ALARA: Analytic and Laplacian Adaptive Radioactivity Analysis

A Complete Package for Analysis of Induced Activation

Volume II
Users' Guide

Paul P.H. Wilson
Douglass L. Henderson

January 1998

UWFDM-1071

FUSION TECHNOLOGY INSTITUTE

UNIVERSITY OF WISCONSIN

MADISON WISCONSIN

ALARA: Analytic and Laplacian Adaptive Radioactivity Analysis

A Complete Package for Analysis of Induced Activation

Volume II
Users' Guide

Paul P.H. Wilson
Douglass L. Henderson

Fusion Technology Institute
University of Wisconsin-Madison
1500 Engineering Dr.
Madison, WI 53706

January 1998

UWFDM-1071

Abstract

While many codes have been written to compute the induced activation and changes in composition caused by neutron irradiation, most of those which are still being updated are only slowly adding functionality and not improving the accuracy, speed and usability of their existing methods. ALARA moves forward in all four of these areas, with primary importance being placed on the accuracy and speed of solution.

By carefully analyzing the various ways to model the physical system, the methods to solve the mathematical problem and the interaction between these two issues, ALARA chooses an optimum combination to achieve high accuracy, fast computation, and enhanced versatility and ease of use.

The physical system is modeled using advanced linear chains, which include the contributions from straightened loops in the reaction scheme, while the truncation philosophy minimizes the discrepancies between the model and the real problem. The mathematical method is then adaptively chosen based on the characteristics of each linear chain to use analytically exact methods when possible and an accurate expansion technique otherwise.

Future modifications to ALARA include new functionality by implementing methods to use new data libraries, implementing methods to get new information from existing libraries, enhancing usability, and improving speed by fine tuning and parallel processing.

Contents

1. Introduction

When designing any system with a large neutron flux, an important characteristic is the amount of induced activation expected in the system's components during operation, at the end of life and at various times after the shutdown of the system. Many codes have been written to perform such calculations for a variety of systems, from accelerators to fission and fusion reactors. The special conditions of fusion reactors, such as high neutron flux/fluence and pulsed operation, have led to many variations of these codes.

The calculation of induced radioactivity in the first wall, blanket and shield materials is an important task for the design and safety of fusion reactors. The neutron products of the D-T reaction induce radioactivity by interacting with and transporting through these materials with much higher initial energies and populations than those of fission reactors of similar power. The results of these radioactivity calculations are used extensively in safety and design analyses to determine such parameters as the nature of the radioactive waste, the amount of shielding required for radiologically sensitive components, and the decay heating after shutdown. Like other engineering calculations, the accuracy of the results is important; overly conservative approximations result in costly and complicated designs while liberal approximations result in safety and technical hazards to the operators, scientists, public and equipment.

To solve this problem a code must perform two steps. First it must model the physical system in time, space and isotopic composition, creating a system of linear first order ordinary differential equations [ODE's]. Second, the solution to this system of ODE's must be found using a numerical technique. Both steps are non-trivial since the physical problem, while finite in time and space, is theoretically infinite in final isotopic composition, and the resulting ODE's have characteristics which can make their efficient and accurate solution difficult.

ALARA is a new computational tool for performing such calculations. Given a group-wise neutron flux, ALARA uses data from a variety of libraries to determine the altered material composition which is then used to calculate the activity, and β-, γ-, and α-heating. In addition, a group-wise γ-ray source flux can be computed by ALARA to be used for the calculation of doses. Finally, if provided with an adjoint importance field based on flux-to-dose conversion factors and the gamma source distributions, ALARA can directly calculate the biological dose.

1.1. Historical Attempts and Failings

The computational solutions to this problem have been well studied. Many different approaches for modeling the physical problem have been combined with at least as many mathematical solution methodologies. Each combination has its advantages and disadvantages,[1] but none have arrived at an optimum mixture of accuracy, efficiency and usability. Even ignoring the issue of usability, there are few codes which are keeping up with the demands of greater accuracy in modeling and solutions without becoming inconveniently slow.

One of the best performers in the past, in terms of speed and accuracy, has been the DKR[2–4] family of codes.[a] Unfortunately, even though it reaches the ultimate in mathematical accuracy and efficient operation, its physical modeling has left it subject to much philosophical criticism. Namely, DKR is unable to model loops in the reaction scheme. If an isotope undergoes a series of transmutations and decays which lead back to itself, DKR ignores any such contribution in all but the simplest of cases. While it has been shown that this is a somewhat valid criticism,[5] those codes which have addressed this problem in the past have many other failings. Another criticism of DKR is its inability to track and log the production of light ions, often important when analyzing the mechanical integrity of a material. On the other hand, DKR has pioneered the ability to exactly model pulsed irradiation histories and use mathematically exact solution methods while solving a multi-dimensional input problem.

Two of the most popular alternatives to DKR are FISPACT[6] and RACC.[7,8] While FISPACT is heavily used in Europe, it has a number of disadvantages. First and foremost, it is unable to accurately and exactly model the pulses which are today part of the designs of so many fusion reactor systems. This has been shown to be an important issue in the calculation of activity for some isotopes, leading to errors of up to several orders of magnitude.[9] Furthermore, it uses an ODE solver which is step-wise in time and, given the stiffness of the system, requires a slow and tedious calculation. Finally, as a 0-dimensional code, it is only able to find a solution for one given spectral distribution with each operation. While RACC has historically had the same problems with pulse modeling and mathematical method, the newest version, RACC-P, has addressed these issues and now models the pulsing

[a]The DKR family of codes has evolved much since the original authoring of DKR to the most recent version known as DKR-Pulsar. Throughout this report, DKR will refer generally to the entire family and specifically to the most recent version.

exactly and uses a matrix solution method to increase the speed. It does, however, have its own drawbacks. In certain regimes, the solution method for each matrix is subject to significant errors which can then be amplified as this matrix is used to repeatedly calculate the final answer. The data handling methods of RACC are its biggest obstacle to efficient and accurate operation. First, it employs a philosophy to truncate the reaction schemes which leads to inconsistent precision in the solution. It also recreates the reaction schemes for each point in space with a different flux spectrum, a very time consuming process which must be accelerated by solving the problem with a flux which is averaged over a number of spatial points.

While ALARA is an entirely new code product, the methods and philosophies embodied in DKR were chosen as a starting point for its development. The basic philosophies of exact modeling of pulsing, consistent truncation of reaction schemes, and mathematically exact solution methods were retained and the main criticisms addressed. The design philosophy is outlined later in this chapter. A short summary of the features is given in Chapter 2. Chapter 3 describes the basic usage of ALARA including command line options, required libraries and support files. A detailed description of the input file format is included in Chapter 4 and Chapter 5 describes the format of the output.

1.2. Design Philosophy

ALARA has been designed with three basic principles in mind: accuracy, speed, and simplicity. These three qualities have been maximized in ALARA after extensive research of the models involved in such calculations.[1,5] The errors, time of execution, and learning curve have all been made "as low as reasonably achievable".[b] The methods used to model the physical system and to perform the mathematical solution are carefully combined to preserve or enhance the accuracy while accelerating the solution. Throughout all this, there is an underlying effort to ensure that ALARA will be easy to use by providing a simple, well-documented input file format, checking this input for errors, and providing a broad, flexible range of options.

[b] This phrase is the origin of the term ALARA, a well known philosophy in the nuclear industry related to the minimization of radiation exposure when working in radioactive environments.

1.2.1. Accuracy

The accuracy of the final solution is affected both by how realistically the physical system is modeled and by what mathematical methods are employed for the final solution. Unfortunately, these two requirements often conflict; as the physical model becomes more realistic the required mathematical methods become more approximate or error prone. When modeling the physical problem, two of the most important issues are how to deal with loops in the reaction scheme and how to truncate the theoretically infinite isotopic composition to a finite problem. While the effect of the latter on the mathematical method is negligible, the former has a great impact. In the past, the unwritten rule has been that realistic treatment of loops requires complicated/inefficient mathematical methods. ALARA has broken that rule by finding a physical approximation to the loops which retains problem accuracy and allows for quite simple and efficient mathematical methods. The keys to ALARA's mathematical accuracy are the ability to adaptively choose the mathematical technique and the accuracy of those techniques. Two of the three mathematical techniques which ALARA employs are mathematically exact!

1.2.2. Speed

The most significant factor affecting the speed is the chosen class of mathematical method. In particular, unless a linear transformation matrix method is used the time required to exactly model a pulsed history will be large. ALARA employs such matrix methods, solving for the linear transformation from the initial isotopic composition to the final composition for each pulse and inter-pulse dwell period, and then multiplying these matrices to obtain a complete linear transformation for the entire history. In addition to this decision, speed was considered throughout the code design process. For example, data library formats and internal data handling have been implemented with modern techniques to enhance versatility without sacrificing speed.

1.2.3. Simplicity

While accuracy and speed have long been issues in the creation of engineering codes, their simplicity is of increasing importance. In this context, simplicity is an issue for both modification/maintenance and use of the code. Since ALARA has been written in C++, it benefits from some of the philosophies of object-oriented code design. This allows the code

itself to be more readable to future programmers and also facilitates enhanced modularity. This modularity means that if new functionality is added to the code, it can be optimized internally with minimal detrimental effect on the existing code.

ALARA has also been designed with the user in mind. Even though improved methods have existed for years, many codes have continued to use input formats which are reminiscent of punch card input entry. Furthermore, most tools in this field have been designed for the solution at a single spatial point, requiring many subsequent and slightly altered runs to get any kind of spatial information. ALARA allows the user to find the solution to an activation problem in a variety of different multi-dimensional geometries, using a flexible system to define the material properties and allowing a complicated pulsed/intermittent irradiation history and a variety of after-shutdown solution times. Furthermore, the input file can be fully commented, preventing the common difficulty of creating a long list of seemingly disconnected numbers for code input.

Finally, the data used by ALARA can come from one of a variety of sources. To accommodate this, a companion code, ALARA Data Conversion [ALARA_DC], has been written and is described in Appendix A.

2. ALARA Features

ALARA has a number of features which distinguish it from other activation codes.

Geometry Definition

- 3-dimensional activation calculations
- point, slab, cylindrical, spherical and toroidal geometries

Mixture Definition

- predefined material and element libraries
- arbitrary combinations of predefined materials, user-defined materials, elements and isotopes

Interval (Fine Mesh) Definition

- interval volumes defined by user OR calculated by code
- grouping of intervals within zones defined by user OR calculated by code

Truncation

- user defined calculation precision

Flux History

- multiple sequential pulsing schedules
- within each schedule
  - independent pulse width
  - independent scalar flux
  - multiple nested pulsing levels

Internal Features

- adaptive solution method based on individual chain characteristics
- calculation of "looped" chains to user defined precision

Output Options

- spatial solutions can be grouped by mixture definition, zone (coarse mesh point), and/or interval (fine mesh point)
- results given at the end of each pulsing schedule and each after-shutdown time
- various types of results are available including:
  - number density
  - volumetric activity
  - total decay heat

  - partial decay heat from alpha, beta and/or gamma
  - gamma source at each interval with user-defined group structure
  - dose calculation if gamma importance field is provided for folding with gamma source
- full output of decay chains including:
  - relative production used in chain creation/truncation decision
  - reaction type information
  - reason for truncation

3. ALARA Usage

The usage of ALARA is fairly straightforward, requiring little knowledge of the inner workings of the code. Of course, to ensure that ALARA is well-suited to the problems that you are trying to solve, you are encouraged to read the technical manual[10] and understand the physical and mathematical modeling characteristics of ALARA.

This chapter will describe the command-line options of ALARA and then describe the basic support files which are necessary to run ALARA.

3.1. Command-line Options

`alara [-s`***tree_output_filename***`] [-h] [-v`***[n]***`]` ***input_filename***

ALARA currently supports 4 command-line options:

`-h` Help

This option will print a short help message describing the command-line options and their usage.

`-s`*tree_output_filename* Tree File

This option allows you to define the name of the output file for the tree creation and truncation information. This information can later be used for basic pathway analysis. The default is to create no tree file. The format of this file is described in Section 5.2.

`-v`*[n]* Verbose

This option alters the verbosity of the output. Without this option, only the final results will be displayed. By using this option, some of the details of the calculation are included in the output. The level of detail is controlled by the optional value, *n*, which has a value between 1 (least detail) and 5 (most detail). If no value is given, it defaults to 1.

***input_filename* Input File**

This option allows you to define the name of the input file to be used by ALARA. If no name is specified, the default of 'input.file' will be used. If the name '-' is given, the input will be read from stdin.

3.2. Support Files

To run ALARA successfully the following files must be in the working directory. They can, however, be links to an original version which exists elsewhere in the file system.

Fixed Names

mergebin Binary Data Library

This file must exist and must contain the binary nuclear data file for transmutation and decay. Despite the name 'mergebin' this file may be either the 'mixed' or 'merged' format as described in Appendices B.3 and B.4 respectively. Section 3.3 describes how these two library formats can be used.

gammabin Binary Gamma Source Library

This file must exist if gamma source and/or dose calculations are requested in the input file. The format of this file is described in Appendix B.5.

matlib Material Library

The format of this text library is described in Appendix C and its usage described in detail in Section 4.4. This file must only exist if material type components will be used in the mixture definitions.

elelib Element Library

The format of this text library is described in Appendix C and its usage described in detail in Section 4.4. This file must only exist if element type components will be used in the mixture definitions.

Variable Names

input file

The input file must exist in the working directory and, if the name 'input.file' is not used, it must be specified on the command-line as described in Section 3.1.

flux file

The flux file, as specified in the input file, must exist in the current working directory. For details on the format of the flux file, see Appendix D.

adjoint gamma flux file

If gamma dose calculations are to be performed, the gamma importance field, or adjoint gamma flux file, must exist in the currently working directory as specified in the input file. More details on its format are also given in Appendix D.

3.3. Comments on Nuclear Data Library Usage

There are two proprietary nuclear data library types which can be used by ALARA: 'mixed' and 'merged'. The details of the formats are described in Appendix B. In both cases, these libraries are created by a freely available companion program, ALARA_DC, described in more detail in Appendix A. Using ALARA_DC, these two formats can be generated from many popular data library formats.

Both 'mixed' and 'merged' libraries combine the transmutation cross-section data and the decay information. It is possible to combine the complete transmutation data and complete decay data from two different sources into one of these libraries.

The only difference between these libraries is that multiple reaction channels with the same product isotope are combined into a single reaction channel in the 'merged' type, while they are left distinct in the 'mixed' type. While the 'mixed' type may provide more detailed results for pathway analysis in the tree file, it will require longer runtime or may hinder the accurate truncation of the chains. The longer runtime is the result of two different subtrees being rooted in the same reaction product. That means that all the chains which make up the subtree will have to be calculated twice. The hindered truncation accuracy is because each reaction channel may independently produce too little of a product to warrant the continuation of the chain, while together they may produce enough to continue the chain.

For most calculations, the merged format is recommended. However, if a careful understanding of the reaction mechanism is an important part of your pathway analysis, the mixed format may be preferred.

4. Input File Description

The input file for ALARA has been designed to ensure that the input information is easy to understand, edit and comment. This is possible by using a relatively free format which allows comments and blank lines.

4.1. General

The input file for ALARA is divided into 10 major segments:

1. Title
2. Geometry
3. Mixtures
4. Zones
5. Intervals
6. Flux
7. Truncation
8. Operation History
9. After-Shutdown History
10. Output Definition

The segments must appear in this order. Each of the following sections describes the input for that segment and gives an example of commented input.

All lines in which the first non-space character is the pound sign (or number sign) (#) are considered as comments. Comments can also be used after any single word input (an input value which has no whitespace) by using the same comment character (#). Such comments extend to the end of the current line. Blank lines are permitted anywhere in the input file.

When length units are implied in the input for sizes and dimensions, it is only important that all implied units be consistent but not what unit is implied.

4.2. Title

A title for the problem must be given as the first input. This full line of input is used only in the output for the benefit of the user.

```
# Title of this analysis
Sample input file
```

4.3. Geometry

The required entries of the geometry segment are:

1. primary geometry type
2. specific geometry type

and must appear in that order. The possible combinations of values for these two entries are:

Table 4.1. Geometry Definitions for ALARA

Primary Geometry Types	Specific Geometry Types	Number of Dimensions
point	0	0
slab	x	1
	xy	2
	xyz	3
cylinder	r	1
	rtheta theta-r rz	2
	rtheta-z theta-rz	3
sphere	r	1
	rtheta theta-r rphi	2
	rtheta-phi theta-rphi	3
torus[1]	r phi[2]	1
	rphi	2
irregular[3]	pseudo-1D	(1)

1. If the primary geometry type is 'torus', the last entry in the geometry segment must be the major radius.

2. If the primary geometry type is 'torus' and the specific type is 'phi', the minor radius must precede the major radius entry (see 1).

3. The irregular geometry type is functionally identical to the 1-dimensional slab geometry type (slab - x). It is intended for use when the neutron transport results are from a Monte Carlo solution and the various intervals do not represent a regular geometry. When this geometry type is used, the interval definition flag (see Section 4.6) must be 'input'.

```
# Geometry type of system
# point | slab | cylinder | sphere | torus | irregular(MCNP)
sphere

# Geometry sub-type of system
# point: 0
# slab: x | xy | xyz
# cylinder: r | rtheta | theta-r | rz | rtheta-z | theta-rz
# sphere: r | rtheta | theta-r | rphi | rtheta-phi | theta-rphi
# torus: r | phi | rphi
# irregular: pseudo-1D (user MUST!! specify intervals explicitly)
rtheta

##### if torus option: phi
# torus minor radius
# 1

#### if any torus option:
# torus major radius
# 5
```

4.4. Mixtures

The term 'mixture' is used in ALARA to describe a user-defined combination of components which define the constituent isotopes of a geometrical region. A mixture is not directly attached to any geometrical region, but may be used to fill one or more geometrical regions.

The first entry is an integer defining the number of mixtures which are in the problem. This is followed by a set of entries for each mixture definition.

For each mixture, the first entry is an integer confirming which mixture this is (starting with mixture #1). This is followed by a full-line entry giving a user-defined name for the mixture. This is used only for the benefit of the user during output. The next entry is the number of components which make up this mixture.

Each component has an entry card which begins with and depends on the single character flag to define the type of component.

Material (m): a predefined material from the material database

This component entry consists of the component type flag, 'm', the material name as given in the material database, and its relative density. The relative density is a floating point number between 0 and 1 defining the fraction of the density given in the material database to be used for this component.

Element (e): a natural element

This component entry consists of the component type flag, 'e', the chemical symbol of the element, the density flag, the density, and the atomic (Z) number of the element. The density flag is either 'N' or 'D' depending on whether the given floating point density is a number density or a mass density. If the density is given as zero (0), the natural density from the element library is used.

Isotope (i): a specific isotope

This component entry consists of the component type flag, 'i', the chemical symbol of the element, the mass number (A), the density flag, the density, and the atomic number (Z) of the element. The mass number is the integer mass number. The density flag is either 'N' or 'D' depending on whether the given floating point density is a number density or a mass density.

Like Other (l): like a previously defined mixture

This component entry consists of the component type flag, 'l', the mixture number to which this component is similar, and the relative density. The similar mixture number is the same number given as the first entry for each mixture. The relative density is a floating point number between 0 and 1 defining the fraction of the density defined in the similar mixture.

```
#total number of mixtures
2

# mixture #1
#  mix#
   1
#  mixture name
borated carbon
#  number of components
   2
#  component definition
#    matflag      name|sym   [A]  [densFlag]   [No]      [Z]
#  m | e | i | l            mass#   D | N     density  atomic#
#-------------------------------------------------------------
       e           c                  N      0           12
       i           b         10       N      7.893e+22    5

# mixture #2
#  mix#
   2
#  mixture name
borated carbon mixed with water
#  number of components
   2
#  component definition
#    matflag      name|sym   [A]  [densFlag]   [No]      [Z]
#  m | e | i | l            mass#   D | N     density  atomic#
#-------------------------------------------------------------
       l           1                            0.5
       m           WATER                        0.5
```

4.5. Zones

The term 'zone' is used in ALARA to describe a geometrical region containing a contiguous set of one or more intervals, all of which have the same mixture. A synonym for 'zone' is 'coarse mesh point'.

The first entry of the zones segment is a list of the number of zones in each dimension used in the problem. If a dimension is not explicitly declared in the geometry definition, no

value is given here and ALARA will automatically use one infinite zone containing a single interval in this dimension.

This is followed by a list of entries, one for each zone, containing the ordinal zone number (starting with 1) and the mixture contained in that zone. The ordinal zone number is calculated by numbering the zones increasing first through the first dimension, then through the second, and then the third, as needed.

```
#The total number of coarse mesh zones in each dimension
#
#       x       y       z
        2       2

#The mixture that is contained in each zone
# zone# mixture#
1       1
2       2
3       2
4       1
```

4.6. Intervals

The term 'interval' is used in ALARA to describe a geometrical region within a zone for which a flux has been calculated in a previous transport calculation. A synonym for 'interval' is 'fine mesh point' and it is the the smallest geometrical region which ALARA uses.

The input data for intervals has two possible formats which are generally dependent on the data available to the user from the neutron transport calculation. The first entry of the intervals segment is a word which begins with either the letter 'c' or the letter 'i' and defines either of the following formats.

Code calculated 'c': interval data calculated by ALARA

In this format, the first entries are lists which define the boundaries of the zones in each required dimension. A float number is given for each of the N+1 boundaries of the N zones. This is followed by a set of lists which define the number of intervals per zone in each required dimension. A number is given for each of the N zones.

Input data 'i': interval data given as input

In this format, the first entry is an integer defining the number of intervals in the problem. This is followed by one entry for each interval which consists of the ordinal interval number (starting with 1), a floating point number defining the volume, and an integer defining in which zone this interval exists.

```
#calculate interval information flag
# c | i
# example of ‘c’ flag
# !!!!!! commented out in this example
  code
#
# The boundaries of the coarse mesh zones
# dimension 1
  0   0.5  1.0
#
# dimension 2
  0  0.7854 1.57079
#
# dimension 3
# none
#
# number of intervals in each zone in dimension 1
  3 4
#
# number of intervals in each zone in dimension 2
  2 2
#
# number of intervals in each zone in dimension 3
#



# c | i
# working example for ‘i’ flag
#  i
#The total number of fine mesh intervals
# 4

#The fine mesh interval volumes
#IF interval information flag = i
#int#   volume          zonenum (first zone is 1)
#1      0.5               1
#2      0.6               2
#3      1                 3
#4      1.5               4
```

4.7. Flux File and Format

The input data required for the flux file and format are a single character defining the format, a filename, a flux scaling factor, and a neutron wall load. ALARA currently supports a single format, 't', which is defined in Appendix D. The filename should include relative path information to find the file on your system from the directory in which ALARA will be run. The flux scaling factor is a floating point number used to uniformly scale all the fluxes in the problem. Using this factor is similar to scaling the scalar magnitude of the neutron source in the transport problem, but without having to rerun the transport problem. The neutron wall load is currently unused, but must be provided as a floating point number.

```
#flux format information
# flux format  filename       Flux Conv. Factor  n Wall loading
# t |?                        FCF                nWallLoad
  t            mfe.flux.file  1                  1
```

4.8. Truncation

The truncation segment consists of two floating point numbers which define the parameters for truncating the chains as they are created by ALARA. The first parameter is the truncation tolerance, and defines what fraction of the initial isotope in the chain must pass through a node in the chain for the chain to be continued at that node. The second number is the ignore tolerance, and defines what fraction of the initial isotope in the chain must pass through a node in the chain for that node to be included in the solution. If the production of a single node is below that ignore tolerance, the chain will be shortened to its parent, and the truncation process will continue. See Section 3.1 of the technical manual for more information on the chain creation procedure.

```
# decay scheme truncation information
# truncation tolerance, ignore tolerance!
 5e-4   5e-4
```

4.9. Operation History

This segment describes the irradiation history for the problem. The current version of ALARA supports multiple sequential pulsing schedules each with multiple levels of pulsing, a single pulse width, and a uniform flux scaling factor. The first parameter is an integer defining the number of sequential pulsing schedules. Following this is a series of input entries for each pulsing schedule.

The first entry for each pulsing schedule is an integer defining the number of pulsing levels, followed by a floating point number defining the pulse width in seconds and a floating point number defining the flux scaling factor. These are then followed by one entry for each pulsing level which consists of the following:

- a floating point number defining the dwell time of the level, in time units defined by the next field,
- a single character defining the time units of the dwell time given in the previous field, and
- the number of pulses in this level.

The possible time units are **s**econds, **m**inutes, **h**ours, **d**ays, **y**ears, and **c**enturies.

```
#flux pulse history
# number of sequential pulsing schedules
 2
# schedule #1
# # of levels of pulsing
  3
# operation time (s)
  2300
# flux scaling factor
  0.75
# level information
#   decay time  units                    # of pulses
#               s | m | h | d | y | c
    200         s                        11
    16          h                        5
    2           d                        520
# schedule #2
# # of levels of pulsing
  3
# operation time (s)
  2000
# flux scaling factor
  1.1
# level information
#   decay time  units                    # of pulses
#               s | m | h | d | y | c
    200         s                        11
    16          h                        5
    2           d                        520
```

4.10. After-Shutdown History

This segment allows the user to define the various after-shutdown times for which the output will be calculated. After a first integer value defining the number of different after-shutdown times, is an entry for each after-shutdown time. Each entry has an integer defining which after-shutdown time it is, followed by a floating point number for the after-shutdown time and single character for the units of this after-shutdown time. The units for the after-shutdown time are the same as for the operation times.

```
#after shutdown information
#  # of after-shutdown times
   6
#  after-shutdown time information
#  time#        time val        unit
#                               s | m | h | d | y | c
   1            10              m
   2            10              h
   3            10              d
   4            1               y
   5            10              y
   6            1               c
```

4.11. Output Definition

The first and only required part of the output definition are two character strings indicating the output resolution and output type, respectively. The output resolution defines the scale on which the results are combined and presented in the output. Three choices are available, and any combination of them can be used:

mixture (m):

Results are given for each mixture. As mentioned in Section 4.4, a mixture is combination of components (predefined materials, elements and isotopes) which is used to fill different zones. This output resolution combines the activation results of all the different zones which contain the same mixture, whether or not they are geometrically near each other, and reports this sum. This would be used, for example, to find the activation in all the water in an entire system.

zone (z):

Results are given for each zone. The results for all intervals (fine mesh points) within a single zone are summed and reported. This would be used, for example, to find the activation in a single component of a system, which may contain many fine mesh points.

interval (i):

This most detailed option gives individually tabulated results for every single fine mesh point.

The output type defines what kind of data is reported. The options are tabulated in the following table:

N	number density
A	specific activity [Bq/volume]
Q	total specific heating [W/volume]
a	specific alpha heating [W/volume]
b	specific beta heating [W/volume]
g	specific gamma heating [W/volume]
s	gamma source (see below)
d	dose calculation from adjoint gamma flux

For either of the last two output types, further entries in this segment are required. In particular, details are needed to define the gamma spectrum which will be used for the gamma source and/or the adjoint gamma flux. In both cases, an integer entry defining the number of groups in the gamma group structure, followed by that many floating point entries defining the group boundaries, in electron-Volts. The first of these group boundaries must be zero (0), and the last group is assumed to go from the last input boundary to infinity. If the adjoint gamma flux is to be used for a dose calculation, a filename must be given for the gamma flux importance spectrum which will be folded with the gamma source to calculate the dose.

Finally, there is the option of defining the width of the output in characters by a single integer entry.

```
#output specifications
#output resolution
# form a string out of the following characters:
#       m - mixture resolution
#       z = zone resolution
#       i = interval resolution
# this example asks for output for all three resolutions
  m

# output type
# form a string out of the following characters:
#       N - number density
#       A - activity
#       Q - total heating (assuming localized gamma deposition)
#       a - alpha heating
#       b - beta heating
#       g - gamma heating
#       s - gamma source
#       d - dose calculation from adjoint gamma flux
# this example asks for output for number density, activity,
# and total heating
  N
#AQabgs

# gamma group structure information for output
# number of gamma groups
10

# gamma group boundaries
# !! Always start with 0 !!
# last group will always range to infinite
0 1e-1 1e0 1e1 1e2 1e3 1e4 1e5 1e6 1e7

# if dose convolution is requested
# adjoint flux file name
 adj.flux.file

# adjoint flux format
# ? | ?
# ?

# Width of output in characters
80
```

5. Output File Formats

The output of ALARA is written to the standard output. To save this output in a file, simply redirect it to a file using the standard redirection method of your operating system. In addition, ALARA produces a tree file, as mentioned in Section 3.1.

5.1. Output Format

The output of results of an ALARA calculation is partially dependent on the input strings for both output resolution and output type (see Section 4.11). For each of the chosen output resolutions, a set of tables is given. In each set of tables, there is one table for each chosen output type. Within each table, there is one row for each resultant nuclide and one column for each end-of-schedule and after-shutdown (cooling) time.

5.2. Tree File Format

ALARA also produces a so-called tree file to allow some rudimentary pathway analysis (see Section 3.1). The tree file contains much information about the creation and truncation of the trees and chains used to calculate the transmutation and activation in the problem.

One tree will be created for each initial isotope. All the information given for this isotope is based on the flux chosen for the truncation calculations of this isotope, namely, the group-wise maximum flux across all the intervals in which the initial isotope exists. An entry for an isotope in the tree will look like this:

```
                -(na)->h-3  - (0.00306937 -> 0.00234645)
```

The level of indentation indicates the rank of this isotope in the tree. This can be best seen by viewing the whole file and noting the relative indentation of the line. The information given in such an entry is as follows:

reaction type: (na) This indicates the reaction type. If using a 'merged' library (see Section 3.3), and multiple reactions lead to this product, the reactions will be separated by commas. The information indicates the emitted particles only. Therefore, in this example, the reaction is an (n,na) reaction. Generally, standard symbols are used, such as 'n' for neutrons, 'a' for alpha particles, 'p','d','t' for the three isotopes of hydrogen,

respectively, and 'h' for helium-3. For all neutron reactions, an additional '*' is used to indicate that the product is in an excited isomeric state. Finally, for decay reactions the symbol 'D*' is used.

product nuclide: h-3 The chemical symbol and atomic number of the product isotope. In cases where the product is in an isomeric state, this will be followed by a letter (m,n,...) indicating which isomeric state.

truncation mode: - This single character indicates the result of the truncation calculation at this node. There are five possible results as follows (see Chapter 3 of the Technical Manual):

- **-** This code indicates that the chain continues normally because this isotope passed all the tests.
- ***** This code indicates that only the radioactive decays of the chain will be followed after this node. This arises when the production does not pass the truncation tolerance test, but ensures that the result includes all the radioactive products. Stable products which are descendants of this node may be calculated if they themselves pass the ignore tolerance test.
- | This code indicates that the chain will be fully truncated at this node, and the result will include this node. This arises when the node is a stable isotope and does not pass the truncation tolerance test, but does pass the ignore tolerance test.
- < This code indicates that the chain will be fully truncated at this node and will not be included in the result. This arises when the production of this nuclide does not pass either the truncation or the ignore tolerance test.
- / This code indicates that only the radioactive decays of the chain will be followed after this node. This arises when the production does not past either the truncation tolerance test or the ignore tolerance test, but ensures that the result includes all the radioactive products. Stable products which are descendants of this node will automatically be ignored.

truncation production: (0.00306937 This indicates the relative production at the end of operation (end of the last schedule) of this nuclide from the initial isotope during the truncation calculation. As explained in Chapter 3 of the Technical Manual, this represents the total production of this nuclide during the whole problem, assuming that none of it is transmuted or decays further. If this production is not calculated,

for example, because the chain is only being followed on radioactive reactions and this nuclide is stable, then this entry will be 'N/C' for 'Not-Calculated'.

final estimated production: -> 0.00234645) This indicates an estimate of the relative production at the end of operation (end of the last schedule) of this nuclide from the initial isotope. This value is calculated while performing the truncation test for its first product. Therefore, it represents the true final production (not the total as in the previous entry) and is only calculated if the first product is calculated. The relative production of the first product is not calculated, for example, if the chain is continuing only on the radioactive reactions, and the first product is stable. In this case, this entry will be 'N/C' for 'Not-Calculated'.

Acknowledgements

Much of the development used to create ALARA, and in particular, the handling of loops, was inspired by vigorous discussion with the late Prof. Emeritus Charles Maynard. He always insisted that loops were usually not important, and when they were, the solution could be easily found. I am happy to have proven him correct.

In addition, Jim Sisolak, Jeff Crowell and Prof. Jake Blanchard have all, at times, been good listeners to help me focus my ideas and develop my methods.

Partial support for this work was provided by the Fusion Technology Institute of the University of Wisconsin-Madison under DOE Grants DE-FG02-94ER54244 and DE-AS08-88DP10754 and Sandia National Laboratory contract Al-7232.

References

[1] P.P.H. Wilson and D.L. Henderson, "Qualitative Analysis of Physical and Mathematical Approximations Necessary for Induced Radioactivity Calculations of Fusion Devices", *Fusion Eng. and Design*, **36**, 415 (1997).

[2] T.Y. Sung and W.F. Vogelsang, "DKR: A Radioactivity Calculation Code for Fusion Reactors," UWFDM-170, Fusion Technology Institute, University of Wisconsin-Madison, Madison, Wisconsin, September 1976.

[3] D.L. Henderson and O. Yasar, "DKRICF: A Radioactivity and Dose Rate Calculation Code Package: Vols. I & II," UWFDM-714, Fusion Technology Institute, University of Wisconsin-Madison, Madison, Wisconsin, November 1986. This code package is available from the Radiation Shielding Information Center (RSIC) at Oak Ridge National Laboratory as Computer Code Collection entry CCC-323-DKR.

[4] DKR-PULSAR is a new version of the DKR-ICF code which implements methods from Reference 9 for the exact treatment of pulsed history irradiation. It is being developed by D.L. Henderson and H. Khater at the University of Wisconsin-Madison.

[5] P.P.H. Wilson and D.L. Henderson, "Expanding Towards Excellence: Ironing Out DKR's Wrinkles", UWFDM-995, University of Wisconsin Fusion Technology Institute, Madison, Wisconsin, November 1995.

[6] R.A. Forrest and J-Ch. Sublet, "FISPACT3 - User Manual," AEA/FUS 227, April 1993.

[7] J. Jung, "RACC: Theory and Use of the Radioactivity Code RACC," Argonne National Laboratory Report: ANL/FPP/TM-122, May 1979.

[8] H. Attaya, "Input Instructions for RACC-P," Argonne National Laboratory Report: ANL/FPP/TM-270, September 1994.

[9] J.E. Sisolak, S.E. Spangler and D.L. Henderson, "Pulsed/Intermittent Activation in Fusion Energy Systems," *Fus. Tech.*, **21**, 2145 (May 1992).

[10] P.P.H. Wilson and D.L. Henderson, "ALARA: Analytic and Laplacian Adaptive Radioactivity Analysis, Volume I, Technical Manual," UWFDM-1070, University of Wisconsin Fusion Technology Institute, Madison, Wisconsin, January 1998.

[11] S.E. Spangler, J.E. Sisolak and D.L. Henderson, "Calculational Models for the Treatment of Pulsed/Intermittent Activation Within Fusion Energy Devices," *Fusion Eng. and Design*, **22**, 349 (July 1993).

[12] S.E. Spangler, Master's Thesis, "A Numerical Method for Calculating Nuclide Densities in Pulse Activation Studies," University of Wisconsin-Madison, 1991.

[13] H. Bateman, *Proc. Cambridge Phil. Soc.* **15**, 423 (1910).

[14] E.S. Lee, "Computer Engineering: Computer Algorithms, Data Structures, and Languages," Prepared Notes, University of Toronto, 1989.

A. ALARA_DC: Data Conversion for Code Interfacing

In designing ALARA for maximum usability, consideration was taken for the way that it would be implemented by the end user. Most important was to consider the origins of the various inputs which would be needed to complete each calculation. Every ALARA problem requires four distinct types of input:

1. cross-section, decay and gamma libraries
2. geometry/mixture definitions
3. group-wise flux input, and
4. pulsing and history information.

By facilitating the conversion of these various inputs from other standard formats to that required by ALARA, the ease of use for the end user is enhanced. Such conversions are available for input types 1-3, while input type 4, pulsing and history information is particular to the pulsed history activation calculation. ALARA Data Conversion [ALARA_DC] is being written with these conversion needs in mind.

It was originally written to convert the cross-section, decay and gamma data from various international standard text formats to the proprietary data format required by ALARA (see Appendix B for binary data format). However, because the geometry definitions are required in various formats for neutron transport calculations which generate the group-wise fluxes in various formats, ALARA_DC is being extended to convert these data as well.

While most transport calculations require geometry definitions at least as specific as those needed by ALARA, the mixture definitions often lack certain trace elemental quantities which are unimportant for transport calculations but may be important for activation calculations. For example, a transport calculation through a steel block may define the mixture as iron for the purpose of the transport calculation while the alloying concentrations of nickel, chromium, and other elements are very important in determining the activation characteristics of the material. Therefore, ALARA_DC will be designed to extract the geometry definitions from the transport calculation and allow the user to modify/upgrade the mixture definition. In different cases, this geometry information may be extracted from either transport calculation input (deterministic calculations) or output (Monte Carlo calculations with combinatorial geometries).

The flux data can also take a number of different formats. Some of the available codes share standard binary and/or text based data formats for these results while others have their own formats. ALARA_DC will be extended to first convert the standard shared formats and then the more popular unique formats. In some cases, this conversion will be simultaneous to the geometry conversion, and in others, the fluxes will be interactively extracted from the available output.

B. Binary Reaction Library Format

Because the reaction schemes/chains are created by a depth first search using the data from the transmutation and decay libraries, these libraries need to be accessed extensively and randomly. In the past, such random access was not possible because of the limits on mass storage devices. Currently, in a text format, such random access would still be very tedious. To ensure that this random access does not create a drag on ALARA, it is necessary to either store the entire library in memory or use a binary file format. Because the libraries are often quite large (many MB) a simple binary format was designed. This section will describe the formats for the binary files and their indexes, which are generated in a text format and then appended in binary format to the end of the binary library. The transmutation library and decay library formats are no longer supported directly by ALARA. Instead, one transmutation library and one decay library must be combined into a single mixed or merged library format (see Section B.4 for difference).

The format of the binary file will be described by listing, in order, the data written to the file using the format: *(data type)*Description[**size**].

B.1. Transmutation Library

- *(long)*File Position of Index[**1**]
- *(int)*Number of Parent Isotopes[**1**]
- *(int)*Number of Neutron Energy Groups[**1**]
- *(int)*Flag indicating existence of Group Boundary info[**1**]
- *(float)*Group Boundary Data[**Number of Groups + 1** if above flag]
- *(int)*Flag indicating existence of Integral Flux Data[**1**]
- *(float)*Integral Flux Data[**Number of Groups** if above flag].
- Parent Isotope Info
  - *(int)*Parent KZA[**1**]
  - *(int)*Number of Reactions[**1**]
  - Reaction info once for each reaction
    - *(int)*Daughter KZA[**1**]

        * *(char)*Emitted Particles[**6**]
        * *(float)*Cross-section Data[**Number of Groups**]

This is followed by the index:

- *(char)*Library Type[**1**]
- *(int)*Number of Parent Isotopes[**1**]
- *(int)*Number of Neutron Energy Groups[**1**]
- *(int)*Special Code for Group Boundary Data[**1**]
- *(long)*File Index of Group Boundary Data[**1**]
- *(int)*Special Code for Integral Flux Data[**1**]
- *(long)*File Index of Integral Flux Data[**1**]
- Parent Index Info
    - *(int)*Parent KZA[**1**]
    - *(int)*Number of Reactions[**1**]
    - *(long)*File Index of This Parent[**1**]
    - Reaction info once for each reaction
        * *(int)*Daughter KZA[**1**]
        * *(char)*Emitted Particles[**6**]
        * *(long)*File Index of This Reaction[**1**]

B.2. Decay Library

- *(long)*File Position of Index[**1**]
- *(int)*Number of Parent Isotopes[**1**]
- Parent Isotope Info
    - *(int)*Parent KZA[**1**]
    - *(int)*Number of Decay Paths[**1**]
    - *(float)*Half Life[**1**]
    - *(float)*Average Beta Energy[**1**]
    - *(float)*Average Gamma Energy[**1**]

  - *(float)*Average Alpha Energy[**1**]
  - Reaction info once for each decay path
    * *(int)*Daughter KZA[**1**]
    * *(char)*Daughter Flag[**1**]
    * *(float)*Branching Ratio[**1**]

This is followed by the index:

- *(char)*Library Type[**1**]
- *(int)*Number of Parent Isotopes[**1**]
- Parent Index Info
  - *(int)*Parent KZA[**1**]
  - *(int)*Number of Decay Paths[**1**]
  - *(long)*File Index of This Parent[**1**]
  - Reaction info once for each decay path
    * *(int)*Daughter KZA[**1**]
    * *(char)*Daughter Flag[**1**]
    * *(long)*File Index of This Decay Path[**1**]

B.3. Mixed Reaction Library

- *(long)*File Position of Index[**1**]
- *(int)*Number of Parent Isotopes[**1**]
- *(int)*Number of Neutron Energy Groups[**1**]
- *(int)*Flag indicating existence of Group Boundary info[**1**]
- *(float)*Group Boundary Data[**Number of Groups + 1** if above flag]
- *(int)*Flag indicating existence of Integral Flux Data[**1**]
- *(float)*Integral Flux Data[**Number of Groups** if above flag].
- Parent Isotope Info
  - *(int)*Parent KZA[**1**]
  - *(int)*Number of Reactions[**1**]

  - Reaction info once for each transmutation reaction
    * *(int)*Daughter KZA[**1**]
    * *(int)*Length of Emitted Information[**1**]
    * *(char)*Emitted Particles[**Length of Emitted Info**]
    * *(float)*Cross-section Data[**Number of Groups+1**]

This is followed by the index:

- *(char)*Library Type[**1**]
- *(int)*Number of Parent Isotopes[**1**]
- *(int)*Number of Neutron Energy Groups[**1**]
- *(int)*Special Code for Group Boundary Data[**1**]
- *(long)*File Index of Group Boundary Data[**1**]
- *(int)*Special Code for Integral Flux Data[**1**]
- *(long)*File Index of Integral Flux Data[**1**]
- Parent Index Info
  - *(int)*Parent KZA[**1**]
  - *(int)*Number of Reactions[**1**]
  - *(long)*File Index of This Parent[**1**]
  - Reaction info once for each reaction
    * *(int)*Daughter KZA[**1**]
    * *(int)*Length of Emitted Information[**1**]
    * *(char)*Emitted Particles[**Length of Emitted Info**]
    * *(long)*File Index of This Reaction[**1**]

Note: decay rate for each daughter is stored in extra last element of cross-section array.

B.4. Merged Reaction Library

The merged reaction library format is identical to the mixed format except any reaction channels which lead to the same daughter have been merged into a single reaction channel, with the information about emitted particles being concatenated into a list. See comments in Section 3.3 about the varying usage of these two formats.

B.5. Gamma Source Library

- *(long)*File Position of Index[**1**]
- *(int)*Number of Parent Isotopes[**1**]
- Parent Isotope Info
  - *(int)*Parent KZA[**1**]
  - *(int)*Number of Spectra[**1**]
  - *(int)*Number of Discrete Gammas in each Spectra[**Number of Spectra**]
  - *(int)*Number of Interpolation Regions in each Spectra[**Number of Spectra**]
  - *(int)*Number of Interpolation Points in each Spectra[**Number of Spectra**]
  - Reaction info once for each spectrum
    - *(float)*Discrete Gamma Energies[**Number of Discrete Gammas(i)**]
    - *(float)*Discrete Gamma Intensities[**Number of Discrete Gammas(i)**]
    - *(float)*Interpolation Region Boundaries[**Number of Interpolation Regions(i)**]
    - *(float)*Interpolation Region Types[**Number of Interpolation Regions(i)**]
    - *(float)*Interpolation Point X-values[**Number of Interpolation Points(i)**]
    - *(float)*Interpolation Point Y-values[**Number of Interpolation Points(i)**]

This is followed by the index:

- *(char)*Library Type[**1**]
- *(int)*Number of Parent Isotopes[**1**]
- Parent Index Info
  - *(int)*Parent KZA[**1**]
  - *(int)*Number of Spectra[**1**]
  - *(long)*File Index of This Parent[**1**]
  - Reaction info once for each Spectra
    - *(int)*Number of Discrete Gammas[**1**]
    - *(int)*Number of Interpolation Regions[**1**]
    - *(int)*Number of Interpolation Points[**1**]

The repetition of much of this data in the index as well as the files allows the simple reading and extracting of the index without jumping back and forth in the binary file.

C. Material and Element Library Formats

For the convenience of the user, ALARA uses both a material and element library. The element library simply contains the natural isotopic breakdown of all the elements. The material library contains the elemental breakdown (using natural elemental compositions) of well-known materials. The usage of these libraries is described in more detail in Section 4.4.

C.1. Material Library

- Title indicating which material library this is
- Material info once for each material
  - *(char)*Material name [no white space allowed in name]
  - *(float)*Material density
  - *(int)*Number of elements in material
  - Element information once for each element
    - *(char)* Elemental symbol
    - *(float)* Weight fraction of this element
    - *(int)* Atomic number of element

C.2. Element Library

- Title indicating which element library this is
- Element info once for each element
  - elemental symbol
  - nominal elemental mass [g/mol]
  - atomic number
  - nominal elemental density [g/cm^3]
  - number of constituent isotopes
  - isotope information once for each naturally occurring isotope
    - mass number of isotope
    - atomic abundance of isotope

D. Flux File Formats

Every ALARA run requires a neutron flux file. Furthermore, ALARA has the ability to calculate a dose at a given point in space if a gamma flux importance file is given. The formats of these two files are similar. The first line is the title of the flux file. Both flux files are in a normal ASCII text format. Both files require one complete entry for each interval. A complete entry consists of an integer indicating which interval this flux data is for, followed by one flux value for each group. In the neutron flux file, the very first number (on the second line) must be an integer defining the number of groups.

E. Error Messages

ALARA has 54 built-in warning and error messages, the first 39 of which are used to indicate problems in the input file. When it encounters an error, ALARA will stop. For warnings, a message will be printed to the screen and execution will continue.

Warning 0: Using default filename 'input.file'.

You have specified no input file on the command line and ALARA will attempt to use the default input filename.

Error 1: Input file not found.

The filename which you specified on the command line is not found in the current directory.

Error 2: Invalid geometry type (make sure you have a problem title).

You have specified an invalid geometry type. This may indicate that the problem title is missing.

Error 3: Invalid geometry sub-type.

You have specified an invalid geometry sub-type. This may indicate that the problem title or geometry type is missing.

Error 4: No mixtures specified, finished empty problem.

You have indicated a problem with no mixtures. There is no need to continue.

Error 5: Unable to allocate mixtures; mixtures requested: *n.*

ALARA was unable to allocate enough memory for the *n.* mixtures which you requested.

Error 6: Mixtures out of order (check previous input). Occurred at mixture # *n.*

The mixtures appear to be out of order in the input file. This may indicate that one of the previous mixture definitions was incomplete.

Error 7: Invalid entry for number of components (check previous inputs).

The entry for the number of components is not an integer. This may indicate that one of the previous mixture definitions was incomplete, or that previous parts of this mixture definition are missing.

Error 8: Invalid material flag: *c.*

You have specified an invalid material flag. This flag should be either 'm','e','i', or 'l'. See Chapter 4.

Error 9: Material Library not found.

The material library named 'matlib' was not found in this directory. Either copy the matlib to this directory or make a link to it.

Error 10: Material not found in library: *matname*

The material, *matname*, which you specified in the input file is not available in the material library.

Error 11: Invalid elemental symbol, must be lower case letter (check other input).

The elemental symbol which you specified must be in lower case. This may indicate a problem previously in the input file.

Error 12: Invalid density flag. Should be 'N' or 'D' instead of *c.*

The density flag must be either an 'N' or a 'D' for number density or mass density, respectively.

Error 13: Element Library not found.

The element library named 'elelib' was not found in this directory. Either copy the elelib to this directory or make a link to it.

Error 14: Element not found in library: *sym.*

The element, *sym*, which you specified in the input file is not available in the element library.

Error 15: Missing number of coarse mesh points in dimension *n.*

The number of coarse mesh zones in dimension n is missing.

Error 16: Unable to allocate coarse mesh points, requested: *n.*

ALARA was unable to allocate enough memory for the n coarse mesh points which you requested.

Error 17: Coarse mesh <–> mixture definitions out of order on zone # *n.*

The coarse mesh mixture allocations appear to be out of order starting with the mixture definition for zone n.

Error 18: Invalid mixture number on zone # *n*.

The mixture assigned to zone *n* is not valid.

Error 19: Invalid interval definition selections: *c*.

The interval definition must be either 'i' or 'c' for input or calculated interval definitions, respectively.

Error 20: Too few fine mesh points; must be at least as many as coarse mesh points.

You have not specified enough fine mesh points. Each coarse mesh point must have at least one fine mesh point.

Error 21: Unable to allocate fine mesh points, requested: *n*.

ALARA was unable to allocate enough memory for the *n* fine mesh points which you requested.

Error 22: Fine mesh point definitions out of order on interval # *n*.

The fine mesh point definitions appear to be out of order, beginning with interval *n*.

Error 23: Invalid coarse mesh point number on interval # *n*.

In the definition of interval *n* you have specified an invalid coarse mesh point value.

Error 24: Error in zone boundaries, zone boundary less than previous boundary in dimension *n*.

When defining the zone boundaries, one boundary is less than the previous one, which would mean a negative volume.

Error 25: Error in zone boundaries, perhaps too few boundaries specified.

There is an error in the specification of the zone boundaries which may indicate that too few boundaries have been specified in this or a previous dimension.

Error 26: Error in number of fine mesh points per zone (check previous input).

There is an error in the specification of the number of intervals per zone which may indicate that too few intervals/zone have been specified in this or a previous dimension.

Warning 27: No fine mesh points in zone # *n*.

For some reason, zone *n* contains no intervals. This should only be possible with user input interval definition.

Error 28: Specified flux file not found: *filename.*

The flux file, *filename*, specified in the input file was not found in this directory.

Error 29: Flux inputs out of order. Check earlier flux file data.

The flux entries appear to be out of order. This may be due to a missing group flux in a previous entry.

Error 30: Unable to allocate dwell times and number of pulses for pulsing levels. Requested pulsing levels: *n.*

ALARA was unable to allocate enough memory for the pulsing schedule which you have requested with n pulsing levels.

Error 31: Invalid time unit (Check earlier input): *c.*

You have specified an invalid time unit in one of the pulsing schedules. This may indicate an error in previous input.

Error 32: Unable to allocate after-shutdown times and time units. Requested after-shutdown times: *n.*

ALARA was unable to allocate enough memory for the n after-shutdown times which you have requested.

Error 33: After shutdown times appear to be out of order on time # *n.*

The after-shutdown times appear to be out of order starting with number n.

Error 34: Invalid time unit (Check earlier input): *c.*

You have specified an invalid time unit in one of the pulsing schedules. This may indicate an error in previous input.

Warning 35: Invalid output resolution (may indicate other input errors). Will ignore: *c.*

One of the specified input resolutions is invalid and will be ignored.

Warning 36: Invalid output type (may indicate other input errors). Will ignore: *c.*

One of the specified input types is invalid and will be ignored.

Error 37: No output type specified before end of input file.

No output type was specified before the end of the file was reached.

Error 38: Incorrect number of gamma group boundaries (may indicate earlier errors in input file).

One of the gamma group boundaries appears incorrect. This may indicate an error in previous input.

Warning 39: Unable to open tree file for output. Continuing without tree output.

The tree file specified on the command-line cannot be opened for output.

Error 41: Unable to open binary data library: *filename.*

Unable to find the binary data library *filename*. This library must exist in this directory.

Warning 43: Not calculating gamma source or dose. Unable to open binary gamma library: *filename.*

Unable to open the gamma source library in this directory. The gamma source and/or dose will not be calculated.

Warning 44: Not calculating dose. Unable to open adjoint gamma flux file: *filename.*

The adjoint flux *filename* specified in the input file could not be opened for input. The does will not be calculated.

Warning 45: Not writing gamma source file. Unable to open file for output: *filename.*

Unable to open the gamma source file *filename* for output. The gamma source information information will not be written.

Error 46: Number of flux groups in cross-section file does not match: *n.*

The number of groups specified in the data library is different than the number in the flux file. Further calculation is not possible.

Error 47: Similar material must refer to previously defined mixture. Occurred creating mixture: *n.*

Mixture *n* is specified to be similar to another mixture. This other mixture must already be defined.

Warning 48: Unused mixture not added to root isotope list: *n.*

Mixture *n* appears to be allocated to no zones. This mixture will not be used.

Error 49: kza at this offset doesn't match current kza: *kza.*

The index into the binary data library for this isotope points to the wrong isotope, *kza*.

Error 50: #rxns at this offset doesn't match current index: *n.*

The index into the binary data library for this isotope points to an isotope with a different number of reactions, *n*, than the index specifies.

Error 51: #spectra at this offset doesn't match current index: *n.*

The index into the binary data library for this isotope points to an isotope with a different number of spectra, *n*, than the index specifies.

Warning 52: Reading input from stdin.

The input will be read from the standard input since you specified '-' for an input file on the command line.

Error 53: You appear to have defined a second input file: *filename.*

There is more than one filename specified on the command line: *filename*.

Error -99: Undefined Error, please consult code author.