), I=1, NPITCH)

where NL1	=	4*NPITCH + 2,
NPITCH	=	number of rings of rods,
NTYPE	=	0,
NROD(I)	=	number of rods equally spaced on the ring I, numbered from the centre outward,
MROD(I)	=	number of subdivisions of a rod on ring I,
PROD(I)	=	pitch radius of the ring I, and
QROD(I)	=	angular displacement in radians of one of the rods of the ring I from a reference diameter of the cluster.

A record of the form NL2, ((DR(I,J), I=1, MROD(J)), J=1, NPITCH)

where NL2	=	Σ MROD (J),
DR	=	mesh interval of the radial subdivisions of a rod, in cm.

A record of the form NL2, ((LAYOUT(I,J), I=1, MROD(J)), J=1, NPITCH)

where LAYOUT = material numbers corresponding to the radial subdivision of the rods.

A record of the form NL3, (VOL(I), I=1, NL3)

where NL3	=	NXMI + NL2, and
VOL	=	region volumes with the volumes of the main annuli being given first, followed by rod subdivision volumes in the same order as the previous records.

This completes the description of the geometry. Other sets of geometry data may follow but this feature is supported only by the POW module.

APPENDIX D
AUS FLUX DATA POOLS (FLUXA AND FLUXB)

D1. INTRODUCTION

Unfortunately, a general data pool to store neutron group fluxes has not been incorporated in the AUS scheme. Use is made of the POW module flux dump as a FLUXA data pool and of a WDSN flux dump as a FLUXB data pool. The FLUXA data pool is used for 1D or 2D edge mesh fluxes and the FLUXB data pool is used for 1D mesh average fluxes.

The FLUXA data pool is normally used as the FL1 data set on DD36 and the FLUXB datapool as the FL2 data set on DD37. Both are sequential unformatted data sets.

The sample module in **appendix F** includes a sample subroutine RDFLB for reading a FLUXB data pool.

D2. DETAILED STRUCTURE OF A FLUXA DATA POOL

First record of 33 words :

words 1 and 2	=	NAME (A8), the flux identifier ,
words 3 to 18	=	a 64-character heading ,
words 19 and 20	=	'REALbbbb' or 'ADJOINTb' ,
words 21 and 22	=	'EIGENVbb' or 'SOURCEbb' or 'KINETICS' ,
words 23 and 24	=	'PROBLEMb' or anything else ,
word 25	=	0. or, for kinetics calculation, the power or flux ,
word 26	=	0. or, for kinetics calculation, the time in seconds
word 27	=	number of outers or, for kinetics calculations, the time step number ,
word 28	=	MAXX, the dimensioned size of the X mesh ,
word 29	=	MAXY, the dimensioned size of the Y mesh ,
word 30	=	NGIGD, the number of energy groups (plus the number of delayed groups if a kinetics calculation) ,
word 31	=	NXM, number of X mesh points, *i.e.* number of mesh intervals plus 1 ,
word 32	=	NYM, the number of Y mesh points, and
word 33	=	NG, the number of energy groups.

Second record of the form AKEFF, AEIGEN, BKEFF, BEIGEN, MOP, DLAM, RACCFO, SOMEGA, (OMEGA(I), I=1, NGIGD), (((FLX(I,J,K), I=1, MAXX), J=1, MAXY), K=1, NGIGD) ,

where the floating point variables are REAL*4, apart from FLX which is REAL*8, and

AKEFF	=	the reactivity k corresponding to λ_1 ;
AIEGEN	=	λ_1, a criticality search eigenvalue or 1 ;
BKEFF	=	the reactivity k corresponding to λ_2 ;
BEIGEN	=	λ_2, a second criticality search eigenvalue or 1 ;
MOP	=	convergence stage and takes the values 3 means calculation just started, 2 means outer extrapolation accepted hesitantly, 1 means outer extrapolation accepted readily, 0 means converged ;
DLAM	=	dominance ratio for outer iteration (second biggest/biggest eigenvalue) ;
RACCFO	=	accuracy of last outer ;
SOMEGA	=	trial value for inner SLOR coefficients, usually 1 ;
OMEGA	=	vector of inner SLOR coefficients for each energy group ; and
FLX	=	double precision array of edge fluxes (plus volume weighted

precursor concentrations for kinetics calculations).

As already stated, this is a POW flux dump and, from the second record, only AKEFF, MOP (converged or not) and FLX have a meaning for other modules. Additional 2-record flux dumps may be included in the one data set.

D3. DETAILED STRUCTURE OF A FLUXG DATA POOL

The data pool consists of a number of pseudo files which are separated by the single word record '####'. The records of each file are as follows:

First record of six fixed point words :

word 1	=	an indicator which takes the values 1 normal case, fully converged, 2 normal case, not converged, 5 normal case, fully converged, adjoint may also be given, 6 adjoint case, fully converged, 7 normal case, not converged, adjoint may also be given, 8 adjoint case, not converged ;
word 2	=	the geometry type, taking the values -1 cylinder, 0 slab, 1 sphere ;
word 3	=	product of the number of groups and the number of mesh intervals ;
word 4	=	0 ;
word 5	=	(2n - 1) where n is the P_n order of scattering ; and
word 6	=	0.

Heading record of 18 words :

words 1 and 2	=	NAME (A8), the flux identifier ;
words 3 to 18	=	a 64-character heading.

A record of EIGEN, NG, NR, (FLX(I,J), I=1, NR), J=1, NG) ,

where EIGEN = $1/k_{eff}$ or total activity for source calculations (REAL*8) ;

NG	= number of energy groups ;
NR	= number of mesh intervals ; and
FLX	= the (REAL*8) mesh average scalar flux.

The ANAUSN module writes angular flux starters at the end of this record to assist restarts, but accepts input flux guesses on which the number of starters is inappropriate.

A set of m records of the form (FLX(I,J,L), I=1, NR), J=1, NG) giving the currents for P_n scattering calculations with n>0, where m=n except in cylindrical geometry for which m = (n*(n+4))/4. The data are REAL*8.

APPENDIX E
AUS STATUS DATA POOLS

E1. INTRODUCTION

STATUS data pools are FORTRAN sequential, unformatted data sets which are normally used as the pair of data sets ST1 and ST2 on DD38 and DD39 of the AUS system, respectively. The two STATUS data pools are used in combination, with ST2 acting as a pointer to the main ST1 data set. The data pools consist of a sequence of entries with each entry determining its own function. The form of the entries is fixed, however, so that a module may skip or copy entries without knowing the details of all entries. Each module of a calculation may add additional entries to the end of the data pool by using the sequence: search for end of file, backspace and write. That is, the data pools are 'add-on' data sets in which all entries are retained throughout a calculation sequence.

The data pool contains entries for isotopic compositions, spatial smearing factors and other miscellaneous data which together form a history of the functions that have been performed by the modules during a sequence of calculations. Its major purpose is to allow the automatic evaluation of nuclide reaction rates at any stage of a calculation. The data pool differs from other AUS data pools in several ways; it can embrace an entire calculation sequence, it can include general information, and it can take part in controlling the AUS calculation. Because of this generality, the use that each module makes of the data pool must be carefully defined and suitable labelling conventions must be established to differentiate between data generated within different subsections of the overall calculation. To be more specific, we wish to be able to calculate nuclide reaction rates in the components of a lattice cell which, together with other cells, forms a representation of a reactor core.

E2. DATA POOL CONTENTS IN DETAIL

E2.1 Entry Format

Each entry consists of two or more records, the first of which establishes the type of entry. This first 7-word record is

(A(I),I=1,5),N,M

where A is either a fully qualified material name of 20 characters or it has the form

A(1 - 2)	is $DATA ,
A(3 - 4)	is the name of module writing the entry,
A(5)	is the entry type, *e.g.* TIME, CELL, GRPS, and
| N|	is the number of groups of information in the following records.

If N is negative, each information group begins on a new record thus enabling large quantities of data to be included easily. M is the maximum size of an information group in *4 words.

The trailing records have the form

((B(I,K),I=1,M),K=1,N) — one record for N positive, or
J,(B(I),I=1,J) — repeated |N| times for N negative.

E2.2 Mixing Rule Entry

This is the basic entry in the data pool and it determines the entry format. Mixing rules describe either the mixing of nuclides to form a discrete material or the spatial smearing of materials. The name of any nuclide or material consists of 20 bytes as in the XSLIB data pool. The conventions for naming are given in **section E3.2.**

The entry is (for positive N)

(A(I),I=1,5),N,M+5
((B(I,K),I=1,5),(C(I,K),I=1,M),K=1,N)

where

A is the name of a material formed from N constituents,
B is the set of constituent names, and
C is (a) the concentrations in atoms per barn cm of each nuclide, for M=1, or
(b) the spatial smearing factors for each material f_i, d_{ig} for M > 1, *i.e.* M is usually one more than the number of groups before energy condensation, but is one more than the number of groups after condensation if the nuclide data are condensed. In both cases d_{ig} is given for the same number of groups as the nuclide cross-section data pool.

The factors f_i, d_{ig} are given by

$$f_i = V_i / V_I \quad \text{, and}$$

$$d_{ig} = V_I \phi_{ig} / (\sum_{g \varepsilon G} \sum_{i \varepsilon I} V_i \phi_{ig}) \quad \text{, or}$$

$$d_{iG} = V_I \sum_{g \varepsilon G} \phi_{ig} / \sum_{g \varepsilon G} \sum_{i \varepsilon I} V_i \phi_{ig}) \quad ,$$

where V_i and ϕ_{ig} are region volumes and fluxes which are used to group condense and smear into group G and region I.

E2.3 The TIME Entry

The entry is
```
$DATA    MODNAM TIME  1  3
TNOW,TLAST,PTGRAL
```
where $DATA is the A8 character data '$DATA' ,

MODNAM	is the A8 module name,
TIME	is the A4 character data 'TIME' and the other entry types below are similar ,
TNOW	is the current time in days,
TLAST	is the previous time in days, and
PTGRAL	is the integral of power of the total system with respect to time in watt days or watt days cm^{-3}.

E2.4 The IRAD Entry

The entry is
```
$DATA  MODNAM IRAD  N  6
((A(I,J),I=1,5), B(J),J=1,N)
```
where

A	is the set of names of discrete materials which are burnt up,
B	is the integral of power density with respect to time in watt days cm^{-3} for each material.

E2.5 The CELL Entry

The entry is
```
$DATA  MODNAM CELL  1  3
CNAM   NCELL
```
where

CNAM	is the A8 name of the cell,
NCELL	is a count of cell calculations for the current time step.

E2.6 The GRPS Entry

The entry is

```
$DATA  MODNAM GRPS  1  N
(IGB(I),I=1,N)
```

where

N	is the number of condensed groups plus one,
IGB	is a set of fixed point numbers giving the first group of condensed group 1, and the last group of each condensed group. The numbers are in terms of the previous group set.

E2.7 The GFAC Entry

The entry is

```
$DATA  MODNAM GFAC  1  NG
(A(I),I=1,NG)
```

where

NG	is the number of energy groups; and
A(I)	is the ratio of the k_{eff} flux for a cell calculation to the k_∞ flux, for each group I.

E3. NOTES ON USAGE

E3.1 General Comments

The ST1 data set is the main data pool to which all entries should be added. The ST2 data set serves mainly as a pointer to the data in ST1 and, as such, contains the TIME and CELL entries which give structure to the data pool.

One TIME entry is given for each time step (including time zero) and precedes all other entries for that time. The TIME entry is followed by a single IRAD entry for times greater than zero. A CELL entry is given for each lattice cell calculation and immediately precedes entries written for that cell by a cross-section generation module.

Each set of mixing rule entries immediately follows an appropriate GRPS entry to define the group condensation. The mixing rules must be entered such that the order of materials on the STATUS data pool is the same as that on the cross section data pool. A further requirement is that the order of constituents within a spatial smearing rule be the same as the order of the definitions of those constituents on the STATUS data pool.

The GFAC entry which enables burnup of a cell in a critical spectrum is exceptional because it is written on the ST2 data set.

The use of STATUS data pools requires that materials be divided into two classes called materials and nuclides. A nuclide is present on the main cross-section library of the cross-section generation module and has microscopic cross sections. It appears only on the right hand side of a mixing rule. A material is composed of nuclides and must be defined by a mixing rule. It may be macroscopic or microscopic and may even be identical in cross section to a nuclide. Materials and nuclides may be in the same cross-section data pool, but more generally they are in different data pools which may have a different number of groups.

E3.2 Material Names

The materials and nuclides in an AUS calculation all have 20-byte names which are used to provide a unique identification. These names are constructed using the set of conventions detailed below.

The first requirement is that the name of a material or nuclide must be exactly the same on the STATUS and cross section data pools. The 20 bytes consist of two 8-byte words and one 4-byte word. The first word gives the simple name for a nuclide (*e.g.* U235) or the name of a material supplied by the user in defining that material. The second word gives the name of the cell calculation in which the cross sections were generated and, where necessary, a number to indicate the region of the cell to which the cross sections apply. The user-supplied cell calculation name must be different for different cells within the one system and should be restricted to six characters to allow for the cell region number. The last word is modified for a material (but not a nuclide) each time the material is condensed. These four characters go through the sequence ORIG, MOD1, MOD2, *etc.*

E3.3 Functions of the AUSYS Supervisor Program

The initialisation and simple editing features required for STATUS data pools are supplied by the AUSYS program. The major STATUS data set ST1, like other data pools of the AUS scheme, has allocation and retrieval facilities provided by the AUS catalogued procedure. The allocation of a new data set includes the insertion of an end of file mark which serves to initialise the data set. The ST2 data set is not normally saved; it is initialised by AUSYS at the start of an AUS calculation by copying the last TIME entry and any following CELL entries from ST1 to ST2. If ST1 is new and simply has a file mark, this results in an ST2 data set consisting of a file mark also.

Cards to modify this standard option may be included in the AUSYS input stream following the STEP named or STOP directives. The data set is identified by the card

```
   $DDnn  DISP=NEW
or
   $DDnn  DISP=OLD
```

where
DDnn is the DD name of the data set,
DISP=NEW causes an end of file to be written,
DISP=OLD results in no action.

The identifying card may be followed by a card requesting an end of file to be inserted before a nominated entry. This assists recovery following an error. The card has the form

```
   $EOF   TIME = time  CELL = cell-name
```

If CELL = cell-name is not given, the end of file is inserted before the nominated TIME entry, otherwise it is inserted before the CELL entry for the cell-name for the nominated time. A negative value of time is used as a counter. Thus -2 causes an end of file before the second TIME entry.

Cards to be added at the end of a data set may be included after a $DDnn or $EOF card. The layout of these free format cards is exactly the same as that of an entry in the data pool. However, no notice is taken of end of records. All 5-word alphanumeric names must be given as two words of length 1 to 8 characters and one word of length 1 to 4 characters. Special characters or blanks may be used to separate information. The layout of the cards is compatible with the punched output produced by the PSTAT subroutine of AUSYS. A $DD99 card is used to terminate the STATUS data cards if other AUSYS input is provided.

To restart a normal lattice burnup calculation which is correct as far as it has gone, the user does not need to supply STATUS input to restart immediately before the CHAR module. To restart after the CHAR module, the following input is required :

```
   $DD39   DISP=NEW
   $DATA   MIRANDA TIME  1  3   time   0. 0.
```

where time is the current burnup time .

APPENDIX F
A SAMPLE MODULE

```
//GSREDIT  JOB ('********/00000611',N1),G.S.ROBINSON,
//  CLASS=1,TIME=1
// EXEC FORTVCL,PLIB='AUS.FORTLIB'
//FORT.SYSIN DD *
C
C  the main routine of a sample AUS module which does some simple editing
C          following a 1 dimensional flux calculation
C
      COMMON NG,NN,NMESH,NGD,NND,NMD,A(2)
C          obtain no. of words(*4) of storage available
      CALL NARRAY(NARD)
C          allow room for buffers,etc
      NARD=NARD-5000
C          request NARD words of storage
C  IARD is returned as address of first word of storage relative to *4 vrctor A
      CALL VARRAY(A,IARD,NARD)
C  set array sizes.  these would be obtained in practice by a preliminary
C           read of data pools
      NGD=50
      NND=20
      NMD=50
      NXS=NGD*NND
C          assign storage
      LFLUX=IARD
      LXM=LFLUX+NMD*NGD*2
      LVOL=LXM+NMD
      LAY=LVOL+NMD
      LTR=LAY+NMD
      LABS=LTR+NXS
      LANUF=LABS+NXS
      LSCAT=LANUF+NXS
      LANS=LSCAT+NGD*NGD*NND
      LAANS=LANS+NND*NGD*2
      LMAX=LAANS+NND*2
      IF(LMAX.LE.IARD+NARD)GO TO 3
      WRITE(3,2)
    2 FORMAT(' STORAGE EXCEEDED')
C          note error exits dont need condition code set
      CALL VEXIT(0)
C          call calculation subroutine
    3 CALL EDIT(A(LFLUX),A(LXM),A(LVOL),A(LAY),A(LTR),A(LABS),
     1 A(LANUF),A(LSCAT),A(LANS),A(LAANS))
      STOP
      END
```

Appendix F (continued)

```
      SUBROUTINE EDIT(FLUX,XM,VOL,LAY,TR,ABS,ANUF,SCAT,ANS,AANS)
      COMMON NG,NN,NMESH,NGD,NND,NMD,A(2)
      REAL*8 FLUX(NMD,NGD)
      DIMENSION XM(NMD),VOL(NMD),LAY(NMD),TR(NGD,NND),ABS(NGD,NND)
     1 ANUF(NGD,NND),SCAT(NGD,NGD,NND),ANS(NND,NGD,2),AANS(NND,2)
      REAL*8 HEAD(2)/'NU-FISS ','ABS.'/
C          read geom data pool
      CALL RDGEOM(7,NN,NMESH,XM,VOL,LAY)
C          read fluxb data pool
      CALL RDFLB( 9,NMD,NFM,NG,FLUX)
C          read xslib data pool
      CALL RDXS(10,NGD,NN,NGX,TR,ABS,ANUF,SCAT)
C          edit of nu-fiss and abs  by material and group
      DO 3 IG=1,NG
      DO 1 I=1,NN
      ANS(I,IG,1)=0.
    1 ANS(I,IG,2)=0.
      DO 2 IX=1,NMESH
      IM=LAY(IX)
      VF=VOL(IX)*FLUX(IX,IG)
      ANS(IM,IG,1)=ANS(IM,IG,1)+ANUF(IG,IM)*VF
    2 ANS(IM,IG,2)=ANS(IM,IG,2)+ABS(IG,IM)*VF
    3 CONTINUE
      DO 8 J=1,2
      WRITE(3,4)HEAD(J),(I,I=1,NN)
    4 FORMAT('0',A8,' BY MATERIAL AND GROUP'/1X,10I12)
      DO 5 I=1,NN
    5 AANS(I,J)=0.
      DO 7 IG=1,NG
      WRITE(3,6)IG,(ANS(IM,IG,J),IM=1,NN)
    6 FORMAT(1X,I3,3X,1P10E12.5/(7X,1P10E12.5))
      DO 7 IM=1,NN
    7 AANS(IM,J)=AANS(IM,J)+ANS(IM,IG,J)
      IG=0
    8 WRITE(3,6)IG,(AANS(IM,J),IM=1,NN)
      WA=0.
      WB=0.
      DO 9 IM=1,NN
      WA=WA+AANS(IM,1)
    9 WB=WB+AANS(IM,2)
      AKINF=WA/WB
      WRITE(3,10)WA,WB,AKINF
   10 FORMAT('0NUF ',1PE12.5,'   ABS ',E12.5,'   KINF ',E12.5)
C          release storage and set condition code to 1
      CALL VEXIT(1)
      STOP
      END
C
C
      SUBROUTINE VEXIT(N)
C          release storage
      CALL VARRAY
C          set condition code and exit
      CALL CEXIT(N)
      STOP
      END
      SUBROUTINE RDGEOM(IU,NN,NMESH,XM,VOL,LAY)
C          read geom data pool
      DIMENSION XM(2),VOL(2),LAY(2),HEAD(18)
      READ(IU)HEAD,ND,IGEOM,NLS
      READ(IU)NMESH,(XM(I),I=1,NMESH)
      READ(IU)
      READ(IU)L,(LAY(I),I=1,L)
      IF(NLS.NE.1)GO TO 6
C           calculate volumes
      X=0.
      V=0.
      DO 5 I=1,NMESH
      X=X+XM(I)
      IF(IGEOM-1)1,2,3
    1 VV=X
      GO TO 4
    2 VV=3.14159*X*X
      GO TO 4
    3 VV=4.18879*X*X*X
    4 VOL(I)=VV-V
    5 V=VV
      GO TO 7
C           cluster geometry
    6 READ(IU)
      READ(IU)L,(XM(NMESH+I),I=1,L)
      READ(IU)L,(LAY(NMESH+I),I=1,L)
      READ(IU)NMESH,(VOL(I),I=1,NMESH)
C           NN is material of highest number
    7 NN=0
      DO 8 I=1,NMESH
    8 IF(LAY(I).GT.NN)NN=LAY(I)
      REWIND IU
      RETURN
      END
C
C
      SUBROUTINE RDFLB(IU,NMD,NMESH,NG,FLUX)
C           read FLUXB data pool
      REAL*8 FLUX(NMD,2),EIGEN
      READ(IU)
      READ(IU)
      READ(IU)EIGEN,NG,NMESH,((FLUX(I,J),I=1,NMESH),J=1,NG)
      REWIND IU
      RETURN
      END
      SUBROUTINE RDXS(IU,NGD,NN,NG,TR,ABS,ANUF,SCAT)
C          read XSLIB data pool
      DIMENSION TR(NGD,2),ABS(NGD,2),ANUF(NGD,2),SCAT(NGD,NGD,2)
      DIMENSION IST(150),ST(150)
      EQUIVALENCE (ST(1),IST(1))
      CALL ARDP(IU,0)
      CALL ARDT(1,1,1,1,1)
      CALL ARDN(DUMY,-20)
      CALL ARDN(IST,10)
      NG=IST(3)
      IF(NN.EQ.0)NN=IST(2)
C          start material loop
      DO 3 L=1,NN
      CALL ARDP(IU,1)
      CALL ARDN(DUMY,-6)
      CALL ARDN(IST,5)
      I=20+NG+IST(1)+IST(2)+IST(3)+IST(4)
      CALL ARDN(DUMY,-I)
C          start group loop
      DO 3 I=1,NG
C          set transport,absorption and nu-fission
      CALL ARDS(LPS,LV,ST)
      TR(I,L)=ST(1)
      ABS(I,L)=ST(2)
      ANUF(I,L)=ST(3)
C          set up full scatter matrix
      DO 1 J=1,NG
    1 SCAT(J,I,L)=0.
      CALL ARDS(LPS,LV,ST)
      K=I-LPS
      DO 2 J=1,LV
    2 SCAT(J+K,I,L)=ST(J)
    3 CONTINUE
      CALL ARDP(IU,0)
      RETURN
      END
/*
//LKED.SYSLMOD DD DSN=GSR.EDIT(EDIT),DISP=(NEW,CATLG)
// EXEC AUS,LIB=ENDF200G,
//    XS1='GSR.ROTOR.XS1',DISPXS1=OLD,
//    FL2='GSR.ROTOR.FL1',DISPFL2=OLD,
//    GM1='GSR.ROTOR.GM1',DISPGM1=OLD
//GO.GSREDIT DD DSN=GSR.EDIT(EDIT),DISP=SHR
//GO.SYSIN DD *
*DD1
STEP *
      LINK GSREDIT(3,13),(7,GM1),(9,FL2),(10,XS1)
      END
STOP
/*
```

APPENDIX G
SOME MINOR MODULES OF THE AUS SYSTEM

G1. INTRODUCTION

This appendix describes a number of the minor modules of the AUS scheme. The AUSIDD module is a one-dimensional diffusion module intended for use in multigroup calculations to provide condensation spectra. It is complementary to the multidimensional POW modules [Pollard 1974]. The three simple modules ORNL, MERGEL and JOINER are used for the manipulation of cross sections. The EXPAND module duplicates all the data pool output of a cell calculation for use in initialising a global burnup calculation.

G2. AUSIDD — A ONE-DIMENSIONAL DIFFUSION MODULE

G2.1 Description

AUSIDD is a one-dimensional multigroup diffusion module which has been designed to provide neutron flux and adjoint distributions for condensation of group cross sections. The use of the general-purpose two-dimensional diffusion theory module POW [Pollard 1974] is being phased out to allow for its replacement by the three-dimensional version POW3D. However, POW3D is not well suited to many-group one-dimensional calculations and AUSIDD has been written to complement POW3D. AUSIDD has the advantage of allowing fission spectra to be material-dependent. It also differs from the POW and POW3D modules in having a finite difference scheme with mesh points at the centre of the mesh intervals.

Because it is intended that the use of AUSIDD be restricted to calculations to provide condensation spectra, the module contains no acceleration techniques apart from a re-normalisation of the total neutron loss to the source in forward calculations. In eigenvalue problems, the iteration scheme is the simple power method. In problems with upscatter, Gauss-Seidel iteration is used. This iteration over groups is combined with the power iteration. That is, only one pass through the groups is made for each power iteration. The lack of acceleration techniques may result in a large number (100 or more) of iterations in problems with many thermal groups or with little difference between the two largest eigenvalues.

The module obtains cross sections from an AUS data pool on FORTRAN unit 10. The module writes an AUS geometry data pool on FORTRAN unit 7 and an AUS FLUXB data pool on unit 9. A geometry data pool on unit 7 may be used as input.

G2.2 Input

The input to AUSIDD is in the form of keywords followed by a string of data items. The specification of geometry is the same as in the MIRANDA module [Robinson 1986c] which is compatible with that of the POW and POW3D modules. Other data are similar to the ANAUSN module [Clancy 1982]. In the description, information given in upper case should be reproduced exactly. The data are read with the SCAN free input routine [Bennett & Pollard 1967].

The geometry may be read from an AUS data pool on unit 7 or specified by the following entries:

XM ibc x_1 x_2 ... x_n obc ,
RM ibc x_1 x_2 ... x_n obc , or
RM(SPHERE) ibc x_1 x_2 ... x_n obc

where slab, cylindrical or spherical geometry respectively are implied,
ibc, and obc are the inner and outer boundary conditions, which are either zero for a reflective boundary or the linear extrapolation distance in mean free paths for a free boundary (0.71 is normally used), and
x_i is the width of the ith mesh interval in cm.

REG (x int_1 int_2 ... int_n name_1), ...

where x may be specified as MX or MR or omitted,

int_i are the mesh interval numbers which are filled by the material name_1, and
name_i may have the form of a material name or be M(j) which specifies the jth material on the cross-section data pool. Only the first 8-byte word of a material name is given.

Examples

```
RM 0 10*1  10*0.5 0.71
REG  1(1)10 CORE 11(1)20 BLNK
REG MR 1(1)10 M(1)  MR 11(1)20 M(2)
```

The other possible entries are

BSQ b — where b is the buckling in the transverse direction;

EPS e_1 — where e_1 is the required accuracy in k_{eff}, default 0.0001;

EPSP e_2 — where e_2 is the required accuracy in group flux at any point, default 0.0005;

ADJOINT — is specified if both forward and adjoint calculations are required, the default is forward only;

WRXS — causes cross sections to be printed;

WRSF — causes the calculated fluxes to be printed;

MQV int_1, int_2 ... int_n
gives the set of interval numbers int_i at which a volume source is specified, the default is an eigenvalue calculation;

QV (s_{11} s_{21} ... s_{m1}) (s_{12} s_{22} ... s_{m2})...
where s_{ij} specifies the source density in energy group i for interval int_j, m is the number of groups, and exactly m*nqv values must be given, where nqv is the number of intervals on the MQV entry; and

START — causes the calculation to begin.

If the geometry data pool is available and the default options are satisfactory, no input data (an empty data set) need be given.

G3. ORNL - A MODULE TO GENERATE ORNL CROSS SECTIONS

G3.1 Description

Many of the transport codes distributed by the Radiation Shielding Information Center at the Oak Ridge National Laboratory (ORNL) use input cross sections of similar form. These codes accept cross sections as part of the card image input stream and read them with the FIDO input routines. The ORNL module provides a partial connection between AUS and these transport codes by generating a set of ORNL cross sections from an AUS cross-section data pool. All the cross sections required in a transport calculation should be generated in one run of the ORNL module, because the group cross-section table must be of fixed form for all materials. The module takes the AUS cross sections from FORTRAN unit 10 and produces card images in the form of free or fixed format FIDO. input on FORTRAN unit 2. The first card image produced is a comment card giving the cross-section table length and the position of the total and self-scatter cross sections.

G3.2 Input

The input to ORNL is in the form of keywords, some of which are followed by an integer. In the description, the keywords are given in upper case. No input data (an empty date set) need be given if the defaults are satisfactory.

NMAT n

specifies the number of materials to be processed. That is the first n materials of the AUS data pool are used. The default is all the materials.

NL *n*

specifies the P_n order of scattering to be used. The default is to use the order on the AUS data pool.

REAC ir

gives the AUS reaction number of an optional additional reaction to be stored in position 1 of the cross section table. The default is no additional reaction.

FIXED

specifies fixed format FIDO card images. The default is free format.

ORNL
DOT3
MORSE

are keywords which specify possible variations in the cross sections. ORNL, which is the default, gives the standard ANISN [Engle 1967] format. DOT3 gives the DOT III [Rhoades & Mynatt 1973] variant of the format. MORSE gives the same format as ORNL but a different value which is more suitable in a Monte Carlo code is used for the total cross section. When the keyword MORSE is used, the total cross section is taken from AUS reaction 5 and the P_0 self-scatter term is adjusted rather than the normal reverse procedure. The simple flux weighting used with AUS reaction 5 is more suitable for Monte Carlo calculations.

G4. MERGEL - A MODULE TO MERGE MATERIAL CROSS-SECTION FILES

G4.1 Description

Many applications of the AUS scheme require that a set of cross sections for a global calculation be constructed from a number of cross-section generation calculations. MERGEL is a simple module to merge the contents of two AUS cross-section data pools which have the same group structure. The output is a new data pool containing materials from both the input data pools.

G4.2 Input

The input which is taken from FORTRAN unit 1 consists of one card image used as a title to be written as words 5 to 20 of the output data pool identification, followed by a set of integers which are read under free format. The required integers are

```
lib1 lib2 lib3  nn
(lp(i),i=l,nn),(nl(i),i=l,nn)
```

where

lib1 and lib2 are the FORTRAN unit numbers of the two input data pools,
lib3 is the FORTRAN unit number of the output data pool,
nn is the number of materials on the output data pool,
lp(i) is the position of the ith output material on the input data pool nl(i), and
nl(i) takes the value 1 for lib1 or 2 for lib2.

The unit number of lib3 may be the same as lib1 or lib2. In that case FORTRAN unit 29 is used as a scratch data set. The value of lib2 may be 0 if a selection of the materials from lib1 is required. If nn=0, the output consists of all the lib1 materials followed by all the lib2 materials.

The information on group structure and other data not related to any particular material is taken from lib1. The group energy boundaries on lib1 and lib2 are tested for compatibility before they are merged. If the values on one of the input data pools are zero, this also is regarded as acceptable provided only that the number of groups is the same on lib1 and lib2.

An additional option is available for use in cell burnup calculations. If nn is negative, the output consists of

(-nn*(ncell-1)+lp(1)) materials from lib1,
-nn materials from lib2,
remaining materials from lib1, and
remaining materials from lib2.

Here ncell is the number of cell calculations that have been performed in the burnup run, which is obtained from the ST2 STATUS data pool on FORTRAN unit 4. The intention is that lp(1) is the number of materials that are independent of burnup, nn is the number of macroscopic materials produced in each cell calculation and the remaining materials give microscopic nuclide cross sections.

G5. JOINER - A MODULE TO FORM IRRADIATION-DEPENDENT CROSS SECTIONS

The nuclide cross sections on an AUS cross section data pool for use by the CHAR module [Robinson 1986b] may be irradiation-dependent. The JOINER module is used to construct such a data pool from a normal cross-section data pool without such dependence. The data are combined into an AUS type 4 cross-section data pool in which all cross-section and scattering-matrix data are irradiation-dependent.

The data on FORTRAN unit 11, which are for one irradiation value, are added to the irradiation-dependent data on unit 10 and the output written on unit 12. No card image input is required. The nuclides on unit 11 must be in the same order as the nuclides on unit 10. Unit 10 may contain more nuclides than unit 11 and these nuclides are written unchanged on unit 12.

G6. EXPAND - A MODULE TO DUPLICATE OUTPUT FROM A CELL CALCULATION

G6.1 Description

The AUS system may be used to perform global burnup calculations in which at each burnup step a cell calculation is performed for each reactor zone to provide material cross sections and microscopic nuclide cross sections for that zone. The EXPAND module, which duplicates the data pool output from a cell calculation, is useful for initiating such a calculation.

The data pools input to the module are the pair of STATUS data pools ST1 and ST2 on FORTRAN units 4 and 5, respectively, macroscopic material cross sections on FORTRAN unit 10 and, optionally, microscopic nuclide cross sections on unit 11. A scratch data set is required on FORTRAN unit 6. The output consists of updated data pools in which all output for nominated cells has been duplicated and put in a requested order. Cell names must be exactly four characters because the module generates six-character cell names, leaving the last two characters of the eight-character cross-section data source word to describe the region within a cell. If nuclide cross sections are not updated, neither are the nuclide (as distinct from material) names on the STATUS data pool.

The module provides the facility to condense the spatial smearing factors within a cell in calculations in which the smearing of cell cross sections is followed by a further condensation step. A condensation of the smearing factors is required if the nuclide cross sections are also condensed in the second step. The EXPAND module is the only AUS module which provides for condensation of smearing factors.

G6.2 Input

The card image input on FORTRAN unit 1 consists of a string of numeric and alphanumeric data which is read by the SCAN [Bennett & Pollard 1967] free input routine. The data are

```
ntype,ifxs2,jst,jcond,dt
(cellna(i),ncell(i), (lpos(j,i),j=1,ncell(i)), (lab(j,i),j=1,ncell(i)) ,i=1,ntype)
```

where

ntype	is the number of cells to be duplicated or repositioned,
ifxs2	is 1 if microscopic nuclide data is to be updated, else 0,
jst	is 1 if the STI data pool is to be added to, or 0 if STI is to be restarted,
jcond	is the number of energy groups into which spatial smearing factors are to be condensed, or zero if no condensation,
dt	is the time step in days, given to assist identification only,
cellna	is the cell name of an input cell calculation to be duplicated or repositioned,
ncell	is the number of output cells of this type,
lpos	is a set of integers giving the position of cells in the output data pools, and
lab	is a set of labels used to modify the input cell names, which may be integers less than 100 or words consisting of two alphanumeric non-blank characters.

Example

For input data pools giving data for three cells in the order REFL, CORE,BLNK to obtain output for REFL, four copies of CORE and three copies of BLNK, the required input is

```
2 0 0 0 0.001
CORE 4  2 3 4 5  1 2 3 4
BLNK 3  6 7 8    1 2 3
```

The output data pools would contain data for the cells named

REFL,CORE01,CORE02,CORE03,CORE04,BLNK01,BLNK02,BLNK03

in that order.