AAEC/E369

AAEC/E369

AUSTRALIAN ATOMIC ENERGY COMMISSION

RESEARCH ESTABLISHMENT

LUCAS HEIGHTS

AUS – THE AUSTRALIAN MODULAR SCHEME FOR REACTOR NEUTRONICS COMPUTATIONS

by

G. S. ROBINSON

December 1975

ISBN 0 642 99721 7

AUSTRALIAN ATOMIC ENERGY COMMISSION

RESEARCH ESTABLISHMENT

LUCAS HEIGHTS

AUS – THE AUSTRALIAN MODULAR SCHEME FOR REACTOR NEUTRONICS COMPUTATIONS

by

G. S. ROBINSON

ABSTRACT

A general description is given of the AUS modular scheme for reactor neutronics calculations. The scheme currently includes modules which provide the capacity for lattice calculations, 1D transport calculations, 1 and 2D diffusion calculations (with feedback–free kinetics), and burnup calculations. Details are provided of all system aspects of AUS, but individual modules are only outlined. A complete specification is given of that part of user input which controls the calculation sequence. The report also provides sufficient details of the supervisor program and of the interface data sets to enable additional modules to be incorporated in the scheme.

National Library of Australia card number and ISBN 0 642 99721 7

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; IBM COMPUTERS; NEUTRON DIFFUSION EQUATION; NEUTRON TRANSPORT THEORY; NEUTRONS; REACTOR LATTICES

CONTENTS

(continued)

CONTENTS (continued)

*(Sections without an * are more of interest to AUS module writers than AUS users)*

* 1. INTRODUCTION

In recent years, much effort has been allocated in reactor research laboratories throughout the world to the development of modular code schemes which enable a sequence of dependent reactor calculations to be undertaken. Interest in modular schemes has arisen from advances in computer design, a desire to carry out complex calculations and a desire to minimise the duplication of coding of algorithms. The majority of such schemes are used in the reactor neutronics field, for example, the Argonne system ARC (Toppel 1968), the Savannah River scheme JOSHUA (Honeck *et al.* 1969), the British scheme WIMS–E (Sherwin 1974), and the German system KAPROS (Kusters 1973).

The approaches adopted in the various modular schemes are quite dissimilar, for there are many ways in which the objective of a system of linked calculations can be realised. The essence of a modular scheme is simply that data can be passed from one part of a computation to another without manual intervention. The data may be passed through machine core, or *via* external direct access devices such as disks, and quite sophisticated methods may be set up for the storage and retrieval of information. The modules may vary in size from small subroutines to very large programs. The means by which the modules are linked together and the methods of user data input also vary widely. All these variables affect the development of the scheme, the ease with which the scheme may be extended or modified, the flexibility of the system, and also the ease with which the system may be used.

The AUS scheme was developed with a major constraint; a minimal amount of time (about one man–year) had to be spent in its initial development. A primary objective was that existing stand–alone codes were required to be easily incorporated into the scheme; this has an important bearing on the possible inclusion of codes borrowed later from other countries. The inclusion of large codes as single modules has resulted in a scheme containing large modules which perform more than one function and hence some duplication of coding. Another result of the time constraint was that complete planning of the overall scheme was not undertaken; the scheme tended to grow rather than develop within a set plan. Lastly, standardisation of user input to the scheme has been achieved to only a limited extent. This causes some inconvenience to the user but almost all of the problem specification for a given calculation sequence has to be entered once only.

The method of dynamically linking the required module is an adaptation of that used in the previous AAEC system for linking computer codes. This unpublished system was built around the GYMEA data preparation code (Pollard & Robinson 1969) and made use of AELINK (Mason & Richardson 1969) to load the required codes, and the FORTRAN compilation subroutine FOREX (Robinson 1968) to control the calculation sequence. The AUS system, which has also been based on AELINK and FOREX, is a rationalisation and development of this previous system.

The AUS scheme has been running on an IBM360/50 at the AAEC for about three years and has now reached a fairly stable state. AUS is not dependent on a particular version of the computer operating system and it performs all input/output using the standard FORTRAN package. Extensive use of assembler language in the supervisor program does make it machine dependent.

* 2. BASIC CONCEPTS OF THE SCHEME

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.

*(Sections without an * are more of interest to AUS module writers than AUS users)*

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 OS/360 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, normally 360K bytes, 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 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/360 Job Control Language (JCL) for the AUS scheme is contained in the *AUS catalogued procedure*. The procedure has DD (data definition) cards for modules, data pools, utility data sets, and data sets specific to individual modules. Except for modules, the DD names are all of the 3– or 4–character form DDn. An AUS user may specify that particular named data sets are to be used for any of the DD cards 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 cards of the form *DDn, beginning in column one. The set of cards following each *DDn card 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 card input to a module is that it cannot contain cards 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 deck 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. Before loading the module, AUSYS writes a 'snapshot' of the path on a disk 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.SYS when the module is link edited. Although user input to the various modules has not been standardised, all modules, except a few being phased out of the scheme, 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 deck given below illustrates some of the features of the scheme. The six cards following *DD1 form the path program which, in turn, links the three modules MIRANDA, ANAUSN and EDIT1D with input following *DD2, *DD3 and *DD4 respectively. The three modules

together form a lattice calculation. The six cards following *DD1 could be replaced by the single card STEP MAE, as they constitute a standard path.

```
// EXEC AUS
//GO.SYSIN DD *
*DD1
STEP *
      LINK MIRANDA(1,2)
      LINK ANAUSN(1,3)
      LINK EDIT1D(1,4)
      END
STOP
*DD2
HEAD SR5
DEFN D2OM D2O .9975 H2O .ØØ25
DEFN U U238 .9928 U235 .ØØ72
DEFN FUEL U .Ø47829
DEFN CAN AL .Ø432264
DEFN MOD D2OM .Ø33277
REQD FUEL CAN MOD
RM Ø 5*.253 .12 5*.5Ø5686   Ø
REG 1(1)5 FUEL 6 CAN 7(1)11 MOD
RESREG Ø 1.265 .12 2.9Ø2843  Ø SMEAR 5*1 2  5*3
BUCK 3.ØØ-4
GROUPS 26 1 3(2)11(6)41 51 61 71 8Ø(4)128
START
STOP
*DD3
*DD4
1 1 3.ØØ-4  Ø Ø Ø Ø Ø
/*
```

3. CURRENT MODULES

3.1 Introduction

This section briefly describes the modules currently available and comments on the standard usage of these modules. Further information on the accessing of data pools by each module is given in Section 6. Mention is also made of a few programs which, though not fully integrated AUS modules, do create or read AUS cross–section data pools.

3.2 MIRANDA – Cross–section Data Preparation

The MIRANDA module (Robinson 1976) includes a multi–region (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 major library is AUS.ENDFB of 128 groups on which data from ENDFB IV is being progressively included to replace data expanded from the AUS.GYMEA 127 group library. (The AUS.GYMEA library is an AUS version of the GYMEA libraries; see Section 3.3).

MIRANDA is the standard AUS module for all cross–section preparations for both thermal and fast reactor calculations. Data from MIRANDA for each region of a cell is 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 GYMEA – Cross–section Data Preparation

This module is an updated version of the GYMEA code of Pollard & Robinson (1969). With the development of the MIRANDA module, GYMEA is being phased out of the AUS scheme. The 127–group cross–section libraries are of special (*i.e.* non–AUS) GYMEA form and contain resonance parameters for fuel nuclides. The data comes from a variety of sources with the UKAEA data files being the major contributors.

3.4 ICPP – Isotropic Collision Probability

The ICPP module (Clancy *et al.* 1976) has been developed particularly to calculate many–group, few–region fluxes within a cell which are suitable for further group condensation. Rapid collision probability routines are available for slab, cylindrical, spherical and cluster geometry. For many–region cell calculations after group collapsing, ICPP is preferred for slabs and clusters, and ANAUSN is preferred for spheres and cylinders. This module is based on the collision probability routines of Doherty (1969a, 1969b, 1969c, 1970).

3.5 ANAUSN – 1D, S_N Transport

The ANAUSN module (Clancy *et al.* 1976) is a general purpose, one–dimensional discrete ordinate program with anisotropic scattering which is similar to ANISN (Engle 1967), except that it has access to AUS data pools and it has some modification in numerics. Fixed source, reactivity and criticality search calculations can be performed with this module, which is the standard module for cylindrical cells and 1D transport whole reactor calculations.

3.6 WDSN – 1D, S_N Transport

The WDSN module (Clancy *et al.* 1976) is based on the WDSNST program (Brissenden & Green 1968) which includes axial leakage of the form e^{iBz}. Apart from the ability to use AUS data pools, the module includes some modifications to improve convergence. The WDSN module is now used only for some special applications following the incorporation in AUS of the more general S_N transport module ANAUSN.

3.7 EDIT1D – 1D Editing

The EDIT1D module (G. Doherty, AAEC unpublished report) provides editing and condensing facilities following any of the one–dimensional transport calculations above. Buckling searches and a method of computing axial and radial diffusion coefficients for a cell are available. A more general edit module is currently being developed to replace this early AUS module which is limited to P_0 scattering and does not access the STATUS data pool of isotopic compositions.

3.8 POW –1 and 2D Diffusion Including Kinetics

The POW module (Pollard 1974) is a general purpose, 0,1, and 2D diffusion code which includes feedback–free kinetics. The module includes general criticality search options and extensive editing facilities, including perturbation calculations. The module also includes a data preparation option and has access to two standard resonance–shielded, fast reactor cross–section sets of Hansen & Roach (1961) and ABBN (Bondarenko 1964). POW can be used to prepare kinetic data such as delayed neutron spectra from a special purpose data set.

3.9 CHAR – Burnup

The CHAR module (Robinson 1975) uses an analytical method to solve the isotope depletion equations with the cross–section library providing the chain mechanisms. Single cell or whole reactor burnup calculations may be undertaken in as many regions as desired. At the end of a burnup step, the module either remixes macroscopic material cross sections for a further

spatial flux calculation, or it simply updates the STATUS data pool of isotopic compositions for use by a data–preparation module.

3.10 AUSED – Cross–section Editing

The AUSED module (Harrington 1976) is used to load, edit, copy, punch or list an AUS cross–section data pool. It is mainly used to maintain the basic libraries for data preparation modules but, since the intermediary libraries have the same form, it may also be used for temp–orary cross–section modification. The conversion from tabulated resonance integrals to sub–group parameters is also carried out by this module.

3.11 Utility Modules

A number of simple modules are available to carry out a variety of data manipulation and editing tasks. These include:

(a) MERGEL – the module combines two AUS cross-section data pools to form one with selected isotopes and is used in building up a cross-section library.

(b) FIVE – the module plots flux spectra obtained from AUS flux data pools.

(c) XSPLOT – the module plots selected cross sections as a function of energy from AUS cross–section data pools.

3.12 Associated Programs

The following programs make limited use only of an AUS cross–section data pool. They have not been included as modules of the AUS scheme but are executed separately before or after an AUS calculation.

(a) MURAL – The British 2000–group multi–region spectrum calculation code for fast reactors (MacDougall *et al.* 1968) has been modified to produce condensed cross sections on an AUS cross-section data pool.

(b) ZHEX – A three dimensional diffusion code for hexagonal, z geometry (Doherty 1974) which accepts input cross sections from an AUS cross–section data pool.

(c) DXYZ – A companion code of ZHEX in x, y, z geometry (G. Doherty, AAEC unpublished report).

4. DETAILS OF THE SUPERVISOR PROGRAM AUSYS

*4.1 Introduction

The AUSYS program is an AELINK (Mason & Richardson 1969) control program, therefore to understand its function requires some knowledge of AELINK. AELINK consists of a number of Assembly Language programs and subroutines which provide a dynamic program management facility by the use of the assembly language macros LINK and XCTL. AELINK provides facil–ities for a control program to execute modules, modify DD names, pass OS/360 standard parameter fields to modules, and retrieve condition codes. The sequence of events in an AELINK run is:

(a) AELINK processes the input stream by loading the data following each *DDn card 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 (b). Only a 256–byte segment of AELINK always remains in core 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 disk 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 an *DD1 card 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 to be provided by the user must be exactly as underlined and punched 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 input cards

STOP

where the set of FOREX cards is a complete non–standard path.

C.

SEEK named

STEP *

Set of FOREX input cards

STOP

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

The STEP cards 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 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 cards 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). This directive, as well as

$DDn DISP=OLD

may also be followed by a set of cards to be loaded onto a STATUS data pool (Robinson 1975). The DISP=OLD form is used if data is to be added to an existing data pool.

Data required as direct input to the path is given following the directive card:

$DD99

which must be the last $DD card 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 lim–its may be changed, however, by the path (Section 4.3.5).

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)

(d) a negative integer –k which means the integer ℓ in the kth word in COMMON (Section 4.3.5) 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)
```

where × in column 6 denotes a continued statement.

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 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 effect 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)

allows punched output from the path.

4.3.4 ALIAS for DD names

In order 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	LIB
DD33	XS1
DD34	XS2
DD40	XS3
DD35	GM1
DD36	FL1
DD37	FL2
DD38	ST1
DD39	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 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.

4.3.6 Routines available to a path

The standard FORTRAN functions which are available are:

EXP, ALOG, ALOGIO, ATAN, SIN, COS, SQRT, TANH, AND, OR .

Standard subroutine sets which may be called are the I/O routines for an AUS cross–section data pool (Section 8.4) 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/360. (The END statement in FOREX is executable and serves the same purpose.)

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 the arguments for the ARDT subroutine (Section 8.4) and are all equal to 2 for a complete listing.

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

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

UPDATE

STEP named

A set of cards 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 cards required are:

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 card and a final STOP card.

Functions to delete or reorganise the data sets have not been included in AUSYS. A file of cards containing the data to be permanently included in the libraries is maintained and used to reload the libraries from time to time. The following example adds a path ABC, adds a standard LINK for the module CC, and executes the path. The DD cards change the disposition of the AUS.PATHLIB and AUS.LINKLIB data sets from SHR to OLD:

```
//          EXEC  AUS
//GO.FT31F001     DD  DISP=OLD
//GO.FT32F001     DD  DISP=OLD
//GO.SYSIN        DD  *
*DD1
UPDATE
STEP      ABC
          LINK A (1,2)
          LINK B (1,3)
          LINK CC (1,4)
          END
STOP
UPDATE
LINK CC 3 (1,2), (3,13), (5,21)
```

```
PRINT
STEP ABC
*DD2
  :
/*
```

* 5. JOB CONTROL LANGUAGE REQUIREMENTS

5.1 Introduction

The Job Control Language of OS/360 is used by AUS for all allocation and retrieval of data sets. The large number of DD cards 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 cards entered by the user for a simple AUS run are

```
//    EXEC  AUS
//GO.SYSIN  DD *
Data cards 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 card.

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 card 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 control card 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, DD13 are reserved for plotter, punch and print output respectively. The data sets on DD14 to DD20 are further temporary card image data sets which may be used as scratch data sets by a module or a path. The data sets on DD21 to DD30 are a corresponding set of unformatted data sets. A further convention is that modules use the lower numbers of each group of data sets for scratch units and that paths use the higher numbers. At present, DD14, DD15 and DD21 to DD25 are scratch data sets for at least one module.

The data sets on DD31 and DD33 to DD40 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.

The data sets on DD55 to DD60 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/360. 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, 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) 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, one could specify

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

in the first job and

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

or

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

in the second job.

Any data set name which has a node (*e.g.* AUS) present in the OS/360 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.

Other symbolic parameters are LIB (=DD31) to specify the main cross–section library and POW to specify a temporary version of the POW module. The default value for LIB is ENDFB, which specifies that the 128–group AUS.ENDFB library is to be used. Other libraries which may be used are AUS.GYMEA (127 groups), AUS.HANSEN (16 groups) and AUS.ABBN (26 groups).

5.4 Use of Other JCL Parameters

The default value of the region size for the AUS scheme is 360K bytes which is adequate for most calculations. The modules of the scheme generally make use of whatever core is available but may require more than 360K for some large calculations. To increase the core to 480K, for example, use:

```
// EXEC  AUS, any symbolic parameters,
// REGION.GO=480K
```

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 with some comments on normal usage. Note that the distinction between a cross–section library and a cross–section data pool is that a library contains standard microscopic data, which may be potential scattering and temperature dependent, while a data pool contains macroscopic or microscopic data which is specifically for some reactor system.

MIRANDA – Cross–section preparation module

FORTRAN UNIT	DD NAME	ALIAS	DESCRIPTION
FT01F001	DD2	–	Card input.
FT03F001	DD13	–	Printer output.
FT04F001	DD38	ST1	Input/Output STATUS data pool (Input in burnup calculations).
FT05F001	DD39	ST2	Input/Output pointer STATUS data pool.
FT06F001	DD21	–	Scratch.
FT07F001	DD35	GM1	Output geometry data pool.
FT08F001	DD31	LIB	Input cross–section library, normally AUS.ENDFB.
FT09F001	DD37	FL2	Output FLUXB data pool.
FT10F001	DD33	XS1	Output cross–section data pool.
FT11F001	DD34	XS2	Alternative for XS1.
FT12F001	DD40	XS3	Alternative for XS1.

ICPP – Collision probability module

FT01F001	DD2	–	Card input.
FT03F001	DD13	–	Printer output.
FT11F001	DD37	FL2	Input/Output FLUXB data pool (Input flux guess).
FT13F001	DD35	GM1	Input/Output geometry data pool.
FT14F001	DD33	XS1	Input cross–section data pool.
FT16F001	DD34	XS2	Output cross–section data pool (not normally used).

ANAUSN – 1 D Discrete ordinate transport module

FT01F001	DD2	–	Card input.
FT02F001	DD12	–	Card output.
FT03F001	DD13	–	Printer output.
FT08F001	DD21	–	Scratch.
FT09F001	DD33	XS1	Input cross–section data pool.
FT10F001	DD35	GM1	Input/Output geometry data pool.
FT11F001	DD37	FL2	Input/Output FLUXB data pool (Input flux guess).
FT12F001	DD22	–	Scratch.
FT22F001	DD34	XS2	Output cross–section data pool (not normally used).
FT23F001	DD40	XS3	Alternative for XS2.

EDIT1D – 1 D Edit module

FORTRAN UNIT	DD NAME	ALIAS	DESCRIPTION
FT01F001	DD2	–	Card input.
FT03F001	DD13	–	Printer output.
FT11F001	DD37	FL2	Input FLUXB data pool.
FT13F001	DD35	GM1	Input geometry data pool.
FT14F001	DD33	XS1	Input cross–section data pool.
FT16F001	DD34	XS2	Output cross–section data pool.

POW – 1, 2 D Diffusion and Kinetics module

FT01F001	DD2	–	Card input.
FT02F001	DD12	–	Card output.
FT03F001	DD13	–	Printer output.
AEPLOT	DD11	–	Plotter output.
FT04F001	DD57	–	Input special data set.
FT05F001	DD21	–	Scratch.
FT06F001	DD35	GM1	Input/Output geometry data pool.
FT07F001	DD14	–	Scratch.
FT08F001	DD31	LIB	Input cross–section library,(not normally used).
FT09F001	DD36	FL1	Input/Output FLUXA data pool (Input flux guess).
FT10F001	DD33	XS1	Input/Output cross–section data pool.
FT11F001	DD34	XS2	Alternative for XS1.
FT12F001	DD22	–	Scratch.
FT13F001	DD23	–	Scratch.
FT14F001	DD24	–	Scratch.
FT15F001	DD25	–	Scratch.
FT16F001	DD98	–	Dummy.
FT17F001	DD38	ST1	Not presently used.
FT18F001	DD40	XS3	Alternative for XS1.

CHAR – Burnup module

FT01F001	DD2	–	Card input.
FT03F001	DD13	–	Printer output.
FT04F001	DD35	GM1	Input geometry data pool.
FT05F001	DD36	FL1	Input FLUXA data pool.

CHAR – Burnup module (continued)

FORTRAN UNIT	DD NAME	ALIAS	DESCRIPTION
FT06F001	DD37	FL2	Input FLUXB data pool (inputs FLUXA or FLUXB).
FT07F001	DD33	XS1	Input/Output cross–section data pool.
FT08F001	DD34	XS2	Input microscopic cross–section data pool.
FT09F001	DD38	ST1	Input/Output STATUS data pool.
FT10F001	DD39	ST2	Input/Output pointer STATUS data pool.
FT11F001	DD21	–	Scratch.
FT12F001	DD22	–	Scratch.
FT13F001	DD23	–	Scratch.
FT14F001	DD40	XS3	Alternative for XS2.

AUSED – Cross–section edit module

FT01F001	DD2	–	Card input.
FT02F001	DD12	–	Card output.
FT03F001	DD13	–	Printer output.
FT04F001	DD56	–	Input special data set.
FT07F001	DD21	–	Scratch.
FT08F001	DD31	LIB	Input/Output cross–section data pool or library.
FT10F001	DD33	XS1	As for LIB.
FT11F001	DD34	XS2	As for LIB.
FT12F001	DD40	XS3	As for LIB.

MERGEL – Cross–section merge module

FT01F001	DD2	–	Card input.
FT03F001	DD13	–	Printer output.
FT04F001	DD39	ST2	Input pointer STATUS data pool.
FT10F001	DD33	XS1	Input/Output cross–section data pool or library.
FT11F001	DD34	XS2	As for XS1.
FT12F001	DD40	XS3	As for XS1.
FT29F001	DD21	–	Scratch.

FIVE – Flux plot module

FORTRAN UNIT	DD NAME	ALIAS	DESCRIPTION
FT01F001	DD2	–	Card input.
FT03F001	DD13	–	Printer output.
AEPLOT	DD11	–	Plotter output.
FT10F001	DD36	FL1	Input FLUXA data pool.
FT11F001	DD37	FL2	Input FLUXB data pool (inputs FLUXA or FLUXB).
FT12F001	DD31	LIB	Input cross–section library.
FT13F001	DD33	XS1	Input cross–section data pool.
FT14F001	DD34	XS2	Alternative for XS1.
FT20F001	DD55	–	Input special data set.

XSPLOT – Cross–section plot module

FT01F001	DD2	–	Card input.
FT03F001	DD13	–	Printer output.
AEPLOT	DD11	–	Plotter output.
FT08F001	DD31	LIB	Input cross–section data pool or library.
FT10F001	DD33	XS1	As for LIB.
FT11F001	DD34	XS2	As for LIB.
FT12F001	DD40	XS3	As for LIB.

GYMEA – Cross–section preparation module

FT01F001	DD2	–	Card input
FT02F001	DD12	–	Card output.
FT03F001	DD13	–	Printer output.
FT04F001	DD21	–	Scratch.
FT06F001	DD35	GM1	Output geometry data pool.
FT07F001	DD58	–	Input special data set.
FT08F001	DD14	–	Scratch.
FT09F001	DD15	–	Scratch.
FT10F001	DD22	–	Scratch.
FT11F001	DD59	–	Input special data set.
FT12F001	DD60	–	Input special data set.
FT20F001	DD33	XS1	Output cross–section data pool.
FT21F001	DD38	ST1	Output STATUS data pool.

GYMEA – Cross–section preparation module (continued)

FORTRAN UNIT	DD NAME	ALIAS	DESCRIPTION
FT22F001	DD39	ST2	Output pointer STATUS data pool.
FT24F001	DD24	–	Scratch.
FT25F001	DD25	–	Scratch.
	WDSN – 1 D Discrete ordinate transport module		
FT01F001	DD2	–	Card input.
FT03F001	DD13	–	Printer output.
FT09F001	DD21	–	Scratch.
FT11F001	DD37	FL2	Input/Output FLUXB data pool (Input flux guess).
FT13F001	DD35	GM1	Input/Output geometry data pool.
FT14F001	DD33	XS1	Input cross–section data pool.
FT15F001	DD98	–	Dummy.

* 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 card:

```
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 may also add his own 'standard' paths to the library and this has been of value.

Single module paths are available for the modules MIRANDA, GYMEA, WDSN, ANAUSN, ICPP 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
        LINK MIRANDA (1,2)
        LINK ICPP (1,3)
        LINK EDIT1D (1,4)
        END
STOP
STEP MAE
        LINK MIRANDA (1,2)
        LINK ANAUSN (1,3)
        LINK EDIT1D (1,4)
        END
STOP
STEP MIEAE
        LINK MIRANDA (1,2)
        LINK ICPP (1,3)
        LINK EDIT1D (1,4)
        LINK ANAUSN (1,5), (9, XS2), (22, XS1)
        LINK EDIT1D (1,6), (14, XS2), (16, XS1)
        END
STOP
STEP MIEIE
        LINK MIRANDA (1,2)
        LINK ICPP (1,3)
        LINK EDIT1D (1,4)
        LINK ICPP (1,5), (14, XS2), (16, XS1)
        LINK EDIT1D (1,6), (14, XS2), (16, XS1)
        END
STOP
```

8. AUS CROSS–SECTION DATA POOLS (XSLIB)

8.1 Introduction

An XSLIB data pool is used for the storage of neutron 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 200. 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 is single precision.

The libraries AUS. ENDFB and AUS.GYMEA which may be input to the MIRANDA module are XSLIB data pools of the subgroup type. The libraries AUS.HANSEN and AUS.ABBN which are available through the data preparation facilities of the POW module are XSLIB data pools of the tabulated cross–section 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 tabulated cross–section or subgroup data and are essentially oriented to a particular reactor or cell.

8.2 Record Layout

An XSLIB 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 8.4). 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.

8.3 Detailed Structure

8.3.1 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 groups (NG)

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 cross–section temperatures (NXST)

word 6 = number of cross–section σ_p tabulations (NSP)

word 7 = number of scatter matrix temperatures (NSCATT)

word 8 = number of scatter matrix σ_p tabulations (NSCSP)

word 9 = number of scatter matrix P_n order + 1 (NP)

word 10 = type of cross–section tabulation (INDRES) and takes the values

(a) = 1, cross sections are functions of potential scattering (for AUS.HANSEN);

(b) = 2, cross sections are functions of σ_o which depends on the total cross sections (for AUS.ABBN);

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

Contents

NN records of 11 words each

The N'th 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 cross–section temperatures

word 8 = NSP, number of cross–section σ_p tabulations

word 9 = NSCATT, number of scatter matrix temperatures

word 10 = NSCSP, number of scatter matrix σ_p tabulations

word 11 = NP, number of scatter matrix P_n order + 1 (note that NP may be zero *i.e.* 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 several cell calculations say.

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.

8.3.2 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., $-\bar{D}$ is given where $\bar{D}$ 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 reorder isotopes before analytic solution of the depletion equations. NB is a 7 decimal digit fixed point word usually PCCAAAM (see Robinson 1975)

where P = 1 for a fission product, else 0

CC = the charge number

AAA = the mass number

M = 0 for a metastable isomer, else 1.

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 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 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

$$RI/\delta u = \sum_{i=1}^{NSP-1} \frac{w_i s}{s + \sigma_{tot}^i}$$ where s is related to σ_p.

A record of NSCATT words

giving the temperatures (K) at which 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 scatter matrices are tabulated.

Cross–section and Scatter Matrix Data

The data in this block is given for each group in turn. For any group the data is 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 $\leqslant$ 2 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 is:

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

Additional entries may be added depending on the library type, namely

(a) for macroscopic materials additional parameters may be

XS(9) = radial diffusion coefficient

XS(10) = axial diffusion coefficient

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

(i) XS(9) = average peak resonance height $\bar{\sigma}_o$

XS(10) = average value of $2E_r/\Gamma$

where E_r is resonance energy and Γ is total width;

(ii) XS(9) = XS(10) = 0 implies narrow resonance theory;

(iii) XS(9) = 1., XS(10) = 0 implies very wide resonance extending over many groups; or

(iv) XS(9) = $-\sigma$ inel, XS(10) = 0 implies narrow resonance theory and inelastic scattering; and

(v) additionally XS(11) may be the ratio of isotropic to anisotropic P_o removals.

For *subgroup parameter* records the data is:

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,

where the last 2 may be omitted.

Scatter Matrices

For each P_n scattering order, the following data is given in turn unless NP=0 when no data is 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, $1 \leq KV \leq NG$

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

Notes

(1) $\sigma^{\ell}_{g \to g'} = \int_{-1}^{1} \sigma_{g \to g'}(\mu) P_{\ell}(\mu) d\mu$, where μ is the cosine of the scattering angle.

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

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

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

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

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 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.

This completes a material file

8.4 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 routine and the output routine are separate and use different buffers, one data set may be written while another is being read.

8.4.1 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,

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_o 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 record. If the parameter is greater than one, the correspond–ing 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_o scatter vectors are returned,
MNXST=MNSP=0	no cross–section records are returned,
MNSCAT=MNSCSP=MNP=0	no scatter records are returned.

8.4.2 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,

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) .

8.5 Example

The sample module given in Appendix B includes a subroutine RDXS which might be used to read a simple XSLIB data pool without tabulated cross sections. In this sample read subroutine, only the essential data has been read and the rest has been skipped. Only the data σ_{tr}, σ_a, $\nu\sigma_f$ and $\sigma_{g \rightarrow g'}$ is 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.

9. AUS GEOMETRY DATA POOLS (GEOM)

9.1 Introduction

A GEOM data pool contains information on the geometry of some reactor system or component. The data includes 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 B includes a subroutine RDGEOM for reading a GEOM data pool describing a one–dimensional geometry.

9.2 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,

word 21 = NLS is 1 normally but 5 for cluster geometry. Then 2 + ND + NLS is the number of records describing a geometry.

Mesh interval record of form NXMI, (XM(I), I=1, NXMI) 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,

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 has the meanings:

<0. implies periodic,

=0. implies reflective,

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

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,

LAYOUT is the array of material numbers (LAYOUT(IX,IY,IZ) in the 3D case) where the number corresponds to the material order on an XSLIB data pool.

This completes the geometry description except for a cluster geometry which requires the following four 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,

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 = $\sum$ MROD (J),

DR = mesh interval in cm of the radial subdivisions of a rod.

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 ,

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 a geometry. Other sets of geometry data may follow but this feature is supported only by the POW module at present.

10. AUS FLUX DATA POOLS (FLUXA AND FLUXB)

10.1 Introduction

Unfortunately a general data pool to store neutron group fluxes has not been incorporated in the AUS scheme. As an interim measure use is made of the POW module flux dump as a FLUXA data pool and of a WDSN type flux dump as a FLUXB data pool. The FLUXA data pool is used for 1D or 2D edge mesh fluxes while 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. They are both sequential unformatted data sets.

The sample module of Appendix B includes a sample subroutine RDFLB for reading a FLUXB data pool.

10.2 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, is the power or flux

word 26 = 0. or, for kinetics calculation, is the time in seconds

word 27 = number of outers or, for kinetics, 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

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,

FLX = double precursor array of edge fluxes (plus volume weighted precision 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 but this feature is supported only by the POW module.

10.3 Detailed Structure of a FLUXB 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:

First record of 6 fixed point words

word 1 = an indicator which takes the values

1 normal case, fully converged

2 normal case, not converged

5 normal half of adjoint, fully converged

6 adjoint half of adjoint, fully converged

7 normal half of adjoint, not converged

8 adjoint half of adjoint, 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

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)

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

NG = number of energy groups

NR = number of mesh intervals

FLX = the (REAL*8) mesh average scalar flux.

A set of n records of the form ((FLX(I,J,L), I=1, NR), J=1, NG) for P_n scattering calculations with n>0. The data is REAL*8.

The ℓ'th record gives $\phi_{ig\ell}$,

where $\phi_{ig\ell} = \int_{-1}^{1} \phi_{ig}(\mu)\ P_\ell(\mu)\ d\mu$

and μ is the cosine of the angle made with the outward normal.

11. AUS STATUS DATA POOLS

A STATUS data pool is used for storing isotopic compositions, smearing factors for mixing materials, and other information useful to burnup or editing modules. This data may be stored for a number of subsidiary calculations (*e.g.* cell calculations) which together form a complete neutronics model of a reactor. Additional sets of data are added to the data pool as a burnup calculation progresses. Each module adds further entries to the data pool and/or uses some of the latest data entered to build up a history of the complete calculation. A pair of STATUS data pools are used together, *i.e.* one is the main data pool and the other is a pointer to that pool.

The main STATUS data pool is normally used as the ST1 data set on DD38 and the pointer STATUS data pool as the ST2 data set on DD39. They are sequential unformatted data sets.

Complete details of the STATUS data pools are given by Robinson (1975).

12. CODING AN AUS MODULE

12.1 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 have been described in separate sections. 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/360 condition code and PARM list 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 core storage. Descriptions of some standard AAEC FORTRAN library routines which are useful in coding a module are included in the following subsections.

12.2 Use of the Condition Code

The condition code is set by a module to indicate to the supervisor whether or not 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.

12.3 Use of the PARM List

The PARM list available to a module is an internal one which is generated by the supervisor. The list contains a single floating-point word giving the CPU time remaining for the AUS calculation in minutes. This should be used by modules carrying out iterative computations to terminate properly before a time termination. The CPU time taken by the module can be found by making an initial

CALL SCLOCK

at the start of the module and then

CALL CLOCK (TIME)

returns the time in minutes. The PARM list may be accessed in FORTRAN by including the AAEC standard routine PARMGO as the main routine of the module and changing the actual main routine into a subroutine called

MAINL(LENGTH, STRING)

where LENGTH(=4) is an INTEGER*2 variable and STRING is a LOGICAL*1 vector containing the time.

12.4 Style of Input Data

Though user supplied input data to the modules of the AUS scheme has not been standardised, most modules do have a similar style of input. The input data of the GYMEA, WDSN and EDIT1D modules, which are being phased out of the scheme, do not follow the standard pattern.

Modules should use the free input routine SCAN (Bennett & Pollard 1967) to read all user input. As this is a very flexible routine, some standard features which should be adopted are outlined:

(a) Data punched in columns 1 to 72 only;

(b) Keywords of up to eight alphanumeric characters;

(c) Floating-point data punched in abbreviated form, *e.g.* 1, 1., 1.E+0, 1.–0, 1.E 0 are equivalent;

(d) Repeat notation is, for example, 4*0. ≡ 0. 0. 0. 0.;

(e) Increment notation is, for example, 1(2)7 ≡ 1 3 5 7;

(f) Special characters used only at the discretion of the user to improve readability;

(g) A card with an * column 1 is a comment card .

The data should be keyword directed, with particular keywords having the same meanings

as keywords in existing modules where feasible. Default values should be available for all possible input parameters so that data requirements are kept to a minimum for standard calculations. Several modules use the DTAV Keyword input data and automatic preluding subroutine suite (J. Pollard, AAEC unpublished report) which uses SCAN and has the features detailed above.

12.5 Use of Core Storage

All core storage available for an AUS calculation is available to a module apart from a few hundred bytes. The standard region size for AUS calculations is 360K bytes and modules are coded so that this is sufficient except for extreme calculations. Modules are also coded to have flexible dimensions for stored arrays and to automatically make use of the storage available. Thus, when a module does require more than 360K bytes, the user need only increase the region size of the AUS Job Step.

Variable dimensions are achieved either by using the FORTRAN IV 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 storage but considerably increases the time required to code a module.

The VARRAY routine (G. Cox & J. Pollard, AAEC unpublished report) is used for the dynamic procurement of core storage. The four calling sequences are:

(a) CALL NARRAY(N)

which returns the number of 4–byte words of free core storage (and from which the program must deduct storage required by buffers).

(b) CALL VARRAY(AV, IARD, N)

which procures N 4–byte words of storage and returns IARD as the address of the first word of storage made available relative to the *4–vector AV and which is aligned to a *8–word boundary. AV may be any dimensioned variable of the subroutine which calls VARRAY.

(c) CALL VARRAY(AV, IARD, –N)

which releases the core procured with (b).

(d) CALL VARRAY

which releases all storage made available and must be called before the module terminates.

12.6 Example of an AUS Module

The sample module listed in Appendix B 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 as demonstrating 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 cards listed form a complete input deck 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 card for the module is added to the AUS catalogued procedure, and the standard unit assignments for the module are added to the system data set AUS.LINKLIB.

* 13. DESCRIPTION OF SAMPLE COMPUTATION

The output from a sample AUS run is listed in Appendix C. This example has been given primarily to illustrate the simplicity with which a path to carry out a fairly complex calculation sequence can be coded. The amount of printed output has been reduced by including a DD card to eliminate page skipping and directing all module output to the dummy data set DD99. Of course this is not a normal practice and the individual modules have options to reduce printed output. Note that the scheme firstly prints the complete input stream.

The calculation performed is the burnup of a fuel element in cylindrical geometry with a reflective boundary condition using the MIRANDA, ANAUSN, and CHAR modules. The time in days and the reactivity are stored in the T and AK vectors as the calculation progresses. The reactivity is obtained from a FLUXB data pool on the FL2 data set which is assigned to FORTRAN Unit 9. The time could have been obtained from the STATUS data pool and not coded directly. Cross sections are initially obtained from MIRANDA and the system burnt for 23 days in three steps (each of 7.667 days) using the ANAUSN and CHAR modules with CHAR mixing new macroscopic cross sections after each step. At this stage, cross sections for the partly burnt system are obtained from MIRANDA and the resulting reactivity compared with that obtained using cross sections for the clean system. If the agreement is satisfactory no further cross sections are obtained from MIRANDA and the remaining burnup is carried out in 23–day steps using ANAUSN and CHAR only. A summary of results is printed at the end of the calculation.

14. ACKNOWLEDGEMENTS

As the major tool for neutronics calculations within the AAEC, the AUS scheme is the result of the efforts of many members of the Theoretical Physics Section. Major contributions to the development of modules and the preparation of cross–section libraries have been made by Dr. J. Pollard, Dr. B. Clancy, Dr. G. Doherty (now of Wollongong University) and Mrs. B. Harrington. In addition, Dr. J. Pollard provided the initial impetus for the AUS scheme and his encouragement and assistance with the development of many system aspects of AUS are gratefully acknowledged.

15. REFERENCES

Bennett, N.W. & Pollard, J.P. (1967) – SCAN – a free input subroutine for the IBM360. AAEC/TM399.

Bondarenko, I.I. (ed.) (1964) – Group Constants for Nuclear Reactor Calculations. Consultants Bureau, N.Y.

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., Doherty, G. and Robinson, G.S. (1976) – One dimensional transport modules of the AUS scheme. AAEC to be published.

Doherty, G. (1969 a) – Some methods of calculating first flight collision probabilities in slab and cylindrical lattices. AAEC/TM489.

Doherty, G. (1969 b) – Solution of the multigroup collision probability equations. AAEC/E197.

Doherty, G. (1969 c) – Solution of some problems by collision probability methods. AAEC/E199.

Doherty, G. (1970) – Collision probability calculations in cluster geometry. AAEC/E205.

Doherty, G. (1974) – ZHEX – A three dimensional diffusion code for hexagonal, z geometry. AAEC/E307.

Engle, W.W. (1967) – A user's manual for ANISN – A one dimensional discrete ordinates transport code with anisotropic scattering. K–1693.

Hansen, G.E. & Roach, W.H. (1961) – Six– and sixteen–group cross sections for fast and intermediate critical assemblies. LAMS–2543.

Harrington, B.V. (1976) – AUS module AUSED – an editing program for AUS cross–section data pools. AAEC to be published.

Honeck, H.C., Suich, J.E., Jensen, J.C., Bailey, C.E. & Stewart, J.W. (1969) – JOSHUA – a reactor physics computational system. The effective use of computers in the nuclear industry. CONF–690401, p.324.

Kusters, H. (ed.) (1973) – Progress in fast reactor physics in the Federal Republic of Germany. KFK–1632.

MacDougall, J.D., Ross, R.W. & Rowlands, J.L. (1968) – The calculation of neutron spectra and group averaged cross sections using the computer programs FRESCO and MURAL. AEEW–M843.

Mason, C. & Richardson, D.J. (1969) – AELINK – a facility for the dynamic linkage of independent IBM360 computer programs. Proc. Fourth Aust. Computer Conf., Adelaide, August 1969, p.363.

Pollard, J.P. & Robinson, G.S. (1969) – GYMEA – a nuclide depletion, space independent, multigroup neutron diffusion, data preparation code (IBM360 version). Unpublished AAEC report (an updated version of AAEC/E147).

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

Robinson, G.S. (1968) – FOREX – A FORTRAN compilation subroutine for the IBM360. AAEC/E190.

Robinson, G.S. (1976) – AUS module MIRANDA – a multiregion resonance theory, data preparation code. AAEC to be published.

Robinson, G.S. (1975) – AUS burnup module CHAR and the associated STATUS data pool. AAEC/E372.

Sherwin, J. (1974) – User's guide to the WIMS–E modular scheme. AEEW–R881.

Toppel, B.J. (1968) – The Argonne reactor computation system, ARC. ANL–7332.

APPENDIX A

THE AUS CATALOGUED PROCEDURE

```
//*       "AUS" QUERIES: G. ROBINSON
//AUS     PROC  LIB=ENDFB,POW='AUS.POW',PLIM=60000,CLIM=20000,
//              PQ=1,SQ=10,
//              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'
//ALLOC   EXEC  PGM=NEWEDF,REGION=10K,
//           PARM=(&DISPXS1,&DISPXS2,&DISPGM1,&DISPFL1,&DISPFL2,&DISPST1)
//STEPLIB   DD  DSN=AUS.SYS,DISP=SHR
//ALL1      DD  DSN=&XS1,DISP=(&DISPXS1,CATLG),SPACE=(TRK,(&PQ,&SQ)),
//              UNIT=SYSDA,DCB=(RECFM=VBS,BLKSIZE=920)
//ALL2      DD  DSN=&XS2,DISP=(&DISPXS2,CATLG),SPACE=(TRK,(&PQ,&SQ)),
//              UNIT=SYSDA,DCB=(RECFM=VBS,BLKSIZE=920)
//ALL3      DD  DSN=&GM1,DISP=(&DISPGM1,CATLG),SPACE=(TRK,(&PQ,&SQ)),
//              UNIT=SYSDA,DCB=(RECFM=VBS,BLKSIZE=920)
//ALL4      DD  DSN=&FL1,DISP=(&DISPFL1,CATLG),SPACE=(TRK,(&PQ,&SQ)),
//              UNIT=SYSDA,DCB=(RECFM=VBS,BLKSIZE=920)
//ALL5      DD  DSN=&FL2,DISP=(&DISPFL2,CATLG),SPACE=(TRK,(&PQ,&SQ)),
//              UNIT=SYSDA,DCB=(RECFM=VBS,BLKSIZE=920)
//ALL6      DD  DSN=&ST1,DISP=(&DISPST1,CATLG),SPACE=(TRK,(&PQ,&SQ)),
//              UNIT=SYSDA,DCB=(RECFM=VBS,BLKSIZE=920)
//ALL7      DD  DSN=&XS3,DISP=(&DISPXS3,CATLG),SPACE=(TRK,(&PQ,&SQ)),
//              UNIT=SYSDA,DCB=(RECFM=VBS,BLKSIZE=920)
//GO      EXEC  PGM=AELINK,PARM=L,REGION=360K
//SYSPRINT  DD  SYSOUT=A
//AUSYS     DD  DSN=AUS.SYS(AUSYS),DISP=SHR
//GYMEA     DD  DSN=GYMEA.REL19(GYMEA),DISP=SHR
//POW       DD  DSN=&POW.(PROGPGM),DISP=(SHR,PASS)
//WDSN      DD  DSN=AUS.WDSNST(WDSN),DISP=SHR
//ICPP    DD    DSN=BEC.ICPP(PROGPGM),DISP=SHR
//EDIT1D    DD  DSN=AUS.EDIT1D(EDIT1D),DISP=SHR
//MERGEL    DD  DSN=GSR.MERGEL(MERGEL),DISP=SHR
//AUSED     DD  DSN=AUS.AUSED(PROGPGM),DISP=SHR
//CHAR      DD  DSN=GSR.CHAR(CHAR),DISP=SHR
//ANAUSN    DD  DSN=BEC.ANAUSN(PROGPGM),DISP=SHR
//MIRANDA   DD  DSN=GSR.MIRANDA(PROGPGM),DISP=SHR
//FIVE      DD  DSN=AUS.FIVE(PROGPGM),DISP=SHR
//EDIT      DD  DSN=AUS.EDIT(PROGPGM),DISP=SHR
//DD1       DD  UNIT=SYSDA,SPACE=(TRK,(1,10)),
//              DCB=(RECFM=FB,BLKSIZE=880,LRECL=80)
//DD2       DD  UNIT=SYSDA,SPACE=(TRK,(1,10)),
//              DCB=(RECFM=FB,BLKSIZE=880,LRECL=80)
//DD3       DD  UNIT=SYSDA,SPACE=(TRK,(1,10)),
//              DCB=(RECFM=FB,BLKSIZE=880,LRECL=80)
//DD4       DD  UNIT=SYSDA,SPACE=(TRK,(1,10)),
//              DCB=(RECFM=FB,BLKSIZE=880,LRECL=80)
//DD5       DD  UNIT=SYSDA,SPACE=(TRK,(1,10)),
//              DCB=(RECFM=FB,BLKSIZE=880,LRECL=80)
//DD6       DD  UNIT=SYSDA,SPACE=(TRK,(1,10)),
//              DCB=(RECFM=FB,BLKSIZE=880,LRECL=80)
//DD7       DD  UNIT=SYSDA,SPACE=(TRK,(1,10)),
//              DCB=(RECFM=FB,BLKSIZE=880,LRECL=80)
//DD8       DD  UNIT=SYSDA,SPACE=(TRK,(1,10)),
//              DCB=(RECFM=FB,BLKSIZE=880,LRECL=80)
//DD9       DD  UNIT=SYSDA,SPACE=(TRK,(1,10)),
//              DCB=(RECFM=FB,BLKSIZE=880,LRECL=80)
//DD10      DD  UNIT=SYSDA,SPACE=(TRK,(1,10)),
//              DCB=(RECFM=FB,BLKSIZE=880,LRECL=80)
//DD11      DD  SYSOUT=C
```

```
//DD12     DD  SYSOUT=B,OUTLIM=&LLIM,
//              DCB=(RECFM=FB,BLKSIZE=800,LRECL=80)
//DD13     DD  SYSOUT=A,OUTLIM=&PLIM,
//              DCB=(RECFM=FBA,BLKSIZE=1330,LRECL=133)
//DD14     DD  UNIT=SYSDA,SPACE=(CYL,(1,1)),
//              DCB=(RECFM=FB,BLKSIZE=880,LRECL=80)
//DD15     DD  UNIT=SYSDA,SPACE=(CYL,(1,1)),
//              DCB=(RECFM=FB,BLKSIZE=880,LRECL=80)
//DD16     DD  UNIT=SYSDA,SPACE=(CYL,(1,1)),
//              DCB=(RECFM=FB,BLKSIZE=880,LRECL=80)
//DD17     DD  UNIT=SYSDA,SPACE=(CYL,(1,1)),
//              DCB=(RECFM=FB,BLKSIZE=880,LRECL=80)
//DD18     DD  UNIT=SYSDA,SPACE=(CYL,(1,1)),
//              DCB=(RECFM=FB,BLKSIZE=880,LRECL=80)
//DD19     DD  UNIT=SYSDA,SPACE=(CYL,(1,1)),
//              DCB=(RECFM=FB,BLKSIZE=880,LRECL=80)
//DD20     DD  UNIT=SYSDA,SPACE=(CYL,(1,1)),
//              DCB=(RECFM=FB,BLKSIZE=880,LRECL=80)
//DD21     DD  UNIT=SYSDA,SPACE=(CYL,(1,1)),DCB=(RECFM=VBS,BLKSIZE=920)
//DD22     DD  UNIT=SYSDA,SPACE=(CYL,(1,1)),DCB=(RECFM=VBS,BLKSIZE=920)
//DD23     DD  UNIT=SYSDA,SPACE=(CYL,(1,1)),DCB=(RECFM=VBS,BLKSIZE=920)
//DD24     DD  UNIT=SYSDA,SPACE=(CYL,(1,1)),DCB=(RECFM=VBS,BLKSIZE=920)
//DD25     DD  UNIT=SYSDA,SPACE=(CYL,(1,1)),DCB=(RECFM=VBS,BLKSIZE=920)
//DD26     DD  UNIT=SYSDA,SPACE=(CYL,(1,1)),DCB=(RECFM=VBS,BLKSIZE=920)
//DD27     DD  UNIT=SYSDA,SPACE=(CYL,(1,1)),DCB=(RECFM=VBS,BLKSIZE=920)
//DD28     DD  UNIT=SYSDA,SPACE=(CYL,(1,1)),DCB=(RECFM=VBS,BLKSIZE=920)
//DD29     DD  UNIT=SYSDA,SPACE=(CYL,(1,1)),DCB=(RECFM=VBS,BLKSIZE=920)
//DD30     DD  UNIT=SYSDA,SPACE=(CYL,(1,1)),DCB=(RECFM=VBS,BLKSIZE=920)
//DD31     DD  DSN=AUS.&LIB,DISP=SHR
//DD33     DD  DSN=&XS1,DISP=OLD,SPACE=(TRK,(&PQ,&SQ),RLSE)
//DD34     DD  DSN=&XS2,DISP=OLD,SPACE=(TRK,(&PQ,&SQ),RLSE)
//DD35     DD  DSN=&GM1,DISP=OLD,SPACE=(TRK,(&PQ,&SQ),RLSE)
//DD36     DD  DSN=&FL1,DISP=OLD,SPACE=(TRK,(&PQ,&SQ),RLSE)
//DD37     DD  DSN=&FL2,DISP=OLD,SPACE=(TRK,(&PQ,&SQ),RLSE)
//DD38     DD  DSN=&ST1,DISP=OLD,SPACE=(TRK,(&PQ,&SQ),RLSE)
//DD39     DD  UNIT=SYSDA,SPACE=(CYL,(1,3)),DCB=(RECFM=VBS,BLKSIZE=920)
//DD40     DD  DSN=&XS3,DISP=OLD,SPACE=(TRK,(&PQ,&SQ),RLSE)
//DD54     DD  DSN=AUS.EDITCOM,DISP=SHR
//DD55     DD  DSN=AUS.FIVECOM,DISP=SHR
//DD56     DD  DSN=AUS.ALSEDCOM,DISP=SHR
//DD57     DD  DSN=AUS.PCWCOM,
//              DCB=(RECFM=FB,BLKSIZE=80,LRECL=80),DISP=SHR
//DD58     DD  DSN=GYMEA.MON,DISP=SHR
//DD59     DD  DSN=GYMEA.XS4,DISP=SHR
//DD60     DD  DSN=GYMEA.SC4,DISP=SHR
//DD98     DD  DUMMY
//DD99     DD  DUMMY
//SELINKOUT DD UNIT=SYSDA,SPACE=(TRK,(2))
//EXAMPLE1 DD  DSN=AUS.EXAMPLE1,DISP=SHR
//FT30F001 DD  UNIT=SYSDA,SPACE=(CYL,(1,3)),
//              DCB=(RECFM=VBS,BLKSIZE=7294,BUFNO=1)
//FT31F001 DD  DSN=AUS.PATHLIB,DISP=SHR
//FT32F001 DD  DSN=AUS.LINKLIB,DISP=SHR
//*             END OF PROCEDURE "AUS"
```

APPENDIX B

A SAMPLE MODULE

```
//GSR      JOB ('PH109075/P22PHNRA',N1),G.S.ROBINSON,
//             CLASS=J,
//             MSGLEVEL=(2,0),
//             TIME=5
// EXEC FORTHCL,PLIB='PHYS.FORTLIB',PLIB1='AUS.SYS'
//FORT.SYSIN DD *
      SUBROUTINE MAINL(L,PARM)
C
C THE MAIN ROUTINE OF A SAMPLE AUS MODULE WHICH DOES SOME SIMPLE EDITING
C  FOLLOWING A 1 DIMENSIONAL FLUX CALCULATION
C
      LOGICAL*1 PARM(4),T(4)
      INTEGER*2 L
      COMMON TLEFT,NG,NN,NMESH,NGD,NND,NMD,A(2)
      EQUIVALENCE (TLEFT,T(1))
C
C  START A C.P.U. CLOCK .  TIME FROM HERE IN MINS. GIVEN BY CALL CLOCK(TIME)
C
      CALL SCLOCK
C
C  OBTAIN THE TIME REMAINING FOR AUS FROM THE PARM FIELD.  STORE IN TLEFT
C
      DO 1 I=1,4
    1 T(I)=PARM(I)
C
C  OBTAIN NO. OF WORDS(*4) OF STORAGE AVAILABLE
C
      CALL NARRAY(NARD)
C
C  ALLOW ROOM FOR BUFFERS,ETC
C
      NARD=NARD-5000
C
C  REQUEST NARD WORDS OF STORAGE
C  IARD IS RETURNED AS ADDRESS OF FIRST WORD OF STORAGE RELATIVE TO *4 VECTOR A
C
      CALL VARRAY(A,IARD,NARD)
C
C  SET ARRAY SIZES.  THESE WOULD BE OBTAINED IN PRACTICE BY A PRELIMINARY
C   READ OF DATA POOLS
C
      NGD=50
      NND=10
      NMD=50
      NXS=NGD*NND
C
C  ASSIGN STORAGE
C
      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
C  NOTE ERROR EXITS DONT NEED CONDITION CODE SET
C
      CALL EXIT
C
C  CALL CALCULATION SUBROUTINE
C
    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
      SUBROUTINE EDIT(FLUX,XM,VOL,LAY,TR,ABS,ANUF,SCAT,ANS,AANS)
      COMMON TLEFT,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
C  READ GEOM DATA POOL
C
      CALL RDGEOM(7,NN,NMESH,XM,VOL,LAY)
C
C  READ FLUX8 DATA POOL
C
      CALL RDFLB( 9,NMD,NFM,NG,FLUX)
C
C  READ XSLIB DATA POOL
C
      CALL RDXS(10,NGD,NN,NGX,TR,ABS,ANUF,SCAT)
C
C  EDIT OF NU-FISS AND ABS  BY MATERIAL AND GROUP
C
      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'/1X10I12)
      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(1X13,3X1P10E12.5)
      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
C  RELEASE STORAGE AND SET CONDITION CODE TO 1
C
      CALL VEXIT(1)
      STOP
      END
      SUBROUTINE VEXIT(N)
C
C  RELEASE STORAGE
C
      CALL VARRAY
C
C  SET CONDITION CODE AND EXIT
C
      CALL CEXIT(N)
      STOP
      END
      SUBROUTINE RDGEOM(IU,NN,NMESH,XM,VOL,LAY)
C
C READ GEOM DATA POOL
C
      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
C  CALCULATE VOLUMES
C
      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
C  CLUSTER GEOMETRY
C
    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
C  NN IS MATERIAL OF HIGHEST NUMBER
C
    7 NN=0
      DO 8 I=1,NMESH
    8 IF(LAY(I).GT.NN)NN=LAY(I)
```

```
      REWIND IU
      RETURN
      END
      SUBROUTINE RDFLB(IU,NMD,NMESH,NG,FLUX)
C
C  READ FLUXB DATA POOL
C
      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
C  READ XSLIB DATA POOL
C
      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
C  START MATERIAL LOOP
C
      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
C  START GROUP LOOP
C
      DO 3 I=1,NG
C
C  SET TRANSPORT,ABSORPTION AND NU-FISSION
C
      CALL ARDS(LPS,LV,ST)
      TR(I,L)=ST(1)
      ABS(I,L)=ST(2)
      ANUF(I,L)=ST(3)
C
C  SET UP FULL SCATTER MATRIX
C
      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)
//LKED.SYSIN DD *
 INCLUDE SYSLIB(PARMGO)
 ENTRY PARMGO
/*
// EXEC AUS,XS1='AUS.PDXXS2',GM1='AUS.PDXGM1',FL2='AUS.PDXFLX2',
//  DISPXS1=SHR,DISPGM1=SHR,DISPFL2=SHR
//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 C

A SAMPLE AUS OUTPUT

```
//GSRC     JOB ('PH109075/P22PHNKA',N1),G.S.ROBINSON,                    JOB  42
//  CLASS=A,MSGLEVEL=(2,0),TIME=40
// EXEC AUS,LIB=GYMEA
IEF648I INVALID DISP FIELD- PASS SUBSTITUTED
IEF648I INVALID DISP FIELD- PASS SUBSTITUTED
IEF648I INVALID DISP FIELD- PASS SUBSTITUTED
IEF648I INVALID DISP FIELD- PASS SUBSTITUTED
IEF648I INVALID DISP FIELD- PASS SUBSTITUTED
IEF648I INVALID DISP FIELD- PASS SUBSTITUTED
IEF648I INVALID DISP FIELD- PASS SUBSTITUTED
IEF142I - STEP WAS EXECUTED - COND CODE 0000
IEF373I STEP /ALLOC    / START 75084.2102
IEF374I STEP /ALLOC    / STOP  75084.2102 CPU    0MIN 02.48SEC MAIN    6K LCS    0K
*** CONDITION CODE =  000(HEX)
//GO.DD13 DD DCB=(RECFM=FB,BLKSIZE=1330,LRECL=133)
IEF648I INVALID DISP FIELD- PASS SUBSTITUTED
IEF648I INVALID DISP FIELD- PASS SUBSTITUTED
IEF648I INVALID DISP FIELD- PASS SUBSTITUTED
IEF648I INVALID DISP FIELD- PASS SUBSTITUTED
IEF648I INVALID DISP FIELD- PASS SUBSTITUTED
IEF648I INVALID DISP FIELD- PASS SUBSTITUTED
IEF648I INVALID DISP FIELD- PASS SUBSTITUTED
//GO.SYSIN DD *
IEF142I - STEP WAS EXECUTED - COND CODE 0000
IEF373I STEP /GO       / START 75084.2102
IEF374I STEP /GO       / STOP  75084.2241 CPU   37MIN 15.80SEC MAIN 358K LCS    0K
*** CONDITION CODE =  00C(HEX)
IEF375I  JOB /GSRC     / START 75084.2102
IEF376I  JOB /GSRC     / STOP  75084.2241 CPU   37MIN 18.28SEC
HIGHEST CONDN CODE =  00C(HEX)
```

```
*CC1
STEP *
C  SAMPLE AUS RUN ILLUSTRATING PATH CODING IN PARTICULAR
C  MODULE OUTPUT IS NOT PRINTED
      DIMENSION T(20),AK(20)
      ALTER (9,FL2)
      T(1)=0.
      J=1
      CALL FLUX(1,AK(1))
      DO 1 I=1,3
      LINK CHAR(1,4),(3,99)
      J=J+1
      T(J)=T(J-1)+7.66667
    1 CALL FLUX(0,AK(J))
      CALL FLUX(1,AKK)
      ITEST=0
      IF(ABS(AK(J)-AKK).GT.0.002)ITEST=1
      AK(J)=AKK
      DO 2 I=1,6
      LINK CHAR(1,5),(3,99)
      J=J+1
      T(J)=T(J-1)+23.
    2 CALL FLUX(ITEST,AK(J))
      WRITE(3,3)
    3 FORMAT('  SUMMARY OF RESULTS'/'     TIME       KEFF')
      WRITE(3,4)(T(I),AK(I),I=1,J)
    4 FORMAT(1X2F10.4)
      END
      SUBROUTINE FLUX(ITEST,AKEFF)
      IF(ITEST.NE.0)LINK MIRANDA(1,2),(3,99)
      LINK ANAUSN(1,3),(3,99)
      READ(9)
      READ(9)
      READ(9)AKEFF
      AKEFF=1./AKEFF
      RETURN
      END
STOP
*DD2
HEAD POX  PK5 POISONED ELEMENTS NU B/U
DEFN FUELA U235 4.357-4 U238 1.089-4 AL 3.585-2
DEFN FUELB U235 4.336-4 U238 1.083-4 AL 3.568-2
DEFN FUELC U235 4.311-4 U238 1.078-4 AL 3.544-2
DEFN FUELD U235 4.323-4 U238 1.081-4 AL 3.557-2
DEFN OUTER AL 6.019-2 CD 3.475-5
DEFN D2CA D2O 3.31-2
DEFN D2CB D2O 3.31-2
DEFN D2CC D2O 3.31-2
REGD FUELA FUELB FUELC FUELD OUTER D2CA D2CB D2OC
RM C 3*1.01333 .171 .32 .171 .32 .171 .32 .171 .42 .2 6*.549 0
REG 1 2 3 D2OA 4 FUELA 5 D2OB 6 FUELB 7 D2OB 8 FUELC
    9 D2OB 10 FUELD 11 D2OB 12 OUTER 13(1)18 D2OC
RESREG 0 3.04 .171 .32 .171 .32 .171 3.914 0
SMEAR 3*1 2 3 4 5 6 7 8 8*9
GROUPS 21 1 5(4)53 58 68 79 87(10)127
BUCK 2.238-3
ISOTLIB 11 FUELA FUELB FUELC FUELD OUTER BURNUP  PO
OUTPUT ZERO AUS 10 PU GEOM
START
STOP
*DD3
```

```
PDX
ALS
BSC 2.238-3
WRSF
END
*DD4
FLUX DENSITY 1.8E+14 FOR $FUEL
STEP 7.66667 1
XSLIB 8
REMIX
START
*DD5
FLUX DENSITY 1.8E+14 FOR $FUEL
STEP 23 1
XSLIB 8
REMIX
START
```

```
1  ALSYS CODE SUPERVISOR    * 25 MAR 75*
CSTEP  *


    ISN           LIST OF SOURCE STATEMENTS

      1      C  SAMPLE AUS RUN ILLUSTRATING PATH CODING IN PARTICULAR
      2      C  MODULE OUTPUT IS NOT PRINTED
      3            DIMENSION T(20),AK(20)
      4            ALTER (9,FL2)
      5            T(1)=0.
      6            J=1
      7            CALL FLUX(1,AK(1))
      8            DO 1 I=1,3
      9            LINK CHAR(1,4),(3,99)
     10            J=J+1
     11            T(J)=T(J-1)+7.00667
     12          1 CALL FLUX(0,AK(J))
     13            CALL FLUX(1,AKK)
     14            ITEST=0
     15            IF(ABS(AK(J)-AKK).GT.0.002)ITEST=1
     16            AK(J)=AKK
     17            DO 2 I=1,6
     18            LINK CHAR(1,5),(3,99)
     19            J=J+1
     20            T(J)=T(J-1)+23.
     21          2 CALL FLUX(ITEST,AK(J))
     22            WRITE(3,3)
     23          3 FORMAT('  SUMMARY OF RESULTS'/'    TIME       KEFF')
     24            WRITE(3,4)(T(I),AK(I),I=1,J)
     25          4 FORMAT(1X2F10.4)
     26            END
C           ----------------------
C           ----------------------
     27            SUBROUTINE FLUX(ITEST,AKEFF)
     28            IF(ITEST.NE.0)LINK MIRANDA(1,2),(3,99)
     29            LINK ANAUSN(1,3),(3,99)
     30            REAC(9)
     31            READ(9)
     32            READ(9)AKEFF
     33            AKEFF=1./AKEFF
     34            RETURN
     35            END
C           ----------------------
C           ----------------------
ERROR LEVEL OF 0

0CODE          CALLED FROM ISN    4 OF PATH AT TIME       0.04 MINS FROM START
CUNIT ASSIGNMENTS
 (FT01F001,    ,DD 1), (FT03F001,    ,DD13), (FT09F001,FL2,DD37), (
0CODE MIRANDA  CALLED FROM ISN   28 OF PATH AT TIME       0.04 MINS FROM START
CUNIT ASSIGNMENTS
 (FT01F001,    ,DD 2), (FT03F001,    ,DD99), (FT04F001,ST1,DD38), (FT05F001,ST2,DD39), (FT06F001,    ,DD21), (FT07F001,GM1,DD35),
 (FT08F001,LIB,DD31), (FT09F001,FL2,DD37), (FT10F001,XS1,DD33), (FT11F001,XS2,DD34), (FT12F001,XS3,DD40), (
1  ALSYS CODE SUPERVISOR    * 25 MAR 75*
0CODE ANAUSN   CALLED FROM ISN   29 OF PATH AT TIME       5.97 MINS FROM START
CUNIT ASSIGNMENTS
 (FT01F001,    ,DD 3), (FT02F001,    ,DD12), (FT03F001,    ,DD99), (FT08F001,    ,DD21), (FT09F001,XS1,DD33), (FT10F001,GM1,DD35),
 (FT11F001,FL2,DD37), (FT12F001,    ,DD22), (FT22F001,XS2,DD34), (FT23F001,XS3,DD40), (
1  ALSYS CODE SUPERVISOR    * 25 MAR 75*
```

```
OCODE CHAR     CALLED FROM ISN    9 OF PATH AT TIME       7.79 MINS FROM START
CUNIT ASSIGNMENTS
 (FT01F001,    ,DD 4), (FT03F001,    ,DD99), (FT04F001,GM1,DD35), (FT05F001,FL1,DD36), (FT06F001,FL2,DD37), (FT07F001,XS1,DD33),
 (FT08F001,XS2,DD34), (FT09F001,ST1,DD38), (FT10F001,ST2,DD39), (FT11F001,    ,DD21), (FT12F001,    ,DD22), (FT13F001,    ,DD23),
 (FT14F001,XS3,DD40), (
1  ALSYS CODE SUPERVISOR    * 25 MAR 75*
CODE ANALSN   CALLED FROM ISN   29 OF PATH AT TIME       8.58 MINS FROM START
CUNIT ASSIGNMENTS
 (FT01F001,    ,DD 3), (FT02F001,    ,DD12), (FT03F001,    ,DD99), (FT08F001,    ,DD21), (FT09F001,XS1,DD33), (FT10F001,GM1,DD35),
 (FT11F001,FL2,DD37), (FT12F001,    ,DD22), (FT22F001,XS2,DD34), (FT23F001,XS3,DD40), (
1  ALSYS CODE SUPERVISOR    * 25 MAR 75*
OCODE CHAR     CALLED FROM ISN    9 OF PATH AT TIME      10.09 MINS FROM START
CUNIT ASSIGNMENTS
 (FT01F001,    ,DD 4), (FT03F001,    ,DD99), (FT04F001,GM1,DD35), (FT05F001,FL1,DD36), (FT06F001,FL2,DD37), (FT07F001,XS1,DD33),
 (FT08F001,XS2,DD34), (FT09F001,ST1,DD38), (FT10F001,ST2,DD39), (FT11F001,    ,DD21), (FT12F001,    ,DD22), (FT13F001,    ,DD23),
 (FT14F001,XS3,DD40), (
1  ALSYS CODE SUPERVISOR    * 25 MAR 75*
CODE ANALSN   CALLED FROM ISN   29 OF PATH AT TIME      10.98 MINS FROM START
CUNIT ASSIGNMENTS
 (FT01F001,    ,DD 3), (FT02F001,    ,DD12), (FT03F001,    ,DD99), (FT08F001,    ,DD21), (FT09F001,XS1,DD33), (FT10F001,GM1,DD35),
 (FT11F001,FL2,DD37), (FT12F001,    ,DD22), (FT22F001,XS2,DD34), (FT23F001,XS3,DD40), (
1  ALSYS CODE SUPERVISOR    * 25 MAR 75*
OCODE CHAR     CALLED FROM ISN    9 OF PATH AT TIME      12.32 MINS FROM START
CUNIT ASSIGNMENTS
 (FT01F001,    ,DD 4), (FT03F001,    ,DD99), (FT04F001,GM1,DD35), (FT05F001,FL1,DD36), (FT06F001,FL2,DD37), (FT07F001,XS1,DD33),
 (FT08F001,XS2,DD34), (FT09F001,ST1,DD38), (FT10F001,ST2,DD39), (FT11F001,    ,DD21), (FT12F001,    ,DD22), (FT13F001,    ,DD23),
 (FT14F001,XS3,DD40), (
1  ALSYS CODE SUPERVISOR    * 25 MAR 75*
CODE ANALSN   CALLED FROM ISN   29 OF PATH AT TIME      13.19 MINS FROM START
CUNIT ASSIGNMENTS
 (FT01F001,    ,DD 3), (FT02F001,    ,DD12), (FT03F001,    ,DD99), (FT08F001,    ,DD21), (FT09F001,XS1,DD33), (FT10F001,GM1,DD35),
 (FT11F001,FL2,DD37), (FT12F001,    ,DD22), (FT22F001,XS2,DD34), (FT23F001,XS3,DD40), (
1  ALSYS CODE SUPERVISOR    * 25 MAR 75*
OCODE MIRANDA  CALLED FROM ISN   28 OF PATH AT TIME      14.52 MINS FROM START
CUNIT ASSIGNMENTS
 (FT01F001,    ,DD 2), (FT03F001,    ,DD99), (FT04F001,ST1,DD38), (FT05F001,ST2,DD39), (FT06F001,    ,DD21), (FT07F001,GM1,DD35),
 (FT08F001,LI8,DD31), (FT09F001,FL2,DD37), (FT10F001,XS1,DD33), (FT11F001,XS2,DD34), (FT12F001,XS3,DD40), (
1  ALSYS CODE SUPERVISOR    * 25 MAR 75*
OCODE ANALSN   CALLED FROM ISN   29 OF PATH AT TIME      22.00 MINS FROM START
CUNIT ASSIGNMENTS
 (FT01F001,    ,DD 3), (FT02F001,    ,DD12), (FT03F001,    ,DD99), (FT08F001,    ,DD21), (FT09F001,XS1,DD33), (FT10F001,GM1,DD35),
 (FT11F001,FL2,DD37), (FT12F001,    ,DD22), (FT22F001,XS2,DD34), (FT23F001,XS3,DD40), (
1  ALSYS CODE SUPERVISOR    * 25 MAR 75*
OCODE CHAR     CALLED FROM ISN   18 OF PATH AT TIME      23.31 MINS FROM START
CUNIT ASSIGNMENTS
 (FT01F001,    ,DD 5), (FT03F001,    ,DD99), (FT04F001,GM1,DD35), (FT05F001,FL1,DD36), (FT06F001,FL2,DD37), (FT07F001,XS1,DD33),
 (FT08F001,XS2,DD34), (FT09F001,ST1,DD38), (FT10F001,ST2,DD39), (FT11F001,    ,DD21), (FT12F001,    ,DD22), (FT13F001,    ,DD23),
 (FT14F001,XS3,DD40), (
1  ALSYS CODE SUPERVISOR    * 25 MAR 75*
CODE ANALSN   CALLED FROM ISN   29 OF PATH AT TIME      24.37 MINS FROM START
CUNIT ASSIGNMENTS
 (FT01F001,    ,DD 3), (FT02F001,    ,DD12), (FT03F001,    ,DD99), (FT08F001,    ,DD21), (FT09F001,XS1,DD33), (FT10F001,GM1,DD35),
 (FT11F001,FL2,DD37), (FT12F001,    ,DD22), (FT22F001,XS2,DD34), (FT23F001,XS3,DD40), (
1  ALSYS CODE SUPERVISOR    * 25 MAR 75*
OCODE CHAR     CALLED FROM ISN   18 OF PATH AT TIME      25.90 MINS FROM START
CUNIT ASSIGNMENTS
 (FT01F001,    ,DD 5), (FT03F001,    ,DD99), (FT04F001,GM1,DD35), (FT05F001,FL1,DD36), (FT06F001,FL2,DD37), (FT07F001,XS1,DD33),
 (FT08F001,XS2,DD34), (FT09F001,ST1,DD38), (FT10F001,ST2,DD39), (FT11F001,    ,DD21), (FT12F001,    ,DD22), (FT13F001,    ,DD23),
 (FT14F001,XS3,DD40), (
1  ALSYS CODE SUPERVISOR    * 25 MAR 75*
CODE ANALSN   CALLED FROM ISN   29 OF PATH AT TIME      26.92 MINS FROM START
```

```
OUNIT ASSIGNMENTS
 (FT01F001,    ,DD 3), (FT02F001,    ,DD12), (FT03F001,    ,DD99), (FT08F001,    ,DD21), (FT09F001,XS1,DD33), (FT10F001,GM1,DD35),
 (FT11F001,FL2,DD37), (FT12F001,    ,DD22), (FT22F001,XS2,DD34), (FT23F001,XS3,DD40), (
1  AUSYS CODE SUPERVISOR    * 25 MAR 75*
OCODE CHAR      CALLED FROM ISN   18 OF PATH AT TIME      26.32 MINS FROM START
OUNIT ASSIGNMENTS
 (FT01F001,    ,DD 5), (FT03F001,    ,DD99), (FT04F001,GM1,DD35), (FT05F001,FL1,DD36), (FT06F001,FL2,DD37), (FT07F001,XS1,DD33),
 (FT08F001,XS2,DD34), (FT09F001,ST1,DD38), (FT10F001,ST2,DD39), (FT11F001,    ,DD21), (FT12F001,    ,DD22), (FT13F001,    ,DD23),
 (FT14F001,XS3,DD40), (
1  AUSYS CODE SUPERVISOR    * 25 MAR 75*
CCODE ANALSN   CALLED FROM ISN   29 OF PATH AT TIME      29.40 MINS FROM START
OUNIT ASSIGNMENTS
 (FT01F001,    ,DD 3), (FT02F001,    ,DD12), (FT03F001,    ,DD99), (FT08F001,    ,DD21), (FT09F001,XS1,DD33), (FT10F001,GM1,DD35),
 (FT11F001,FL2,DD37), (FT12F001,    ,DD22), (FT22F001,XS2,DD34), (FT23F001,XS3,DD40), (
1  AUSYS CODE SUPERVISOR    * 25 MAR 75*
OCODE CHAR      CALLED FROM ISN   18 OF PATH AT TIME      30.70 MINS FROM START
OUNIT ASSIGNMENTS
 (FT01F001,    ,DD 5), (FT03F001,    ,DD99), (FT04F001,GM1,DD35), (FT05F001,FL1,DD36), (FT06F001,FL2,DD37), (FT07F001,XS1,DD33),
 (FT08F001,XS2,DD34), (FT09F001,ST1,DD38), (FT10F001,ST2,DD39), (FT11F001,    ,DD21), (FT12F001,    ,DD22), (FT13F001,    ,DD23),
 (FT14F001,XS3,DD40), (
1  AUSYS CODE SUPERVISOR    * 25 MAR 75*
CCODE ANALSN   CALLED FROM ISN   29 OF PATH AT TIME      31.56 MINS FROM START
OUNIT ASSIGNMENTS
 (FT01F001,    ,DD 3), (FT02F001,    ,DD12), (FT03F001,    ,DD99), (FT08F001,    ,DD21), (FT09F001,XS1,DD33), (FT10F001,GM1,DD35),
 (FT11F001,FL2,DD37), (FT12F001,    ,DD22), (FT22F001,XS2,DD34), (FT23F001,XS3,DD40), (
1  AUSYS CODE SUPERVISOR    * 25 MAR 75*
OCODE CHAR      CALLED FROM ISN   18 OF PATH AT TIME      32.83 MINS FROM START
OUNIT ASSIGNMENTS
 (FT01F001,    ,DD 5), (FT03F001,    ,DD99), (FT04F001,GM1,DD35), (FT05F001,FL1,DD36), (FT06F001,FL2,DD37), (FT07F001,XS1,DD33),
 (FT08F001,XS2,DD34), (FT09F001,ST1,DD38), (FT10F001,ST2,DD39), (FT11F001,    ,DD21), (FT12F001,    ,DD22), (FT13F001,    ,DD23),
 (FT14F001,XS3,DD40), (
1  AUSYS CODE SUPERVISOR    * 25 MAR 75*
CCODE ANALSN   CALLED FROM ISN   29 OF PATH AT TIME      33.74 MINS FROM START
OUNIT ASSIGNMENTS
 (FT01F001,    ,DD 3), (FT02F001,    ,DD12), (FT03F001,    ,DD99), (FT08F001,    ,DD21), (FT09F001,XS1,DD33), (FT10F001,GM1,DD35),
 (FT11F001,FL2,DD37), (FT12F001,    ,DD22), (FT22F001,XS2,DD34), (FT23F001,XS3,DD40), (
1  AUSYS CODE SUPERVISOR    * 25 MAR 75*
OCODE CHAR      CALLED FROM ISN   18 OF PATH AT TIME      34.95 MINS FROM START
OUNIT ASSIGNMENTS
 (FT01F001,    ,DD 5), (FT03F001,    ,DD99), (FT04F001,GM1,DD35), (FT05F001,FL1,DD36), (FT06F001,FL2,DD37), (FT07F001,XS1,DD33),
 (FT08F001,XS2,DD34), (FT09F001,ST1,DD38), (FT10F001,ST2,DD39), (FT11F001,    ,DD21), (FT12F001,    ,DD22), (FT13F001,    ,DD23),
 (FT14F001,XS3,DD40), (
1  AUSYS CODE SUPERVISOR    * 25 MAR 75*
CCODE ANALSN   CALLED FROM ISN   29 OF PATH AT TIME      35.77 MINS FROM START
OUNIT ASSIGNMENTS
 (FT01F001,    ,DD 3), (FT02F001,    ,DD12), (FT03F001,    ,DD99), (FT08F001,    ,DD21), (FT09F001,XS1,DD33), (FT10F001,GM1,DD35),
 (FT11F001,FL2,DD37), (FT12F001,    ,DD22), (FT22F001,XS2,DD34), (FT23F001,XS3,DD40), (
1  AUSYS CODE SUPERVISOR    * 25 MAR 75*
  SUMMARY OF RESULTS
     TIME       KEFF
      0.0       1.0007
      7.6667    1.0474
     15.3333    1.0847
     23.0000    1.0986
     46.0000    1.0945
     69.0000    1.0800
     92.0000    1.0643
    115.0000    1.0479
    138.0000    1.0310
    161.0000    1.0132
OEND OF AUSYS AFTER      36.93 MINS
```