Final Deliverables for the ACTIV Code Project
(Project No. 05-07741)

Appendix II

User Guide and Sample Problems
for the ACTIV Code System

User Guide and Sample Problems for the ACTIV Code System

Dr. John R. White
Chemical and Nuclear Engineering Department
University of Massachusetts Lowell

August 31, 1997

Introduction

This document overviews the essential information needed to set up and run various components of the ACTIV code system. The system consists of four modules, as follows:

ACTXS - This code, in conjunction with several modules and data libraries from SCALE, generates the multigroup activation library for use in ACTIV. The resultant dataset should be fully consistent with the data used in the DORT transport calculations.

ACHAIN - This routine automatically generates the chain data necessary for setting up the activation calculations in ACTIV. It generates all possible parent-daughter-process relationships consistent with the available base cross section library, the materials to be irradiated, and the options set by the user. It writes a data file that is input directly to ACTIV. This file can be easily modified as needed by the user to specialize a particular calculation.

ACTMAT - This is a small utility routine that uses nominal information about the materials to be irradiated (including impurity information) and creates a dataset of initial conditions for the nuclide vector for direct application within ACTIV.

ACTIV - This is the main computational routine and it performs the space-energy activation analysis calculations on a pointwise and/or zone-averaged basis. The focus is on the activation and decay of excore structural materials.

This user guide is broken into two major sections. The first part gives a summary of the user input required to execute each of the codes. Some brief Input Notes - that hopefully clarify any uncertainty or jargon in the Input Description - are also included. These two subsections for each of the four codes should provide enough information for efficient use of the various modules for any specific application of interest. The second part of this guide summarizes a few sample problem sequences that illustrate the interaction among the ACHAIN, ACTMAT, and ACTIV modules and demonstrate several of the options available in the individual codes. The sample problems currently use the BUGLE96-compatible ACTXS47.LIB activation library generated using the ACTXS module and other components of the SCALE system. Since the generation of this library is rather involved, it is discussed in more detail in a separate document (see Appendices I and III). Thus, the sample problems treated here highlight the use of the ACHAIN and ACTMAT utility modules (for the generation of the nuclide chain data and initial isotope densities used in ACTIV) and the actual DORT-ACTIV sequence utilized to compute the neutron flux distribution and activity profiles in a 1-D model of the Maine Yankee reactor. The sample problems are only explained briefly in this document. Since the typical user is expected

to be familiar with performing DORT shielding analyses, the reader can get as much detail as desired by studying the input files for the various cases (a list of the key files used/generated as part of the sample sequences is given in Appendix I - see the listing of the VERS2.RME file, for example).

ACTXS Input Description

The following execution and input instructions for ACTXS are taken directly from the comment lines within the code. The comments within ACTXS reflect the most current information available for the code. By including this information here, we guarantee that full consistency is maintained between the User Guide and the actual software. The FIDAS input processor is utilized (as common to DORT, SCALE4.3, etc.).

```
c
c ***************************************************************
c
c   EXECUTION INSTRUCTIONS FOR ACTXS
c
c   The code is executed in batch mode using a script file.  The following
c   files are used/created (with default file names):
c    input    - case dependent input data (ascii) (nin)
c    print    - case dependent output edit and debug print (ascii) (nprt)
c    oldlib   - input ANISN formatted cross sections containing only primary
c               nuclear data (no scattering info) (this is the Phase I
c               library) (binary) (nu1)
c    newlib   - output Phase II activation library created (ascii) (nu2)
c    decaylib - ORIGEN decay data (end6dec) (ascii) (nu3)
c    xseclib  - ORIGEN cross section data (xsectpho) (ascii) (nu4)
c
c
c   USER INPUT INSTRUCTIONS FOR ACTXS
c
c      FIDAS Input:
c
c FIDAS INPUT BLOCK #1  (primary dimension setting parameters)
c
c  1$$ Array (10 entries)
c
c1  nnuc  = number of isotopes of interest
c   nnlt  = number of isotopes in ORIGEN light element library (689)
c   nnac  = number of isotopes in ORIGEN actinide library (129)
c   nnfp  = number of isotopes in ORIGEN fission product library (879 - not
c           used)
c   ngrp  = number of energy groups
c
c6  ntabl = table length of ANISN library (should be 10)
c   nasn  = number of nuclides on ANISN library
c
c   fill remaining entries with zeros
c
c
c  END BLOCK #1  (terminate with a 't')
c
c
c FIDAS INPUT BLOCK #2  (primary input data for problem)
c
c  2$$ Array (nnuc entries)
```

```
c   id(nnuc)  -  nuclide id for isotopes of interest
c    where  id = z*10000 + a*10 + is
c    and     z = atomic number,   a = atomic weight.,
c    and    is = 0/1 ground/metastable state
c
c  3$$ Array (nnuc entries)
c   idasn(nnuc)  -  nuclide id's on ANISN library
c    note: This array has one-to-one correspondence with the id(nnuc) array.
c          Enter zero here if no ANSIN data exist for this isotope, otherwise
c          enter the ANISN identifier (from 1 to nasn).  The entries in this
c          array indicate the order of placement of nuclide i (which
c          corresponds to the nuclide denoted as id(i)) in the ANISN library.
c          If a particular isotope present in the desired nuclide list is not
c          present on the ANISN file, simply enter zero.
c
c
c  END BLOCK #2  (terminate with a 't')
c
c
c ***************************************************************************
c
```

ACTXS Input Notes

The ACTXS code will not be needed by the typical user. Activation libraries are designed to be general purpose datasets that are compatible with the specific transport/shielding library used with DORT for the flux computations. A 47-group activation library compatible with the BUGLE96 library is currently available. This activation library, known as ACTXS47.LIB, was derived from VITAMIN-B6 data and it was designed to be used specifically with the BUGLE96 library (and a compatible transport code - DORT, ANISN, etc.). A new activation library is needed only if the transport computations are performed with some other library (especially if it has a different neutron group structure).

The basic procedure for generation of an activation library for use in ACTIV is described in some detail in Appendix III (with special focus of the generation of ACTXS47.LIB). The overall process is broken into two steps or phases. The first step involves the collapse of the infinitely dilute data in the base fine-group master library into the desired broad group structure and subsequent format changes to put the Phase I Activation Library into proper form. This involves the selection of ten specific reaction cross sections and the storage of these in standard ANISN format (with a table length of 10 - i.e. no scattering data are present). The specific reactions and their associated ENDF MT numbers are listed below:

ENDF MT No.	102	107	103	16	104	105	18	27	1452	1
Reaction	n,γ	n,α	n,p	n,2n	n,d	n,t	n,f	n,abs	$\nu\sigma_f$	σ_T

The order of the cross sections in the Phase I library is essential, since ACTXS currently expects only these ten cross sections in the order given. There is no general check for consistency here - so be sure that the Phase I library has been generated properly! Only the first eight of these reactions are actually used within ACTIV.

The second step in the generation of an activation library combines the data from Phase I with a variety of information from the ENDF/B-VI version of the ORIGEN data libraries distributed as part of the SCALE 4.3 package. In particular, two card image libraries (END6DEC and XSECTPHO) are used here. These datasets contain important decay data and nuclear data and photon yield libraries for the light elements, actinides, and fission products. These libraries are read and the hollerith names, natural isotopic abundances, decay data, and appropriate branching fractions are extracted for the isotopes of interest (no neutron cross sections are taken from the ORIGEN libraries). The processing of the ORIGEN data files and the merger of these data with the Phase I library are the functions of the ACTXS code. This module was written at UMass-Lowell as a tool for integrating all the necessary nuclear data from the Phase I activation library and the two ORIGEN libraries into a single file for use in ACTIV.

The user interface to ACTXS simply includes a list of the IDs for the nuclides of interest and the corresponding material number on the Phase I ANISN-formatted library. The base isotope ID uses the ZA number of the particular isotope, where

ID = Z*10000 + A*10 + IS

and Z = atomic number
A = atomic weight
IS = 0/1 to indicate the ground/metastable state.

The input libraries are searched for the nuclides of interest and, if found, the pertinent data are added to the final Phase II activation library written in ANISN ascii format. Note that, in the ORIGEN libraries, the data associated with the first occurrence of the nuclide ID are used and, at present, only the light element and actinide portions of the libraries are searched (there is no need for fission product isotopes in the final activation library, since the emphasis in ACTIV is on the excore structural regions).

The IDs on the ANISN library are assumed to be sequenced as 1, 2, ... nasn (this is important). The idasn(i) array contains the ANISN IDs in a 1-to-1 correspondence with array id(i). Be aware that the correspondence between id(i) and idasn(i) is critical and that the user must be sure that this is correct (or the wrong cross sections will be used). Again, there is no general check for consistency here - so user beware!

The final library has a relatively simple structure and it is written in ascii format. This allows for simple editing of the file to modify existing data or to add additional information from other sources as needed. Appropriate flags identifying the content of the library are incorporated directly within the final library. In this way, ACTIV knows what information is available and it simply eliminates any transmutation paths that are not compatible with the existing library information.

ACHAIN Input Description

The following execution and input instructions for ACHAIN are taken directly from the comment lines within the main program. The internal comments within the code reflect the latest information concerning the current version. By including this information here, we assure that full consistency is maintained between the User Guide and the actual software. Note that the FIDAS input processor is also utilized as the primary user interface in ACHAIN.

```
C
```

```
c ***********************************************************************
c
c   EXECUTION INSTRUCTIONS FOR ACHAIN
c
c   The code is executed in batch mode using a script file.  The following
c   files are used/created (with default file names):
c    input    - case dependent input data (ascii) (nin)
c    print    - case dependent output edit and debug print (ascii) (nprt)
c    actlib   - input activation library generated by ACTXS (ascii) (nu1)
c    chnlib1  - output full parent-daughter-process data (ascii) (nu2)
c    chnlib2  - output filtered parent-daughter-process relationships for
c               selected activation products (ascii) (nu3)
c
c
c   USER INPUT INSTRUCTIONS FOR ACHAIN
c
c      FIDAS Input:
c
c FIDAS INPUT BLOCK #1  (primary dimension setting parameters)
c
c  1$$ Array (10 entries)
c
c   nnuc = # of initial parent isotopes
c   ngen = # of generations of daughters to follow
c   nmax = maximum # of parents for any one generation (sets array dim)
c   ngrp = number of neutron energy groups
c   nap  =  0 - given initial parent ids (stored in idop), determine all
c               possible parent-daughter-process (p-d-p) relationships
c               and output this information to chnlib1 (always)
c         >0 - also filter above p-d-p chains and select only those
c               processes that eventually lead to one of the nap desired
c               activation products (the filtered p-d-p chains are saved
c               in file chnlib2)
c
c   fill remaining entries with zeros
c
c
c  END BLOCK #1  (terminate with a 't')
c
c
c FIDAS INPUT BLOCK #2  (primary input data for problem)
c
c  2$$ Array (nnuc entries)
c   idop(nnuc)  -  nuclide id for original parents
c    where    id = z*10000 + a*10 + is
c    and       z = atomic number,   a = atomic weight,
c              is = 0/1 ground/metastable state
c
c  Note: If the id number ends in 0000 (i.e. both a = 0 and is = 0), the
c        original nuclide represents a naturally occurring element with
c        multiple stable isotopes.  In this case the natural abundance
c        information in the activation library will be used to expand
c        the element into all its stable components.
c
c  3$$ Array (nap entries)
c   idap(nap)  -  nuclide ids for desired activation products
c
c
c  END BLOCK #2  (terminate with a 't')
c
```

```
C
C ***************************************************************************
C
```

ACHAIN Input Notes

This program reads an input vector containing nuclide IDs and uses information from the activation library to generate all of the possible non-trivial nuclide chains for ngen generations. The daughters from the first generation parents (the isotopes specified in the idop array) become the second generation parents, and so on. Processes that are not allowed due to the lack of cross section data are automatically deleted from the activation chains. The daughter from any transition is identified by the addition of a suitable constant to the 6-digit nuclide ID (ZA number) for the parent isotope. The 14 possible reactions that can be treated and the required constant for each process are identified below:

Entry	Reaction	State*	Equation Representation	Constant
1	n,γ	0	${}^{A}_{Z}X + n \rightarrow {}^{A+1}_{Z}X$	+10
2	n,γ	1	${}^{A}_{Z}X + n \rightarrow {}^{A+1}_{Z}X^{*}$	+11
3	n,α	0	${}^{A}_{Z}X + n \rightarrow {}^{A-3}_{Z-2}Y + {}^{4}_{2}He$	-20030
4	n,p	0	${}^{A}_{Z}X + n \rightarrow {}^{A}_{Z-1}Y + {}^{1}_{1}H$	-10000
5	n,2n	0	${}^{A}_{Z}X + n \rightarrow {}^{A-1}_{Z}X + 2\left({}^{1}_{0}n\right)$	-10
6	n,2n	1	${}^{A}_{Z}X + n \rightarrow {}^{A-1}_{Z}X^{*} + 2\left({}^{1}_{0}n\right)$	-9
7	n,d	0	${}^{A}_{Z}X + n \rightarrow {}^{A-1}_{Z-1}Y + {}^{2}_{1}H$	-10010
8	n,t	0	${}^{A}_{Z}X + n \rightarrow {}^{A-2}_{Z-1}Y + {}^{3}_{1}H$	-10020
9	β^- decay	0	${}^{A}_{Z}X \rightarrow {}^{A}_{Z+1}Y + e^{-}$	+10000
10	β^- decay	1	${}^{A}_{Z}X \rightarrow {}^{A}_{Z+1}Y^{*} + e^{-}$	+10001
11	β^+ decay	0	${}^{A}_{Z}X \rightarrow {}^{A}_{Z-1}Y + e^{+}$	-10000
12	β^+ decay	1	${}^{A}_{Z}X \rightarrow {}^{A}_{Z-1}Y^{*} + e^{+}$	-9999
13	α decay	0	${}^{A}_{Z}X \rightarrow {}^{A-4}_{Z-2}Y + {}^{4}_{2}He$	-20040
14	isomeric transition	0	${}^{A}_{Z}X^{*} \rightarrow {}^{A}_{Z}X$	-1

*Note: A state of 0 implies the ground state and unity refers to the metastable state.

Using the above reaction-constant pairs, the daughter ID is determined from the expression: Daughter ID = Parent ID + Constant. A check to assure that the daughter is in the proper state (ground versus metastable) is also made.

The primary goal of ACHAIN is to determine all possible parent-daughter-process triplets that are consistent with the initial parent isotopes, the data available in the activation library, and the 14 different processes that are allowed in ACTIV. An outline of the algorithm used in ACHAIN to achieve this goal is as follows:

1. Input initial vector of parents and read data from the activation library.
2. If the parent ID refers to a naturally occurring element, expand that element to include all the naturally occurring isotopes.

Loop over ngen generations

3. Find daughters for each parent based on the above reactions (if they are nonzero). If a daughter is not contained in the base activation library, it will not be carried along as a parent in the next generation.
4. Include production paths for hydrogen, deuterium, tritium, and helium as appropriate.
5. Set up valid parent-daughter-process relationships.
6. Eliminate multiple entries in the vector of daughters for the current generation.
7. Select only those daughters that have not been previous parents as the parents for the next generation.
8. Edit and save the parent-daughter-process information for use in ACTIV.

The final output from the code is a comprehensive nuclide list and the full set of parent-daughter-process triplets necessary for input to the ACTIV code - which does the actual activation calculation using the matrix exponential method..

ACHAIN also has the option to filter through the full parent-daughter-process chains and retain only the processes that will eventually lead, either directly or indirectly, to a selected set of activation products (as identified in the idap array). This option is very useful in practice, because it allows the use of only the chains that affect the isotopes of interest in a particular analysis.

Finally, it should be emphasized that ACHAIN was developed as a tool to assist in the development of appropriate transmutation data for use in ACTIV. The output libraries (full or filtered) are written in a straightforward fashion in ascii format, and they can be edited as needed for a particular application. The user can add, delete, or modify any of the entries as desired, or he or she can generate a complete chain library from scratch (without ACHAIN). As always, it is the user's responsibility to make sure that all important processes are included in the final library of chain data to be used in ACTIV. ACHAIN simply helps in this process.

ACTMAT Input Description

The execution and input instructions for ACTMAT were taken directly from the comment lines within the code. This practice guarantees that full consistency is maintained between the User Guide and the actual software. The FIDAS input processor is also utilized within ACTMAT as with the other codes of the ACTIV system.

```
c
c ****************************************************************
```

```
c
c   EXECUTION INSTRUCTIONS FOR ACTMAT
c
c   The code is executed in batch mode using a script file.  The following
c   files are used/created (with default file names):
c    input    - case dependent input data (ascii) (nin)
c    print    - case dependent output edit and debug print (ascii) (nprt)
c    chnlib   - input chain data library (possibly generated by ACHAIN)
c               (ascii) (nu1)
c    actlib   - input activation library generated by ACTXS (ascii) (nu2)
c    matlib   - output activation materials library (for ACTIV) (ascii) (nu3)
c
c
c   USER INPUT INSTRUCTIONS FOR ACTMAT
c
c FIDAS INPUT BLOCK #1  (primary dimension setting parameters)
c
c  1$$ Array (1 entry)
c
c   nmat = number of materials
c
c
c  END BLOCK #1  (terminate with a 't')
c
c
c  LOOP over following input arrays NMAT times
c   Basic Structure: --> title
c                    |   13$$, 14**    t
c                    --> 15$$, 16**  and 17$$, 18**    t
c
c
c Hollerith Title for Material j (a48 format)
c   hmtitl(j) -  material description
c
c
c FIDAS INPUT BLOCK #2
c
c  13$$ Array (5 entries)
c   matid(j)  -  material id (used in 9$$ array in ACTIV input)
c   nume1     -  number of elements in material with weight percent units
c   nume2     -  number of elements in material with ppm units
c
c   fill remaining entries with zeros
c
c  14** Array (5 entries)
c   densty(i) -  material mass density (g/cc)
c
c   fill array with zeros
c
c
c  END BLOCK #2  (terminate with a 't')
c
c
c FIDAS INPUT BLOCK #3
c
c  15$$ Array (nume1 entries)
c   id1(nume1)  - nuclide ids with weight percent units
c
c  16** Array (nume1 entries)
c   den1(nume1) - actual weight percent of isotopes in material j
```

```
c
c  17$$ Array (nume2 entries)
c   id2(nume2)  - nuclide ids with ppm units
c
c  18** Array (nume2 entries)
c   den2(nume2) - actual ppm of isotopes in material j
c
c    where  id1 or id2 = z*10000 + a*10 + is
c    and    z = atomic number,   a = atomic weight,
c           is = 0/1 ground/metastable state
c
c   Note: If the id number ends in 0000 (i.e. both a = 0 and is = 0), the
c         original nuclide represents a naturally occurring element with
c         multiple stable isotopes.  In this case the natural abundance
c         information in the activation library will be used to expand the
c         element into all its stable components.
c
c
c  END BLOCK #3  (terminate with a 't')
c
c
c ***************************************************************************
c
```

ACTMAT Input Notes

The ACTMAT program reads input nuclide vector information and appropriate initial density data for nmat materials and creates an activation material library (matlib) for use in ACTIV. Nuclide information from an appropriate activation chain library (chnlib) is needed to construct the final material data for inclusion in the material library. Input densities for the materials are in g/cc and the individual components are entered as weight percent (wo) or in parts per million (ppm) (as indicated by the 15$$ and 16** arrays or the 17$$ and 18** arrays, respectively). The output nuclide vector for use in ACTIV has units of atoms/b-cm. The output nuclide vector in the matlib dataset is in the same order as that given in the input chnlib database.

The input data and computations for ACTMAT are quite straightforward. For material j, given the physical density, ρ_j, and the weight percent of an individual isotope i in the material, w_{ij}, the desired atom density can be written as

$$N_{ij} = \rho_j \times \frac{w_{ij}}{100} \times \frac{.60225}{MW_i}$$

where

N_{ij} = atom density (atoms/b-cm) of isotope i in material j

MW_i = molecular weight of isotope i (g/gmole)

For naturally occurring elements (nuclide id ends in 0000), one inputs the weight percent of element k in the material, w_{kj}, and the desired atom densities for each of the isotopes within the element can be written as

$$N_{ij} = \rho_j \times \frac{w_{kj}}{100} \times \frac{a_i}{100} \times \frac{.60225}{MW_k}$$

where a_i = atom percent abundance of isotope i in natural element k

MW_k = molecular weight of element k (g/gmole)

If parts per million (ppm) is given instead of weight percent (w/o), one simply replaces ($w_{kj}/100$) with ($ppm_{kj}/10^6$) in the above equation.

In ACTMAT, the molecular weights for individual isotopes, MW_i, are determined simply by extracting the mass number (approximate molecular weight) from the nuclide identification. For naturally occurring elements, the molecular weight for element k is computed as

$$MW_k = \sum_{i \in k} \frac{a_i}{100} \times MW_i$$

where the atom percent abundance information is obtained from the base activation library.

The material data computed here are then stored in the matlib dataset for use in ACTIV. The material ID specified in the 13$$ array is used in ACTIV to identify the activation material to be used in each material zone of the model (see the 9$$ array in ACTIV). This material ID can be any unique integer identification tag. The activation material library can contain any number of activation materials. Only the specific subset given in the ACTIV 9$$ array is used in a particular problem.

ACTIV Input Description

The following execution and input instructions for ACTIV are taken directly from the comment lines within the code. The comments within ACTIV reflect the very latest information concerning the current version of the code. By echoing this information here, we guarantee that full consistency is maintained between the User Guide and the actual software. The FIDAS input processor is utilized (as common to DORT, SCALE4.3, etc.).

```
c
c ***************************************************************
c
c
c   EXECUTION INSTRUCTIONS FOR ACTIV
c
c   The code is executed in batch mode using a script file.  The following
c   files are used/created (with default file names):
c    input    - case dependent input data (ascii) (nin)
c    print    - case dependent output edit (ascii) (nprt)
c    actlib   - multigroup activation library (ascii file generated by the
c               ACTXS code) (nu1)
c    chnlib   - parent-daughter-process information (ascii file generated by
c               ACHAIN) (nu2)
c    matlib   - initial densities by material (ascii file generated by the
c               ACTMAT code) (nu3)
c    flxin    - pointwise multigroup flux (binary if from DORT or DOTSYN and
c               ascii if from ANISN or ascii scalar flux) (nu4)
c
c    flxout   - binary scalar flux file generated if iflx.ne.0 (nu8)
c
c    rstrlib1 - input final-time densities to be used as a restart file
c               (ascii file) (nu11)
c    rstrlib2 - output final-time densities for subsequent restarts (ascii
c               file) (nu12)
c    debug    - debug print (ascii) (ibug)  [only used if internal debug
c               switch is on]
```

```
c
c    also uses direct access file for storage of cross sections on unit nu10
c
c
c   USER INPUT INSTRUCTIONS FOR ACTIV
c
c
c FIDAS INPUT BLOCK #1 (primary dimension setting parameters and edit options)
c
c  1$$ Array (20 entries)
c
c1  im     = number of mesh in x-direction
c   jm     = number of mesh in y-direction
c   igm    = total number of energy groups
c   izm    = number of zones in model
c   ingeom = geometry option (0/1/2/3/4  x/r/xz/rz/rtheta)
c
c6  iflx   = format of input flux file (-2/-1/0/1/2 ascii scalar flux/ANISN
c           RTFLUX/binary scalar flux/DORT VARFLM/DOTSYN)
c   nacti  = number of i pointwise activation analyses to perform
c   nactj  = number of j pointwise activation analyses to perform
c   nactz  = number of zone-average activation analyses to perform
c   ineut  = number of neutron energy groups (default ineut = igm)
c
c11 nispe  = number of isotopes for special density edit
c   nxspe  = number of cross sections for special average cross section edit
c   nbgrp  = number of broad groups for calculation of average cross sections
c   nts    = number of time steps for activation calculation
c   iequ   = 0/1 - apply equilibrium option/do not apply equilibrium option
c
c16 iau    = sets units for activity edits (0/1 - activities in Bq/g / Ci/g)
c   irstr  = 0/1 - use restart density file for initial densities (no/yes)
c            (the matlib library from ACTMAT is always required
c   isoref = nuclide id to be used as reference for scaling factor edits
c
c
c   fill remainder of array with zeros
c
c
c  3$$ Array (10 entries)
c
c  These edit switches should be 0/1 for no edit/edit as desired:
c
c1  iemap  = edit zone and material maps
c  iebinfo = edit lots of basic information about special edit nuclides
c            (nuclide ids, decay data, parent-daughter-process information,
c            and description of activation materials)
c   ieact  = full space-time activity edit
c   ieden  = full space-time density edit
c   ieave  = edit approximate zone average activity versus time (determined
c            from 1-d profiles)
c
c
c   fill remainder of array with zeros
c
c
c  5** Array (5 entries)
c
c1  xmult  = normalization factor for all fluxes (default = 1.0)
c   efac   = factor for use in applying the equilibrium option
```

```
c    note: minimum value is 2.0 and default value is 10.0 (larger values will
c          reduce the number of nuclides treated with the equilibrium option)
c
c
c   fill remainder of array with zeros
c
c
c  END BLOCK #1  (terminate with a 't')
c
c
c FIDAS INPUT BLOCK #2  (primary input data for problem)
c
c  2** Array (jm+1 entries)
c   ymb(jm+1) - second dimension mesh boundaries (2 entries required for 1D
c               geometry -> 0.0 1.0)
c
c  4** Array (im+1 entries)
c   xmb(im+1) - first dimension mesh boundaries
c
c  8$$ Array (im*jm entries)
c   izmesh(im,jm) - zone by mesh information
c    note: this is usually the 8$$ array from the DORT/ANISN input -- however
c          the user can redefine the zone by mesh array to give any desired
c          zonewise analyses
c    note: the 2**, 4**, and 8$$ arrays may also need to be redefined
c          (relative to the DORT input) when using fluxes from the CHGISET
c          or PASTEFL codes
c
c  9$$ Array (izm entries)
c   imatz(izm) - activation material by zone
c    note: in general this is NOT the same 9$$ array as in the DORT input --
c          here the materials refer to activation materials, not the materials
c          used to do the neutronics calculations.
c          the material composition (including impurities) to be used in the
c          activation analyses are defined in the material library generated
c          by ACTMAT.
c          also note that a material id = 0 in the 9$$ array indicates that
c          the zone and associated mesh points are not to be activated (no
c          structural materials to be activated are present).
c
c  10$$ Array (nacti entries)
c   jloc(nacti) - j location of first dimension pointwise activations
c
c  11$$ Array (nactj entries)
c   iloc(nactj) - i location of second dimension pointwise activations
c
c  12$$ Array (nactz entries)
c   nzloc(nactz) - zone numbers for zone average activations
c
c  13$$ Array (nispe entries)
c   idispe(nispe) - ids of isotopes for special density edit
c    note: normal density edit in the code includes only those isotopes in the
c          above array -- all isotopes in the full nuclide vector are
c          included in the data files used for subsequent analysis
c
c  14$$ Array (nxspe entries)
c   idxspe(nxspe) - ids of isotopes for special average cross section edit
c
c  15$$ Array (nxspe entries)
c   itxspe(nxspe) - type of data for special average cross section edit
```

```
c    note: the types range from 1 to 8 (nxs) as follows:
c          1 - n,gam    2 - n,alpha   3 - n,p   4 - n,2n
c          5 - n,d      6 - n,t       7 - n,f   8 - n,abs
c    note: the 14$$ and 15$$ arrays have a 1-to-1 correspondence -- a total
c          of nxspe average cross sections will be computed and edited
c
c  16$$ Array (ineut entries)
c   idbtf(ineut) - broad group by fine group indexing for average cross
c          section edit (for nbgrp > 0)
c
c
c  END BLOCK #2  (terminate with a 't')
c
c
c FIDAS INPUT BLOCK #3
c
c  21** Array (nts entries)
c   pow(nts) - power normalization for each time step (this is the factor
c          which multiplies the absolute flux for a given time step)
c    note: if pow = 0, then a shutdown time step is assumed
c
c  22** Array (nts entries)
c   delta(nts) - time increment (in days) for each time step
c    note: a combination of the number of time steps and the time increment
c          per step allows full flexibility over the depletion/activation
c          calculation and the data available for subsequent edit purposes
c
c  23$$ Array (nts entries)
c   itespe(nts) - time dependent edit switch
c     where  0 - no edit at this time point
c           >0 - edit space dependent activities at this time point
c              1 - fractional contribution by isotope plus total activity
c              2 - also include absolute activities by isotope plus total
c                  activity
c              3 - also include scaling factors by isotope relative
c                  to the nuclide associated with isoref (see 1$$ array)
c    note: this switch gives detailed space dependent activity information for
c          the isotopes listed in the 13$$ array for the time points with
c          non-zero entries.
c    note: if full detail is needed in time and space, then ieact (see 3$$
c          array) should be turned on.
c
c
c  END BLOCK #3  (terminate with a 't')
c
c
c ***************************************************************
c
```

ACTIV Input Notes

The ACTIV code is the main computational module that performs the actual activation calculations. In addition to the standard input file containing case-specific information, a number of other datasets are also required as input to ACTIV for a particular computation:

1. A scalar flux file from DORT or other suitable transport code must be input to ACTIV. This file contains the space-energy neutron flux information for the particular geometry of interest.
2. A library of multigroup cross sections with the same group structure as used in the neutron flux computation is also required. This library must also contain all the pertinent decay and branching ratio information for each isotope. This dataset, which is denoted as **actlib** within ACTIV, is generated via the ACTXS code. In the current system, the ACTXS47.LIB library is used (compatible with the 47 neutron group BUGLE96 shielding library).
3. A dataset containing nuclide transformation information for the problem of interest is also needed. This library (called **chnlib** within ACTIV) contains the IDs of the isotopes to be included in the nuclide vector and it has complete parent-daughter-process information for the activation chains to be included as production paths within the transmutation matrix.
4. The initial density vector and other information for each activation material is supplied in the **matlib** library.
5. Finally, if a restart case is specified, a **rstrlib1** file containing the space dependent final-time isotope densities from a previous ACTIV calculation must be specified. The material and geometry description for a restart case must be fully compatible with the run that generated the restart file.

The first part of ACTIV reads the user input file and all the input data libraries and prepares the code for the specific computations that have been requested. In particular, the current version assumes a 1-D or 2-D geometry and it can perform activation calculations for any number of first dimension profiles (i-profiles), second dimension profiles (j-profiles), or zone average activations (z-profiles).

The unique aspect of ACTIV is the full space-energy coupling that is built into the computational algorithm. In activation analyses, one is interested in the interaction between the nuclide density field and the neutron flux field within the system over relatively long periods of time (usually alternating periods of power operation and shutdown followed by some period of decay after the end of life of the plant). The nuclide field obeys the nuclide transmutation equation, while the flux field in the excore region is determined by solving the neutron transport equation. In general, the coupling between the nuclide and neutron fields is nonlinear, but in the excore regions this nonlinear interaction is very weak and all practical methods for solving the nonlinear transmutation equations assume that the flux is separable in time, which gives rise to a transmutation equation of the form

$$\frac{d}{dt}\underline{N}(\vec{r},t) = \underline{\underline{M}}(\vec{r})\underline{N}(\vec{r},t) \qquad (1)$$

$\underline{N}$ is referred to as the nuclide vector and $\underline{\underline{M}}$ is the transmutation matrix or transmutation operator. In the linearized "quasi-static" approximation, the transmutation matrix, which consists of microscopic reaction rates and decay constants, is time independent. With this assumption, eqn. (1) simply represents a system of constant coefficient first-order differential equations for each spatial location of interest. Note that since there is no space-space coupling, eqn. (1) can be solved for only the spatial points of interest independent of information at other points in the system.

Constant coefficient linear systems have solutions that are exponential in behavior. Generalizing to the matrix problem of interest here, the solution to eqn. (1) becomes (dropping the spatial dependence for convenience),

$$\underline{N}(t) = e^{\underline{\underline{M}}(t-t_0)}\underline{N}(t_0) \tag{2}$$

where $\underline{N}(t_0)$ represents the initial isotope concentration information. Equation (2) can be written in a more convenient form as

$$\underline{N}(t+\Delta t) = e^{\underline{\underline{M}}\Delta t}\underline{N}(t) \tag{3}$$

or, using a discrete time index, we have

$$\underline{N}_{k+1} = e^{\underline{\underline{M}}\Delta t}\underline{N}_k \tag{4}$$

This form is particularly useful since $e^{\underline{\underline{M}}\Delta t}$ is simply a constant matrix for some constant Δt. Letting $\underline{\underline{G}} = e^{\underline{\underline{M}}\Delta t}$, eqn. (4) becomes

$$\underline{N}_{k+1} = \underline{\underline{G}}\underline{N}_k \tag{5}$$

Thus, the evaluation of the nuclide field versus time requires only repeated matrix-vector multiplication.

The $\underline{\underline{G}}$ matrix is referred to as the state transition matrix or simply the matrix exponential. In ACTIV, this matrix is evaluated using a truncated Taylor series expansion,

$$\underline{\underline{G}} = e^{\underline{\underline{M}}\Delta t} = \underline{\underline{I}} + \underline{\underline{M}}\Delta t + \frac{1}{2!}\left(\underline{\underline{M}}\Delta t\right)^2 + \cdots \tag{6}$$

where Δt is computed internally so that eqn. (6) converges within a reasonable number of terms in the expansion. Once $\underline{\underline{G}}$ has been determined, successive matrix multiplications are performed using eqn. (5) to obtain the nuclide density vector at all desired times. Equations (5) and (6) are referred to as the matrix exponential solution to the original transmutation equation given by eqn. (1) (at each spatial point of interest).

The unique space-energy coupling in ACTIV is built into the description of the transmutation matrix, $\underline{\underline{M}}$. This matrix contains information about all the production and loss mechanisms associated with a variety of possible nuclide transmutation processes (see the table of possible reactions listed on page 6 of this document). Equation (1) written for the i^{th} isotope is given by

$$\frac{d}{dt}N_i = \sum_j \left(\langle\sigma_x\phi\rangle + \lambda\right)_{ij} N_j - \left(\langle\sigma_a\phi\rangle + \lambda\right)_i N_i \tag{7}$$

where

$\langle\sigma_a\phi\rangle_i$ = microscopic absorption rate in isotope i

λ_i = total decay constant for isotope i

$\langle\sigma_x\phi\rangle_i$ = microscopic production rate via reaction x that transmutes isotope j into i

λ_{ij} = decay constant (with the branching fraction) for transmutation of isotope j into i

and the sum over index j accounts for the fact that multiple production paths to isotope i are possible. Thus, from this representation, we see that the diagonal elements of $\underline{\underline{M}}$ containing the loss terms are given by

$$m_{ii} = -\left(\langle \sigma_a \phi \rangle + \lambda\right)_i \tag{8}$$

and the off-diagonal elements containing the production terms are simply of the form

$$m_{ij} = \left(\langle \sigma_x \phi \rangle + \lambda\right)_{ij} \tag{9}$$

Now, in general, the microscopic reaction rate, $\langle \sigma_x \phi \rangle$, represents an integral over energy. With a multigroup formulation, the energy integral is represented as a discrete summation,

$$\langle \sigma_x \phi \rangle = \sum_g \sigma_{xg} \phi_g \tag{10}$$

Thus, eqns. (8) and (9), with $\langle \sigma_x \phi \rangle$ replaced by eqn. (10), represent the formulation for determining the elements of the transmutation matrix within ACTIV.

A key point here is that the multigroup activation cross sections, σ_{xg}, from the activation library, and the multigroup flux, ϕ_g, from the transport calculation have the same group structure so that eqn. (10) represents a formal integration over energy. Also, since the flux spectrum is different at every spatial point in the system [i.e. $\phi_g \rightarrow \phi_g(\vec{r})$], eqn. (10) represents a spatially dependent reaction rate that has been properly integrated with the multigroup flux spectrum appropriate for the specific point of interest. If the neutron spectrum varies significantly from one mesh point to another (as it often does – see the JPDR results in Appendix IV, for example), then the space-energy coupling used here becomes essential for an accurate computation of the induced activities. The proper treatment of this full space-energy coupling was he primary motivation for the design of the ACTIV code.

With the above discussion to identify the equations and terminology used within ACTIV, much of the ACTIV input becomes rather straightforward. Much of the input is geared towards defining the geometry and material distributions (2**, 4**, 8$$, and 9$$ arrays), the locations where the activation analyses are to be performed (10$$, 11$$, and 12$$ arrays), the various edit options (3$$, 13$$, 14$$, 15$$, 16$$, and 23$$ arrays), and the time sequence for the various intervals of power operation and shutdown (21** and 22** arrays). In fact, there are only three items that require special mention, as follows:

Zone Average Activation

The unique aspect of ACTIV is its full space-energy coupling. The i-profile and j-profile activation sequences treat this coupling rigorously following the equations given above. However, for full coverage of a large system, one could easily require thousands of spatial mesh, making a totally rigorous analysis very time consuming. To address this concern, a zone-average activation analysis capability has been implemented into ACTIV. This option does the

same calculations as indicated above, except a zone-averaged multigroup flux is used instead of the pointwise fluxes. In particular, $\phi_g(\vec{r})$ in eqn. (10) is replaced by ϕ_{gz} where

$$\phi_{gz} = \int \phi_g(\vec{r})d\vec{r} \Big/ \int d\vec{r} \tag{11}$$

and the integral is performed over the desired zone volume. If a discrete ij notation is used to represent the spatial mesh, we have

$$\phi_{gz} = \frac{1}{v_z} \sum_{ij \in z} \phi_{ijg} v_{ij} \tag{12}$$

where v_{ij} is the mesh volume and v_z is simply the sum of all the mesh volumes comprising zone z.

Although the use of the zone-averaged flux is not formally rigorous, it represents a good approximation since the multigroup activation cross sections are weighted with the appropriately averaged flux spectrum for that zone. However, as for any averaging process, the smaller the zone volume the better the overall approximation. This is especially important if the spectrum varies considerably over the zone volume. In this case, one can simply modify the 8$$ array to define smaller regions for the zone average activations.

Equilibrium Option

As discussed in the literature (see documentation for the VENTURE system, for example), if an entry in the $\underline{\underline{M}}\Delta t$ matrix exceeds some value (a value of 12 is used in ACTIV), the results of the series expansion in eqn. (6) for the matrix exponential may not have adequate significance due to subtraction of numbers of nearly the same magnitude. The simplest way to alleviate this problem is to reduce the value of Δt, thereby reducing the elements of $\underline{\underline{M}}\Delta t$. This procedure is done automatically in ACTIV, where the ΔT value set in the 22** array is broken into nstep smaller values, Δt, (i.e. $\Delta t = \Delta T / \text{nstep}$) and the number of times that eqn. (5) is applied is increased accordingly. The maximum value of nstep is 1000 in ACTIV.

This procedure for achieving convergence and adequate numerical significance of the matrix exponential works nicely for many situations. However, for isotopes with large coefficients along the diagonal of the matrix, the process of breaking ΔT into smaller subintervals can lead to a large number of subintervals which, in turn, gives very lengthy calculations (this is why nstep has an upper limit). Thus, for these isotopes, an equilibrium approximation is made, as described below.

In situations where large loss coefficients are encountered (i.e. $m_{ii} > 12*\text{efac}$), it is reasonable to assume that the isotope in question will take on the end-of-timestep steady state value very rapidly. Considering isotope i in the chain $j \rightarrow i \rightarrow k$, we simply eliminate the $i \rightarrow k$ coupling and replace it with an approximate coupling $j \rightarrow k$, with coupling coefficient

$$m_{kj} = \frac{m_{ij} m_{ki}}{m_{ii}} \tag{13}$$

We then drop nuclide i from the calculation and assume that isotope k is produced directly from nuclide j with production rate $m_{kj}N_j$. Additionally, the end-of-timestep density for equilibrium isotope i is determined from the steady-state assumption, giving

$$N_i(t_f) = \frac{m_{ij}}{m_{ii}} N_j(t_f) \tag{14}$$

This equilibrium treatment, although very simple, seems to be adequate in most straightforward situations. However, there has not been a lot of rigorous testing to date for the general case. This option is set as default in ACTIV, but it can be turned off completely if desired (see the iequ option in the 1$$ array). Also, the value of efac can be increased to reduce the number of isotopes that are identified as requiring the equilibrium option. The default value for efac is 10, but again, there has not been any real effort to optimize this value. Simply reducing the ΔT specified in the 22** array also minimizes the use and subsequent effects of the equilibrium option.

Nuclide-Specific Edits

Finally, a special note concerning the output edit may be in order. ACTIV does all its computations with the full nuclide vector defined in the chain library, and it always uses the full multigroup fluxes and activation cross sections. However, to keep the output edit reasonably manageable, only the broad group fluxes and selected broad group cross section edits are available (defined via the 14$$, 15$$, and 16$$ arrays). Additionally, the only nuclide-specific edits (densities, activities, scaling factors, etc.) that are allowed are defined by the isotope IDs given in the 13$$ array. The only exception here is for the total activity edit – this includes the total activity for all the isotopes in the full nuclide vector. The fractional contribution (FC) edit, however, is only associated with the special-edit isotopes identified in the 13$$ array and the sum of all the fractional contributions (sum FC) only adds the contribution from the special-edit nuclides. A value of sum FC much less than unity implies that not all the important activities at that time point are being edited.

All the remaining edits from ACTIV are self explanatory.

Some Sample Problem Sequences

Three sample problems are discussed briefly here and the inputs and outputs for these cases are distributed as part of the ACTIV code system (see listing of the VERS2.RME file containing a description of the contents of the release of Version 2 of the ACTIV code system). These problems demonstrate the use of the ACHAIN, ACTMAT, and ACTIV modules, highlighting the use of several different options within the codes. All the problems use a relatively simple 1-D computational model of the Maine Yankee reactor, with particular focus on the excore structures. The goal of the sample problems is to illustrate the basic operation of the codes, and the particular cases used do not represent a formal activation analysis for this reactor system. Other than the use of the 1-D Maine Yankee geometry for realism in the physical modeling, all the other code inputs were chosen with a focus on demonstration of the code option, and not on actual analysis for the Maine Yankee facility. Thus, the numerical values are not very meaningful and one should only focus on the mechanics of code operation.

The three sample problems roughly imitate the simulation of a typical small, medium, and large problem within ACTIV. A summary description of each case follows (the reader should also refer to the internal documentation in the VERS2.RME readme file distributed with the codes):

SMPL0 Sequence

This problem represents the simplest of the three cases. It does the activity calculation for a single zone with only 8 mesh points (the thermal shield). The chain library, containing only 11 isotopes in the nuclide vector and 9 off-diagonal production reactions, was generated by hand (without the use of ACHAIN). A single activation material consisting of stainless steel with a cobalt impurity level of 800 ppm is described in the ACTMAT input. These data were input to ACTIV along with the scalar fluxes from the 1-D DORT calculation, and a 3000 day full power burn followed by a 1500 day shutdown interval was simulated. Since the overall problem is quite small, several of the edit options were also activated. With only two computation steps, hand calculations (and selected auxiliary calculations with the Matlab code) were used to verify the computations performed in ACTIV. Thus, this sample problem was used as primary validation for many of the calculational procedures and edit options within ACTIV.

SMPL1 Sequence

This sample sequence is somewhat more realistic of a typical computation with the ACTIV code system. The activation analysis models 20 full power years of operation (7300 days) followed by a 1200 day zero-power decay period (total simulation time was 8500 days). All the excore structural regions through the front part of the shield tank were activated (four different varieties of steels with no impurities). The base chain library was generated for the primary components of steel (no impurities) with ACHAIN. This library used the 'filtered' parent-daughter-process (p-d-p) relationships associated with the four specific activation products of interest for this case (^{54}Mn, ^{55}Fe, ^{60}Co, and ^{63}Ni). ACHAIN automatically generates a nuclide vector with 77 isotopes and 238 p-d-p triplets when the full set of isotopes associated with the primary constituents of steel are used, but with the filtering option turned on for the four desired activation products, only 95 production chains and 40 unique isotopes remain. All the remaining calculations in this sequence used the filtered chain library.

The actual ACTIV calculation for this sample problem was performed twice:

Case 1 - The ACT1 run is a simple 8500 day simulation as described above.

Case 2 - The combined ACT1A and ACT1B runs represent the same overall calculation, but now the ACTIV computation is broken into two parts - a 7300 day full power burn followed by a completely separate 1200 day shutdown calculation. The two separate runs were coupled via a restart file generated in the ACT1A case and used by the ACT1B run.

The results of the Case 1 and Case 2 sequences were identical, thus demonstrating the mechanics and proper implementation of restart capability within ACTIV. This capability should prove useful in practical application.

The primary focus of the SMPL1 sequence was the demonstration of the automatic chain generation capability in ACHAIN (including the filtering option) and the restart option in

ACTIV. Both features are essential in a production package for performing excore activation studies.

SMPL2 Sequence

The last sample problem illustrates how the addition of detailed impurity information for the activation materials complicates the overall analysis. In this case, only two different steels were modeled, but they included the full complement of impurities as specified in NUREG/CR-3474 (the impurity level included the average values plus one standard deviation). With all these base materials specified as input to ACHAIN, a full chain library with 338 nuclides and 909 possible production chains was identified. Again, with the use of the filter option (this time for 11 desired activation products), this full library was reduced considerably -- resulting in a nuclide vector with 229 elements and a total of 636 parent-daughter-process relationships for implementation with the matrix exponential solution scheme in ACTIV. The filtered p-d-p library was used in the remainder of this sample sequence.

The ACT2 ACTIV calculation is very similar to the ACT1 case except for the significantly larger vector of unknowns (229 in ACT2 versus 40 in ACT1). The code effectively solves 229 coupled first order differential equations at each activation point (mesh interval or zone) specified in the ACTIV input. Thus, as expected, this case also takes substantially longer to run than the other sample sequences. However, the SMPL2 sequence does indeed demonstrate that realistic problems (that include lots of impurities) can be modeled and executed with the current system.

Finally, it should be emphasized that the above sample problems were designed to illustrate and demonstrate the mechanics of various options and capabilities within the codes. The reader is referred to the discussion of the JPDR benchmark computation in Appendix IV for an illustration of the use of the ACTIV code system within the context of a real application.