ornl

RECEIVED
JAN 0 2 1997
OSTI

ORNL/TM-13347

OAK RIDGE NATIONAL LABORATORY

LOCKHEED MARTIN

DCFPAK: Dose Coefficient Data File Package for Sandia National Laboratory

K. F. Eckerman
R. W. Leggett

MASTER

MANAGED AND OPERATED BY
LOCKHEED MARTIN ENERGY RESEARCH CORPORATION
FOR THE UNITED STATES
DEPARTMENT OF ENERGY

ORNL-27 (3-96)

This report has been reproduced directly from the best available copy.

Available to DOE and DOE contractors from the Office of Scientific and Technical Information, P.O.Box 62, Oak Ridge, TN 37831; prices available from (423) 576-8401, FTS 626-8401.

Available to the public from the National Technical Information Service, U.S. Department of Commerce, 5285 Port Royal Rd., Springfield, VA 22161

This report was prepared a an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency thereof, nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof. The view and opinions of authors expressed herein do not necessarily state or reflect those of the UnitedStatesGovernment or any agency thereof.

ORNL/TM-13347

DCFPAK: Dose Coefficient Data File Package for Sandia National Laboratory

K.F. Eckerman and R.W. Leggett
Health Sciences Research Division
Oak Ridge National Laboratory

Manuscript Completed
July 31, 1996

Prepared for Sandia National Laboratory,
Albuquerque, NM 87185,
under contract AS-2472

Prepared by the
Oak Ridge National Laboratory
Oak Ridge, Tennessee 37831-6285
managed by
LOCKHEED MARTIN ENERGY RESEARCH CORP.
for the U.S. DEPARTMENT OF ENERGY
under contract DE-AC05-96OR22464.

DISCLAIMER

Portions of this document may be illegible in electronic image products. Images are produced from the best available original document.

CONTENTS

ACKNOWLEDGEMENT

The work described in this report was performed at Oak Ridge National Laboratory and was supported by Sandia National Laboratory under contract AS-2472. Earlier activities at ORNL in development of the dosimetric data files addressed here were sponsored by the Department of Energy, the Nuclear Regulatory Commission, and the Environmental Protection Agency. The support of these organizations has been critical to the development of our dosimetry research program and is gratefully acknowledged.

The authors are grateful for the contributions of David B. Clauss, Sandia National Laboratory, whose insights into the dosimetric needs in consequence analysis provided the stimulus for this work, and to Barbara Clark, Oak Ridge National Laboratory, for her assistance in the preparation of the final manuscript for publication.

ABSTRACT

The FORTRAN-based computer package DCFPAK (Dose Coefficient File Package) has been developed to provide electronic access to the dose coefficient data files summarized in Federal Guidance Reports 11 and 12. DCFPAK also provides access to standard information regarding decay chains and assembles dose coefficients for all dosimetrically significant radioactive progeny of a specified radionuclide. DCFPAK was designed for application on a PC but, with minor modifications, may be implemented on a UNIX workstation.

INTRODUCTION

The U.S. Environmental Protection Agency (EPA) has issued a series of "Federal Guidance" reports that provide the dosimetric information needed by government agencies to implement radiation protection programs in a consistent and adequately protective manner. This information is mainly in the form of "dose coefficients", or estimates of dose per unit exposure to individual radionuclides. These dose coefficients allow the user to relate concentrations of radionuclides in air, water, food, or soil to guidance for radiation dose to workers or members of the public.

Two Federal Guidance reports are currently in use. Federal Guidance Report No. 11 (Eckerman et al. 1988) provides dose coefficients, in the form of 50-year integrated dose equivalents, for acute ingestion or acute inhalation of radionuclides, based on the biokinetic and dosimetric models of ICRP Publication 30 (1979, 1980, 1981, 1988). Federal Guidance Report No. 12 (Eckerman and Ryman 1993) provides dose coefficients, in the form of dose per unit time-integrated exposure, for external exposure to radionuclides in air, water, or soil, based on state-of-the-art methods in external radiation dosimetry.

The tabulations in these two Federal Guidance reports include dose coefficients for the radionuclides considered in ICRP Publication 30, Limits for Intakes of Radionuclides by Workers (ICRP 1979, 1980, 1981, 1988). For each radionuclide and exposure mode, dose coefficients are provided for the seven organs or tissues assigned risk weighting factors in ICRP Publication 26 (1977): breast; lung; red marrow; bone surface; thyroid; gonads, representing the higher of the coefficients for ovaries and testes; and "remainder", representing the five remaining tissues receiving the highest doses. The tabulations include the effective dose equivalent, H_E, derived from the risk weighting factors and equivalent doses for these organs. In Federal Guidance Report No. 12, dose coefficients are also provided for skin because it is frequently the most highly irradiated organ for external exposure.

The software and data package DCFPAK (Dose Coefficient File Package) has been developed to allow electronic access to the full set of dose coefficients summarized in these two Federal Guidance reports and to facilitate consideration of dose coefficients for chains of radionuclides. In addition to the published dose coefficients, the DCFPAK libraries includes dose coefficients for 18 organs not addressed in Federal Guidance Report No. 11, and 17 organs not addressed in Federal Guidance Report No. 12 (Table 1). The additional dose coefficients were generated during the development of these two Federal Guidance reports but were excluded from the published tables to make the tables easier to use and to keep the reports at a reasonable length. For each radionuclide and exposure mode, the DCFPAK libraries include both the effective dose equivalent, H_E, based on the tissue weighting factors given in ICRP Publication 26 (1977), and the effective dose, E, based

Table 1. Tissues addressed in DCFPAK, Federal Guidance No. 11, and Federal Guidance No. 12.

Tissue	Abbreviation in DCFPAK libraries	Listed in:		
		DCFPAK	Federal Guidance 11	Federal Guidance 12
Adrenal	Adrenal	Yes	No	No
Urinary Bladder Wall	Bld Wall	Yes	No	No
Bone Surface	B Surface	Yes	Yes	Yes
Brain	Brain	Yes	No	No
Breast	Breast	Yes	Yes	Yes
Esophagus	Esophagu	Yes	No	No
Stomach Wall	St Wall	Yes	No	No
Small Intestine Wall	SI Wall	Yes	No	No
Upper Large Intestine Wall	ULI Wall	Yes	No	No
Lower Large Intestine Wall	LLI Wall	Yes	No	No
Kidney	Kidney	Yes	No	No
Liver	Liver	Yes	No	No
Lung	Lung	Yes	Yes	Yes
Muscle	Muscle	Yes	No	No
Ovaries	Ovaries	Yes	No	No
Testes	Testes	Yes	No	No
Gonad[a]	(Not used)	No	Yes	Yes
Pancreas	Pancreas	Yes	No	No
Red Marrow	R Marrow	Yes	Yes	Yes
Skin	Skin	Yes	No	Yes
Spleen	Spleen	Yes	No	No
Thymus	Thymus	Yes	No	No
Thyroid	Thyroid	Yes	Yes	Yes
Uterus	Uterus	Yes	No	No
Remainder[a]	(Not used)	No	Yes	Yes

[a]Represents multiple tissues considered explicitly in DCFPAK.

on the newer tissue weighting factors given in ICRP Publication 60 (1991). Tissue weighting factors recommended in ICRP Publication 26 and Publication 60 are compared in Table 2.

The DCFPAK libraries contain dose coefficients for each of nine modes of exposure to a given radionuclide: (1) ingestion; (2) inhalation; (3) submersion, meaning external exposure to (radiations emitted from) the radionuclide in air; (4) immersion, meaning external exposure to the radionuclide in water (from swimming); (5) external exposure to the radionuclide on the ground surface; and (6-9) external exposure to the radionuclide distributed to a depth of 1 cm, a depth of 5 cm, a depth of 15 cm, or an infinite depth in soil. The dose coefficients for internal exposures are 50-y integrated doses to organs following an acute intake and are given in units of Sv Bq^{-1}. The dose coefficients for external exposures are dose per unit time-integrated exposure. Dose coefficients for external exposures are in terms of Sv per Bq-s per unit volume of environmental medium (liter of air for submersion, liter of water for immersion, m^2 of soil for ground surface contamination, and m^3 of soil for subsurface contamination of soil).

The dose coefficient for a given radionuclide and exposure scenario does not reflect ingrowth of chain members in the environment, but dose coefficients for ingestion and inhalation do reflect the contribution to dose from ingrowth of chain members in the body after intake of the parent radionuclide. For either internal or external exposure to a radionuclide, DCFPAK determines the radionuclide decay chain (if any) of that radionuclide, truncates the chain after the last potentially significant decay chain member, and assembles the dose coefficients for all potentially significant chain members. For internal exposure, DCFPAK truncates a chain if the cumulative energies for alpha, electron, and photon radiation over a 100-year period are changed less than 1% by the addition of subsequent chain members. The same procedure is followed for external exposures but with consideration restricted to electrons and photons.

The DCFPAK package, which is written in FORTRAN 77, was developed for use as a module of a radiological assessment package currently used at Sandia National Laboratory. However, DCFPAK is generic in nature and should be compatible with other FORTRAN-based radiological assessment packages. It was designed for use on a personal computer or work station and can be run interactively or in batch mode.

Table 2. Tissue weighting factors given in ICRP Publication 26 (1977) and ICRP Publication 60 (1991)[a].

	Tissue weighting factor (w_T)	
Organ or tissue	ICRP Pub. 26	ICRP Pub. 60
Gonads	0.25	0.20
Bone marrow (red)	0.12	0.12
Colon	--	0.12
Lung	0.12	0.12
Stomach	--	0.12
Bladder	--	0.05
Breast	0.15	0.05
Liver	--	0.05
Oesophagus	--	0.05
Thyroid	0.03	0.05
Skin	--	0.01
Bone surface	0.03	0.01
Remainder	0.30	0.05

[a]The values were developed for a reference population of equal numbers of both sexes and a wide range of ages. In the definition of effective dose they apply to workers, to the whole population, and to either sex.

For illustrative purposes DCFPAK was linked to a sample "driver" code called READEM (source code, READEM.FOR and executable file, READEM.EXE). READEM allows an interactive examination of DCFPAK's output for any of the 9 exposure modes and approximately 800 radionuclides addressed in this package.

The DCFPAK package consists of the following data libraries (data files) and software files:

1. nine libraries of dose coefficients, corresponding to the nine exposure modes listed above;

2. a library of information on the half-lives, modes of decay, branching fractions, and decay products of all 838 radionuclides considered in these two Federal Guidance reports;

3. a "data access" module that accesses the 10 data files described above and returns the relevant information;

4. an include file containing information on dimensioning of arrays in DCFPAK;

5. two include files containing common blocks;

6. an include file that provides unit designations to be reserved for IO used by DCFPAK;

7. an "initialization" file that informs DCFPAK of the location of all data files and their record lengths;

8. an illustrative driver code that serves the dual purpose of allowing an examination of available DCFPAK output and illustrating how to establish communication between the user's driver code and DCFPAK's data access module.

To access DCFPAK, the user must insert the following into his driver code: include statements that reference the files described in items 4-6 above, and a call statement, described later, that transfers the input variables describing the radionuclide, particle size, desired output, and other case-specific information to DCFPAK. The user's code and the source code DCFPAK.FOR must then be compiled and the resulting object files must be linked.

DATA FILES IN DCFPAK

Ten data files are included in DCFPAK. The data files are all "formatted direct-access files" consisting of ASCII characters, with all records in a given file of the same length and each record ending with an ASCII carriage-return (CR) and line-feed (LF) character. One of these data files contains information such as the half-time, decay modes, and branching fractions of the radionuclides and is used by DCFPAK to assemble the information on the decay chain for the parent radionuclide. The other nine data files contain dose coefficients corresponding to the nine exposure modes described earlier. A short description of each of the ten data files is given in the following.

DFEXTINT.NDX

This file is used by DCFPAK to assemble the information on the decay chain for the parent. The file contains 839 lines, including a header line with such information as the record format and one line of information for each of the 838 radionuclides addressed by DCFPAK. The line of information for a radionuclide contains the name and half-life of the radionuclide, the key (line number) of the radionuclide record in all other files, and a summary of decay properties of each radionuclide. For example, the line of information for Bi-212 gives the following information in the indicated order:

name of radionuclide (Bi-212);
half-life (60.55 m);
decay modes (B-A, indicating beta-minus and alpha);
beginning line number for Bi-212 in the inhalation dose coefficient file (2282);
record number for Bi-212 in the ingestion dose coefficient file (699);
record number for Bi-212 in each of the external dose coefficient files (690);
record number in DFEXTINT.NDX of decay product Po-212 (504);
branching fraction to Po-212 (6.4070E-01);
record number in DFEXTINT.NDX of decay product Tl-208 (762);
branching fraction to Tl-208 (3.5930E-01);
total energy (MeV) of emitted alpha particles (2.1741);
total energy (MeV) of emitted beta particles (0.4720);
total energy (MeV) of emitted photons (0.1855);
atomic weight (211.991271);
date of creation of the radioactive decay data (05-May-77).

DFINHS.DAT

This file is an expanded version of Table 2.1 of Federal Guidance Report No. 11. The file contains dose coefficients for the case of acute intake of a radionuclide by inhalation. The dose coefficients represent the dose over a 50-year period following intake (Sv Bq^{-1} inhaled). Tabulated dose coefficients are for the default particle size 1 μm (activity median aerodynamic diameter, or AMAD), except for a few cases in which particle size is not relevant (e.g., for vapors). As described below, the tabulated dose coefficients are used to generate dose coefficients for any particle size specified by the user.

For each radionuclide, inhalation dose coefficients are included for each of the lung clearance classes applied to that radionuclide in ICRP Publication 30 (1977, 1980, 1981, 1988). The three main clearance classes are "D", "W", and "Y", where Class D refers to a clearance time of days, W to a clearance time of weeks, and Y to a clearance time of years from the pulmonary region of the lung. The occasionally used Class "V" refers to rapid clearance expected to occur for vapor forms of some elements. Three special clearance classes are used for carbon: "c", applied to labeled organic compounds, "m", applied to carbon monoxide, and "d", applied to carbon dioxide; differences in the three classes are associated mainly with differences in the relative amounts of deposited carbon exhaled and absorbed to blood and in the biokinetics of carbon after absorption to blood.

For each radionuclide, one or more lines of data are included for each clearance class. The first line includes the name of the radionuclide, the clearance class, the gastrointestinal absorption fraction (f_1 value) for material cleared to the gastrointestinal tract, a dose coefficient for each of the 23 tissues considered (Table 1), the effective dose equivalent, and the effective dose. For clearance class D, W, or Y (which apply to particulate matter), a second line of data provides the percentage of the dose coefficient for each organ that is attributable to deposition in the nasal-pharynx (N-P) and pulmonary (P) regions of the respiratory tract. As described below, these percentages are used by DCFPAK to calculate dose coefficients for particle sizes other than 1 μm.

Dose coefficients for particle sizes other than 1 μm can be derived from the tabulated data for 1 μm as a result of the assumption in the underlying respiratory tract model (ICRP 1979) that the rate of clearance of material from the tract is independent of particle size. This assumption implies that the inhalation dose coefficient for a given radionuclide and clearance class is determined by the relative fractions of inhaled activity assigned to N-P, T-B, and P. Suppose, for example, that the user is considering inhalation of Be-10 of Class W and particle size 0.2 μm. For this particle size, the deposition fractions for N-P, T-B, and P are 0.05, 0.08, and 0.5, respectively (with the missing fraction, 0.37, assumed to be promptly exhaled). For particle size 1 μm, the deposition fractions for N-P, T-B, and P are 0.30, 0.08, and 0.25, respectively. According to the file DFINHS.DAT, for Be-10 of Class W and particle size 1 μm, 26% of the inhalation dose coefficient (DC) for breast, for example, is attributable to activity deposited in N-P and 41% is attributable to activity deposited in P. Therefore, 33% is attributable to activity deposited in T-B. The dose coefficient for the breast for 0.2 μm would be

$$[(0.05/0.30) \times (0.26 \times DC)] + [(0.08/0.08) \times (0.33 \times DC)] + [(0.5/0.25) \times (0.41 \times DC)] = 1.1933 \times DC,$$

i.e., 1.1933 times the dose coefficient for particle size 1 μm.

For a given particle size in the range 0.1-20 μm, deposition fractions for the regions N-P, T-B, and P are based on the piecewise linear curves shown in Fig. 1 (after Fig. 5.1 of Part 1 of ICRP Publication 30 (1979)). For particle sizes less than 0.1 μm or greater than 20 μm, DCFPAK assigns fractional depositions in these regions corresponding to a particle size of 0.1 μm or 20 μm, respectively.

The esophagus was not depicted explicitly in the dosimetric human phantom used in ICRP Publication 30 and was not considered as a target organ in that document. However, the esophagus is addressed in DCFPAK because dose to the esophagus is considered in the calculation of the effective dose, E. Because internal dose calculations are not available for the esophagus, the thymus has been used as a surrogate. The esophagus was recently introduced into ORNL's human phantom and was considered explicitly in the external dose coefficients tabulated in Federal Guidance No. 12 and in DCFPAK.

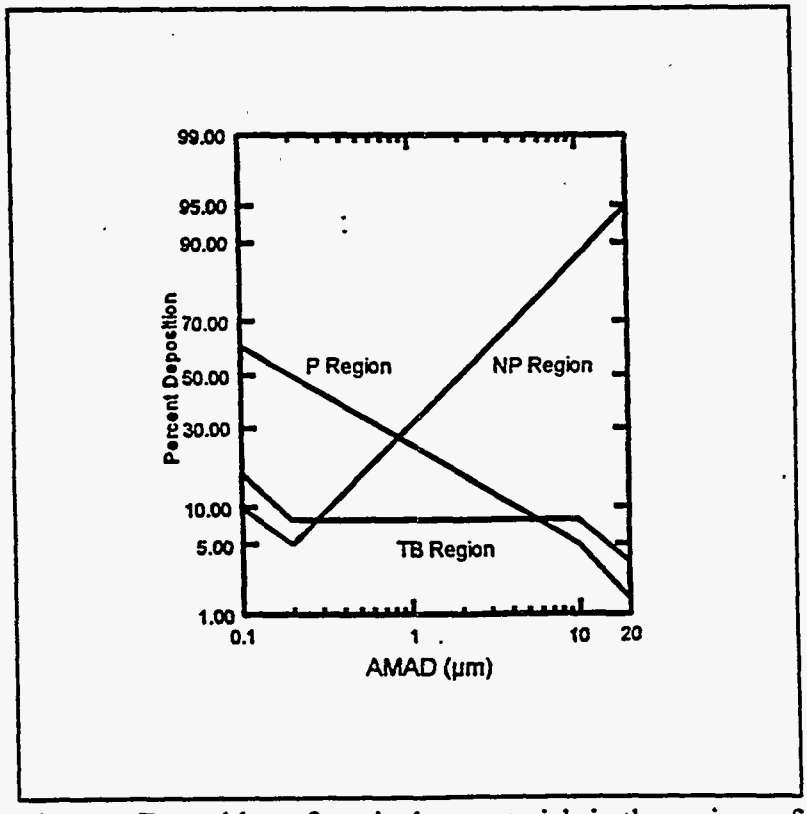

Figure 1 Deposition of particulate materials in the regions of the lung as a function of the activity median aerodynamic diameter (AMAD).

For radionuclides addressed in Part 1 of ICRP Publication 30 (1979), an inhalation dose coefficient of 0.0 is given for brain. This is because the brain was not included in the human phantom used in the ICRP dosimetry until after the completion of Part 1, and because there is no suitable surrogate for brain among the organs addressed.

DFINGS.DAT

This file is an expanded version of Table 2.2 of Federal Guidance Report No. 11. The file contains dose coefficients for the case of acute intake of a radionuclide by ingestion. The dose coefficients represent the dose over a 50-year period following acute intake (Sv Bq^{-1} ingested).

With a few exceptions, one line of data is given for each radionuclide. That line includes the name of the radionuclide, the gastrointestinal absorption fraction (f_1 value) for ingested activity, a dose coefficient for each of the 23 tissues considered (Table 1), the effective dose equivalent, and the effective dose. For a few radionuclides, one or more additional lines of dose coefficients are given, corresponding to alternate f_1 values used for certain forms of the radionuclide. For example, for radioisotopes of strontium, one line of dose coefficients corresponds to the f_1 value of 0.3 used for strontium compounds other than titanate and a second line gives dose coefficients for the f_1 value of 0.01 applied to strontium titanate.

For the reasons given in the discussion of inhalation dose coefficients, an ingestion dose coefficient of 0.0 for brain is given for each of the radionuclides addressed in Part 1 of ICRP Publication 30 (1979). Also, for reasons discussed above, ingestion dose coefficients for the thymus are applied to the esophagus.

FGR12F31.DAT

This file is an expanded version of Table III.1 of Federal Guidance Report No. 12. The file contains dose coefficients for the case of external exposure to the radionuclide in air (submersion). The units are dose per time-integrated air concentration (Sv per Bq-s L^{-1} air).

One line of data is given for each radionuclide. That line contains the name of the radionuclide, a dose coefficient for each of the 23 tissues considered (Table 1), the effective dose equivalent, and the effective dose.

FGR12F32.DAT

This file is an expanded version of Table III.2 of Federal Guidance Report No. 12. The file contains dose coefficients for external exposure to the radionuclide in water ("immersion"), from swimming. The units are dose per time-integrated water concentration (Sv per Bq-s L^{-1} water).

One line of data is given for each radionuclide. That line contains the name of the radionuclide, a dose coefficient for each of the 23 tissues considered (Table 1), the effective dose equivalent, and the effective dose.

FGR12F33.DAT

This file is an expanded version of Table III.3 of Federal Guidance Report No. 12. The file contains dose coefficients for external exposure to the radionuclide on the ground surface. The units are dose per time-integrated surface concentration (Sv per Bq-s m^{-2}).

One line of data is given for each radionuclide. That line contains the name of the radionuclide, a dose coefficient for each of the 23 tissues considered (Table 1), the effective dose equivalent, and the effective dose.

FGR12F34.DAT

This file is an expanded version of Table III.4 of Federal Guidance Report No. 12. The file contains dose coefficients for external exposure to the radionuclide distributed to a depth of 1 cm in the soil. The units are dose per time-integrated concentration (Sv per Bq-s m^{-3}).

One line of data is given for each radionuclide. That line contains the name of the radionuclide, a dose coefficient for each of the 23 tissues considered (Table 1), the effective dose equivalent, and the effective dose.

FGR12F35.DAT

This file is an expanded version of Table III.5 of Federal Guidance Report No. 12. The file contains dose coefficients for external exposure to the radionuclide distributed to a depth of 5 cm in the soil. The units are dose per time-integrated concentration (Sv per Bq-s m^{-3}).

One line of data is given for each radionuclide. That line contains the name of the radionuclide, a dose coefficient for each of the 23 tissues considered (Table 1), the effective dose equivalent, and the effective dose.

FGR12F36.DAT

This file is an expanded version of Table III.6 of Federal Guidance Report No. 12. The file contains dose coefficients for external exposure to the radionuclide distributed to a depth of 15 cm in the soil. The units are dose per time-integrated concentration (Sv per Bq-s m^{-3}).

One line of data is given for each radionuclide. That line contains the name of the radionuclide, a dose coefficient for each of the 23 tissues considered (Table 1), the effective dose equivalent, and the effective dose.

FGR12F37.DAT

This file is an expanded version of Table III.7 of Federal Guidance Report No. 12. This file contains dose coefficients for external exposure to the radionuclide distributed to an infinite depth in the soil. The units are dose per time-integrated concentration (Sv per Bq-s m^{-3}).

One line of data is given for each radionuclide. That line contains the name of the radionuclide, a dose coefficient for each of the 23 tissues considered (Table 1), the effective dose equivalent, and the effective dose.

DCFPAK SOFTWARE

The DCFPAK software files, written in FORTRAN 77, are described below. In the following descriptions, the code that calls DCFPAK's data access module will be referred to as the "user's driver code".

DCFPAK.FOR

This is the FORTRAN source code for the data access module that accesses the data files and returns the relevant information. It should be compiled on the user's system.

PAKPARM.FOR

This "include" file should be referenced by the user's driver code, along with the include file DCFPAK.CMN described below. The file contains the following two lines of FORTRAN code that determine the dimensioning of arrays in DCFPAK:

```
integer mspec, morg, mfact
parameter(mspec = 25, morg = 25, mfact = 9)
```

Here, MSPEC is the maximum length of the decay chain, MFACT is the maximum number of dose factors types, and MORG is the maximum number of organs in the dose factor file.

DCFPAK.CMN

This include file should be referenced by the user's driver code in any routines that access dosimetric information for the chain members. As noted above, PAKPARM.FOR should also be included in these routines. The file contains the common blocks that will contain the information extracted from the data files by DCFPAK:

```
 common /dfacts/ organ(morg), df(mspec, mfact, morg), flinh(mspec),
:                florl(mspec), classo(mspec), iflag(mspec, mfact),
:                nint, next

 common /radat/ thalf(mspec), iu(mspec), nucnam(mspec),
:               branch(mspec, mspec), lmr(mspec),
:               ibr(mspec,mspec), nbr(mspec), nspec
```

The common block DFACTS contains the dose coefficients. The variables in the dimension statements are defined in the include file PAKPARM.FOR. The definitions of the variables in this block are as follows:

ORGAN: A character(*9) array of organ names of maximum length MORG. The 23 organs addressed by DCFPAK are listed in Table 1.

DF: A real variable array of dose coefficients by chain member, exposure pathway, and organ.

F1INH: Real variable array of gastrointestinal uptake (f_1) values for inhalation, by chain member.

F1ORL: Real variable array of gastrointestinal uptake (f_1) values for ingestion, by chain member.

CLASSO: A character(*1) array of clearance class notation for the inhalation coefficients of the chain members.

IFLAG: An array of logical flags, by chain member and pathway, set to true in each case for which DCFPAK returns a coefficient in DF array.

NINT: Integer variable for the number of chain members that are used in the evaluation of inhalation and ingestion intakes (i.e., the length of the chain, or truncated chain, used in the calculation).

NEXT: Integer variable for the number of chain members that must be considered for external exposures.

The common block RADAT, shown above, contains information on the decay chain under consideration. The definitions of the variables in this block are as follows:

THALF: A character(*8) array of the half-life of each chain member.

IU: A character(*2) array of the units of the half-life of each chain member.

NUCNAM: A character(*7) array of the names of the chain members.

BRANCH: A two-dimensional real array (upper triangular matrix) of branching fractions, with an entry representing the fraction of the decays of the ith chain member forming the jth chain member.

LMR: A double precision array of decay constants (d^{-1}) for the chain members.

NBR: An integer array containing the number of chain members that decay directly to the specific chain member.

IBR: An integer array containing the indices of the chain members which form the specific chain member. For example, (ibr(I, ispec), I=1, nbr(ispec)) are the indices of the precursor of member ispec.

NSPEC: The length of the decay chain.

BATCH.CMN

This include file should be referenced by the user's driver code. The file contains two lines of code, including a common block with a logical flag that determines whether DCFPAK is operated in an interactive or a non-interactive mode:

```
logical dbatch
common/dcalbat/dbatch
```

The DBATCH logical variable inserted into the user's code should be set to "false" for the interactive mode and "true" for the non-interactive mode. Setting the DBATCH logical variable to "true" eliminates output to the screen that occurs in the interactive mode.

IOLIST.CMN

This include file should reside in the same directory as the file DCFPAK.FOR when the latter is compiled. This four-line file contains the unit designations for the IO used by DCFPAK:

```
 integer idex, ingx, inhx, i31, i32, i33, i34, i35, i36,
:        i37, olog
 parameter (idex = 10, ingx=11, inhx=12, i31=13, i32=14,
:           i33=15, i34=16, i35=17, i36=18, i37=19, olog=20)
```

The unit designations in IOLIST.CMN must be reserved for DCFPAK input and output; that is, they must differ from unit designations in the user's driver code. In addition to these units, DCFPAK uses unit 40.

DCFPAK.INI

This "initialization" file should be placed in the directory where the software is run. The file informs DCFPAK of the location of all data files and their record lengths. This file enables the user to place DCFPAK's data files in a directory distinct from that used for code output. The following is a sample version of DCFPAK.INI. The real version should show the actual path to each file indicated in the second field:

```
'snlindex.ndx',   'c:\snl\dcf\dfextint.ndx', 125
'ingestsf.dfs',   'c:\snl\dcf\dfings.dat'  , 240
'inhalesf.dfs',   'c:\snl\dcf\dfinhs.dat'  , 232
'submrsin.dfs',   'c:\snl\dcf\fgr12f31.dat', 232
'imersion.dfs',   'c:\snl\dcf\fgr12f32.dat', 232
'grsurf00.dfs',   'c:\snl\dcf\fgr12f33.dat', 232
'grvol_01.dfs',   'c:\snl\dcf\fgr12f34.dat', 232
```

```
'grvol_05.dfs',   'c:\snl\dcf\fgr12f35.dat', 232
'grvol_15.dfs',   'c:\snl\dcf\fgr12f36.dat', 232
'grvolinf.dfs',   'c:\snl\dcf\fgr12f37.dat', 232
'EOF', '  '       <- end of file marker
```

The first field is the file name within DCFPAK, and the second field is the full name of the file, including its path. For example, the record

'snlindex.ndx', 'snl\dcf\dfextint.ndx'

indicates that the file 'snlindex.ndx' is to be read as snl\dcf\dfextint.ndx, where "snl\dcf" is the path to the directory containing the file. The user should not edit the first field and should limit the path in the second field so that the field length does not exceed 64. The third field is the integer value for the record length for the file to be used in the FORTRAN open statement.

Not all FORTRAN compilers define the record length in the same manner when opening direct access files. DCFPAK was developed using the Microsoft FORTRAN compiler, which uses the physical length of the record less 2 bytes corresponding to the carriage return (CR)- line feed (LF) that separate formatted direct records. The integer values in the third field above specify the RECL parameter for the Microsoft compiler. Some other compilers (e.g, Lahey) specify RECL as the length of the physical record, which includes the 2 bytes associated with the CR-LF. For such compilers, DCFPAK.INI should be edited to increase the integer values indicated above by 2. For additional information, see the comments included in the files after the EOF marker.

READEM.FOR

This is the source code for a sample driver code that accesses DCFPAK and returns to the screen the decay chain and dose coefficients for the chain members, based on the user's input. This code is included for illustrative purposes.

READEM.EXE

This is the executable file for READEM.FOR.

CONNECTING DCFPAK TO THE USER'S DRIVER CODE

The DCFPAK data files and software are distributed in a self-extracting ZIP file named SNL.EXE. After extraction, the DCFPAK package can be connected to the user's driver code as follows:

Step 1: Copy the file snl.exe into a directory, preferably the directory that contains the user's driver code.

Step 2: Type "snl" and hit the "Enter" key. This extracts all data and software files. The data files may be moved to another directory. The new address of the data files must be specified in the file DCFPAK.INI.

Step 3: In the user's driver code, insert include statements referencing the include files PAKPARM.FOR, DCFPAK.CMN, BATCH.CMN, and IOLIST.CMN as illustrated in the sample driver source code, READEM.FOR.

Step 4: Insert the following "call" into the user's drive code, as illustrated in READEM.FOR:

CALL DOSECOF (NUKE, F1ORAL, CLASS, F1INHI, AMAD, IPATH)

to obtain the information for radionuclide NUKE. The dose conversion factor information is returned in the common blocks discussed above. The variables NUKE and IPATH must be specified. If the other variables are not specified, default values are supplied by DCFPAK. If the library has multiple sets of dose coefficients (e.g., corresponding to different f1 values for a radionuclide), then DCFPAK will assign the most conservative set of coefficients. To obtain a particular set of coefficients one can specify the f1 values for inhalation and ingestion as well as the clearance class for inhalation. These variables are described in the following section. Illustrative read statements are given in the sample driver source code, READEM.FOR.

Step 5: As discussed in the following section and illustrated in READEM.FOR, insert data input statements into the user's driver code to define the radionuclide (NUKE), the operative exposure pathway (IPATH), and optional input if used.

Step 6: Insert the calls "CALL OPENEM" and "CALL CLOSEM" into the user's driver code in appropriate places for opening and closing the data files, respectively, as illustrated in READEM.FOR.

Step 7: Compile DCFPAK.FOR and the user's driver code, and link the resulting object files.

INPUT TO THE USER'S DRIVER CODE

Prior to calling the routine DOSECOF, a call should be made to the routine NUKEOK to determine if the nuclide of interest is contained in DCFPAK's data files. Thus, the user's code should include the following call:

CALL NUKEOK(NUKE, OK)

The variables are defined as:

NUKE: This character*7 variable represents the name of the radionuclide of interest. The naming convention for radionuclides follows standard practice (e.g., Co-60, Cs-137, Tc-99m). DCFPAK includes a character*7 function CHECK, which is called to place the nuclide name in the proper format.

OK: This is a logical variable which is set to true if the nuclide is present in the data files and is otherwise returned as false.

If the nuclide is present in the data files, then the user's driver code should access the data files through the call to DOSECOF described above:

CALL DOSECOF (NUKE, F1ORAL, CLASS, F1INHI, AMAD, IPATH)

The user's driver code should be set up so that the variables in this call are input variables, even though default values are supplied by DCFPAK for most of these variables. The variables are defined in the following.

NUKE: This variable was defined above.

F1ORAL: This is a real variable representing the value of the f_1 parameter (gastrointestinal uptake factor) associated with the ingested chemical form of the radionuclide. f_1 represents the fraction of the ingested material reaching blood in the absence of radioactive decay. For most radionuclides, a single f_1 value is applied to all chemical forms. However, for uranium, plutonium, and a few other elements, separate f_1 values are applied to relatively soluble and relatively insoluble compounds. If F1ORAL is set to zero, DCFPAK will return the ingestion coefficients for the chemical form that has the highest effective dose.

F1INHI: This is a real variable representing the gastrointestinal uptake factor for inhaled material that moves from the respiratory tract to the gastrointestinal tract. If F1INHI is set to zero, DCFPAK will assign the default value to F1INH1 associated with the radionuclide and the CLASS variable.

CLASS: This is a character*1 variable that specifies the clearance class for inhaled forms of the radionuclide. The dose coefficients are based on the lung model of ICRP Publication 30, and the variable has values of "D", "W", "Y", and "V". Class D refers to a clearance time of days, W to a clearance time of weeks, and Y to a clearance time of years. The less frequently used Class "V" refers to rapid clearance such as occurs for vapor forms of some elements. If CLASS is not supplied, DCFPAK will select the classification of the chemical form giving the highest effective dose.

AMAD: This is a real variable representing the activity median aerodynamic diameter of the aerosol. If unspecified, a default AMAD of 1 μm is applied. If specified, DCFPAK calculates coefficients for the user's supplied AMAD. DCFPAK applies the value of AMAD to all chain members.

IPATH: This is a logical array (size 9) used to flag the type of dose coefficients that are of interest to the user. IPATH should be set to "true" to return tables of dose coefficients for all nine exposure models addressed by DCFPAK.

In addition to assembling the decay chain and the corresponding dose coefficients, DCFPAK can be used to compute the activities of the chain members as a function of time. The calculations assume a unit activity of the chain parent at time zero and no initial activity of other chain members. To make use of this capability, the following call should be inserted into the user's driver code after a call to DOSECOF:

CALL BIRCH(ISPEC, T, A, AINT)

The variables are defined as follows:

ISPEC: This input variable is the index number of the chain member of interest. ISPEC ranges from 1, the chain parent, to NSPEC, the last radioactive member of the chain.

T: This input variable is the time (d) for which the activity of chain member ISPEC is to be computed.

A: This output variable of the routine BIRCH is the activity of ISPEC at time T assuming that one unit of activity was present at T=0 for the parent and all daughter activities at that time were zero.

AINT: The output variable is the time integrated activity of ISPEC during the period of length T, assuming that one unit of activity was present at T=0 for the parent and all daughter activities were zero at T=0. The unit is days. The number of nuclear transformations of ISPEC per Bq of the parent during this period would be 8.64E04 x AINT.

To produce activities for multiple chain members and/or multiple times, the call may be looped over ISPEC and/or T. For long decay chains and short times the computed values A and AINT may exhibit numerical noise for some chain members due to loss of significance in the computations. Although these artifacts can be seen when tabulating values as a function of time, they are typically of no dosimetric consequence. An extended precision version of this routine is available from the authors but was not used in DFCPAK because its computations are considerably slower than those of the present version.

Before any calls to BIRCH, the routine DOSECOF must be called to assemble the decay chain. The user should see READEM.FOR for an illustrative implementation.

The user should examine the source code READEM.FOR for an example of how the routines of DCFPAK should be called.

DEMONSTRATION CODE

The READEM code included with this software package is an interactive code that can be used to examine available DCFPAK output. To run the code, type "readem" and press <Enter>. You will be prompted for the name of the radionuclide, the inhalation clearance class (a default is provided, as explained earlier), and the particle size (a default value of 1 μm is provided). In this demonstration code, default values are used for the other variables in the call DOSECOF. After responding to the three prompts described above, you will be shown the half-lives, branching fractions, and cumulative alpha, beta, and gamma energies of each member of the (non-truncated) decay chain for that radionuclide. Press <Enter> to see inhalation dose coefficients for the parent radionuclide, press <Enter> again to see inhalation dose coefficients for the next decay chain member in the truncated chain, and so forth. Repeatedly pressing <Enter> will take you through the dose coefficients for all members of the truncated chain, for all nine exposure modes addressed in DCFPAK.

Following the listing of the dose coefficients assembled for the chain, you will be asked to provide a time at which the activities of the chain members should be computed. If this computation is not desired, enter a time of zero; you will then be prompted to enter the next radionuclide. Entering the time (in days) will produce a tabulation of the activity and time-integrated activity for the chain members, as derived by the routine BIRCH. This computation is carried out only for the chain members that have been judged by DCFPAK to be significant in the dosimetry.

To exit READEM, press <Enter> when prompted for the next radionuclide.

INSTALLATION ON UNIX WORKSTATION

The following additional steps allow a user to run READEM on a UNIX (SUN) workstation:

1. Convert all files extracted from SNL.EXE (with the exception of convert2unix, wdaf, and wdaf.f) from DOS format to UNIX format. A batch file named convert2unix is included as an example of the necessary unix commands to convert the files.

2. Rewrite the data files as direct access files since the direct access attribute is lost in conversion from DOS to UNIX. A short FORTRAN program named wdaf (write direct access files) is included as an example of how to rewrite the files to restore the direct access attribute.

3. Replace the extension "for" on the files readem.for, dcfpak.for, and pakparm.for with extension "f".

4. Replace the variable "nargs" in readem.f with "iargc".

5. Replace "len_trim" with "lnblnk" in readem.f and dcfpak.f.

6. Compile the package as

   f77 -03 -native -libmil -o readem readem.f dcfpak.f $LIB

REFERENCES

Eckerman, K. F.; Ryman, J. C. Federal Guidance Report No. 12: External Exposure to Radionuclides in Air, Water, and Soil. EPA-402-R-93-081. Washington, DC: U.S. Environmental Protection Agency; Office of Radiation and Indoor Air; 1993.

Eckerman, K. F.; Wolbarst, A. B.; Richardson, A. C. B. Federal Guidance Report No. 11: Limiting Values of Radionuclide Intake and Air Concentration and Dose Conversion Factors for Inhalation, Submersion, and Ingestion. EPA -520/1-88-020. Washington, DC: U.S. Environmental Protection Agency, Office of Radiation Programs; 1988.

International Commission on Radiological Protection. Annals of the ICRP 1. ICRP Publication 26. Oxford: Pergamon Press; 1977.

International Commission on Radiological Protection. Annual limits on intake of radionuclides by workers based on the 1990 recommendations. ICRP Publication 60. Oxford: Pergamon Press; 1991.

International Commission on Radiological Protection. Limits for intakes by workers. ICRP Publication 30. Oxford: Pergamon Press; Part 1: 1979.

International Commission on Radiological Protection. Limits for intakes by workers. ICRP Publication 30. Oxford: Pergamon Press; Part 2: 1980.

International Commission on Radiological Protection. Limits for intakes by workers. ICRP Publication 30. Oxford: Pergamon Press; Part 3: 1981.

International Commission on Radiological Protection. Limits for intakes by workers. ICRP Publication 30. Oxford: Pergamon Press; Part 4: 1988.

APPENDIX A: Source Code Listings

A1. INCLUDE FILES

INCLUDE FILE PAKPARM.FOR

```
*  The following parameter values give the maximium dimensions of the
*  arrays in the dose factor package.  To increase the problem size edit
*  this file and recompile all routines.
*
*     mspec is the maximium number of nuclides in a chain
*     mfact is the maximium number of dose factors types
*     morg is the maxium number of organs in the dose factor file.
*
      integer mspec, morg, mfact
      parameter(mspec = 25, morg = 25, mfact = 9)
*
```

INCLUDE FILE DCFPAK.CMN

```
      character*9 organ
      character*8 thalf
      character*7 nucnam
      character*2 iu
      character*1 classo
      real branch
      double precision lmr
c     integer*4 ibr, nbr, nint, next
      logical iflag
      common /dfacts/ organ(morg), df(mspec, mfact, morg), flinh(mspec),
     :                florl(mspec), classo(mspec), iflag(mspec, mfact),
     :                nint, next
*
      common /radat/ thalf(mspec), iu(mspec), nucnam(mspec),
     :               branch(mspec, mspec), lmr(mspec), ibr(mspec,mspec),
     :               nbr(mspec), nspec
```

INCLUDE FILE BATCH.CMN

```
*     SNL batch flag
      logical dbatch
      common/dcalbat/dbatch
```

INCLUDE FILE IOLIST.CMN

```
      integer idex, ingx, inhx, i31, i32, i33, i34, i35, i36,
     :          i37, olog
      parameter (idex = 10, ingx=11, inhx=12, i31=13, i32=14,
     :           i33=15, i34=16, i35=17, i36=18, i37=19, olog=20)
```

A2. PROGRAM DCFPAK.FOR

```
*-----------------------------------------------------------------------
*
*     DCFPAK: Dose Coefficient File Package  ORNL/TM-13347
*
*     K.F. Eckerman and R.W. Leggett
*
*     Oak Ridge National Laboratory
*
*
*     The following is DCFPAK's source code.  The routines are presented
*
*     in alphabetical order by function; the order is:
*
*        1. computational subroutines,
*
*        2. screen routines,
*
*        3. functions routines.
*
*     The code was written in FORTRAN 77 using the Micrsoft FORTRAN
*
*     Compiler 5.1.  Microsoft's LEN_TRIM function has been used,
*
*     however at the end of the listing is a replacement function that
*
*     can be activated by uncommenting the statements.
*
*     K.F. Eckerman  July 1, 1996.
*
*
*     Routine call tree for DCFPAK
*
*
*
*     DOSECOF  <---------------------- Entry point to DCFPAK
*
*          CHAIN
*
*              BIRCH  <---------------- Can call after DOSECOF
*
*                  BATMAN
*
*                      EXPF1
*
*                          EXPFUN
*
*                      EXPFUN
*
*              CLS
*
*              FRWARD
*
*                  IBINRY
*
```

```
*            ICUTOFF
*            LEN_TRIM
*            ORDER
*            PATH
*            PAUSEIT
*            PRINTM
*                CLS
*            RECVER
*            TIMEST
*        CLS
*        DEPFRAC
*        IBINRY
*        LEN_TRIM
*        NEWDFS
*            SORTEM
*        ZEROM
*
*
*    Call cross-reference
*
*
*    BATMAN called by:    BIRCH
*    BIRCH called by:     CHAIN
*    CHAIN called by:     DOSECOF
*    CHECK called by:     CHEKAB      NUKEOK
*    CHEKAB called by:    NUKEOK
*    CHEKMN called by:    NUKEOK
*    CLS called by:       CHAIN       DOSECOF      PRINTM
*    DEPFRAC called by:   DOSECOF
*    EXPF1 called by:     BATMAN
*    EXPFUN called by:    BATMAN      EXPF1
*
*    FILEINI called by:   OPENEM
*    FRWARD called by:    CHAIN
*    IBINRY called by:    DOSECOF     FRWARD       READEM       NUKEOK
*    ICUTOFF called by:   CHAIN
*    LCASE called by:     FILEINI
*    LEN_TRIM called by:  CHAIN       CHEKAB       DOSECOF      FILEINI
*                         LCASE       LTRIM        CASE
*    LTRIM called by:     CHECK       TABLEM
*    NEWDFS called by:    DOSECOF
*    ORDER called by:     CHAIN
*    PATH called by:      CHAIN
*    PAUSEIT called by:   CHAIN
*    PRINTM called by:    CHAIN
*    RECVER called by:    CHAIN
*    SORTEM called by:    NEWDFS
*    TIMEST called by:    CHAIN
*    ZEROM called by:     DOSECOF
*
*    The user's code should call the following routines
*        OPENEM  <--  to open the data files
*        NUKEOK  <--  to check the the nuclide name is valid.
*        DOSECOF <--  to assemble the decay chain
*        CLOSEM  <--  to close the data files
*
*    1. Computational Routines
*-----------------------------------------------------------------------
*    block data
*-----------------------------------------------------------------------
      include 'pakparm.for'
      include 'dcfpak.cmn'
```

```
      data organ /'Adrenals ', 'Bld Wall ', 'B Surface', 'Brain    ',
     :            'Breasts  ', 'Esophagus', 'St Wall  ', 'SI Wall  ',
     :            'ULI Wall ', 'LLI Wall ', 'Kidneys  ', 'Liver    ',
     :            'Lungs    ', 'Muscle   ', 'Ovaries  ', 'Pancreas ',
     :            'R Marrow ', 'Skin     ', 'Spleen   ', 'Testes   ',
     :            'Thymus   ', 'Thyroid  ', 'Uterus   ', 'H sub E  ',
     :            '   E     ' /
      end
*-----------------------------------------------------------------------
*
  subroutine batman(b0, zk, zkt, an1, an2, t, n)
*
*-----------------------------------------------------------------------
*
*
      include 'pakparm.for'
*
*     call variables.
      double precision b0, zk, zkt, an1, an2, zero
      integer n
      dimension b0(mspec), zkt(mspec), zk(mspec)
*     local variables.
      double precision s1, s2, ss1, ss2, prod, expfun, expf1
      integer i, j, k
      parameter (zero=0.0d0)
      include 'iolist.cmn'
      an1 = zero
      an2 = zero
      do 50 i = 1, n
        if (b0(i) .ne. zero) then
           s1 = zero
           s2 = zero
           ss1 = zero
           ss2 = zero
           do 40 j = i, n
              prod = zkt(n) / zk(n) * zk(j) / zkt(i)
              do 30 k = i, n
                 if (k .ne. j) prod = prod * zk(k) / (zkt(k) - zkt(j))
30            continue
              if (prod .lt. zero) then
                 s1 = s1 + dabs(prod) * expfun(-zkt(j) * dble(t))
                 ss1 = ss1 + dabs(prod) * expf1(zkt(j), dble(t))
              else
                 s2 = s2 + prod * expfun(-zkt(j) * dble(t))
                 ss2 = ss2 + prod * expf1(zkt(j), dble(t))
              end if
40         continue
*
*          only positive values are retained; negatives are zero
*
           if (s2 .gt. s1)   an1 = an1 + b0(i) * (s2 - s1)
           if (ss2 .gt. ss1) an2 = an2 + b0(i) * (ss2 - ss1)
        end if
*
50    continue
*
      return
      end
*-----------------------------------------------------------------------
*
      subroutine birch(imem, t, rx1, rx2)
*
*-----------------------------------------------------------------------
*
      include 'pakparm.for'
      include 'dcfpak.cmn'
      integer mpath, max
      common/calcul/ max(mspec), mpath(mspec, mspec)
*     call variables.
      integer imem
*     local variables.
      integer i, j, mark, jpath, ipath, nmem, m
      double precision zkt, zk, b, b0, an1, an2, x1, x2, zerod
      dimension b(mspec), b0(mspec), zkt(mspec), zk(mspec), mark(mspec),
     :          jpath(mspec), ipath(mspec)
      parameter(zero = 0.0, zerod = 0.0d0)
*
*     Trace the pathway backwards from Imem to decide which elements
*     of the Mpath matrix to choose.
*
      rx1 = zero
      rx2 = zero
      x1 = zerod
      x2 = zerod
      do 30 i = 1, nspec
        mark(i) = 1
        b(i) = dble(branch(i, i))
30    continue
31    nmem = 1
      jpath(1) = imem
*
      if (max(imem) .eq. 0) goto 35
33      imem = mpath(mark(imem), imem)
        nmem = nmem + 1
        jpath(nmem) = imem
        if (max(imem) .gt. 0) goto 33
35    do 40 i = 1, nmem
        ipath(i) = jpath(nmem - i + 1)
40    continue
      imem = ipath(nmem)
      do 50 i = 1, nmem
         b0(i) = b(ipath(i))
         zkt(i) = lmr(ipath(i))
         if (i .lt. nmem) then
            zk(i) =  dble(branch(ipath(i), ipath(i + 1))) * zkt(i)
         else
            zk(i) = zkt(i)
         end if
50    continue
      call batman(b0, zk, zkt, an1, an2, t, nmem)
      x1 = x1 + an1
      x2 = x2 + an2
60    do 80 i = 1, nmem
        b(ipath(i)) = zerod
```

```
      if (i .gt. 1) then
        if (mark(ipath(i)) .ne. max(ipath(i))) then
          m = ipath(i)
          mark(m) = mark(m) + 1
          do 70 j = 1, m - 1
            mark(j) = 1
            b(j) = dble(branch(j, j))
 70       continue
          goto 31
        end if
      end if
 80 continue
    imem = ipath(nmem)
    rx1 = sngl(x1)
    rx2 = sngl(x2)
    return
    end
*-----------------------------------------------------------------------
*
    subroutine chain(nuke)
*
*-----------------------------------------------------------------------
*   routine:  chain
*   author:   k. f. eckerman
*   date:     04/06/89 : 09/25/91 : 08/06/92: 02/28/95 :06/04/96
*   purpose:  assemble decay chain.  the name of the parent nuclide should
*             be passed to the routine as namen(1) in the common block
*             /chains/. upon return the chain members will be contained
*             in namen, and ndau is the length of the chain.
*
    include 'pakparm.for'
    include 'dcfpak.cmn'
    include 'batch.cmn'
* common block /chaind/.
    character*7 named, nuke
    real fhold
    integer iptb, iparb, ibrch, ipar, ipt, n
    logical eob, pob
    common/chaind/named(mspec), fhold(mspec), iptb(mspec),
   :              iparb(mspec), ipt, ibrch, ipar, eob, pob
    common/energy/ealpha(mspec), ebeta(mspec), egamma(mspec)
    dimension eat(mspec), ebt(mspec), egt(mspec)
*   local variables.
    character*78 text
    character*8 t12
    character*2 t12u
    double precision zln2, timest
    parameter(zln2=0.693147181d0, zero=0.0)
    include 'iolist.cmn'
*
*   initialize chain parameters
*
    ibrch = 0
    ipar = 1
    nspec = 1
    eob = .true.
    pob = .false.
    nucnam(1) = nuke
    do 11 i = 1, mspec
       do 10  j = 1, mspec
          branch(i, j) = zero
 10    continue
 11 continue
*
*   assign one unit of activity to the parent, rest are zero
*
    branch(1, 1) = 1.0
    lmr(1) = 0.0d0
*
 20 call frward
    if (nspec .le. 0) then
      i = len_trim(nucnam(1))
      write(*,'(1x,(a),'' is not in data base!'')') nucnam(1)(:i)
      return
    endif
    call recver
    if (.not. eob) goto 20
    nspec = nspec - 1
    if (pob) call order
    do 100 i = 1, nspec
       if (nucnam(i)(:2) .eq. 'Sf') then
          lmr(i) = 0.0D0
       else
          lmr(i) = zln2 / timest(thalf(i), iu(i))
       end if
100 continue
    t12 = thalf(1)
    t12u = iu(1)
    text = nucnam(1)(:len_trim(nucnam(1))) // ' Decay Chain:'
    text = text(:len_trim(text)) // ' Half-lives and Branching'
    text = text(:len_trim(text)) // ' Fractions'
    if(.not. dbatch) then
       write(*,*) text(:len_trim(text))
       write(olog, '(/a)') text(:len_trim(text))
       call printm
       if (nspec .gt. 5) call cls
    end if
    call path
    timess = 36525.0
*
    if(.not. dbatch) then
    text = ': Activity, Transformations, & Cumulative Energies (MeV) a
   :t 100y'
    write(*,*) nucnam(1)(:len_trim(nucnam(1))), text(:len_trim(text))
    write(olog,*)nucnam(1)(:len_trim(nucnam(1))),text(:len_trim(text))
    write(*,'(''     Nuclide  T1/2        A(t)/Ao   intA/Ao(d)   Ealph
   .a    Ebeta    Egamma'')')
    write(olog,'(''     Nuclide  T1/2 .      A(t)/Ao   intA/Ao(d)   Ea
   .lpha    Ebeta    Egamma'')')
    end if
*
    ea = zero
    eb = zero
    eg = zero
```

```
      do 50 ispec = 1, nspec
        if (nucnam(ispec)(:2) .eq. 'Sf') goto 50
        call birch(ispec, timess, rx1, rx2)
        ea = ea + rx2 * ealpha(ispec)
        eb = eb + rx2 * ebeta(ispec)
        eg = eg + rx2 * egamma(ispec)
*
        if (.not. dbatch) then
           write(*,'(i4, 1x, a7, 1x, a8, a2, 1p2d12.5,3e9.2)') ispec,
     :     nucnam(ispec), thalf(ispec), iu(ispec), rx1, rx2, ea, eb, eg
           write(olog,'(i4, 1x, a7, 1x, a8, a2, 1p2d12.5,3e9.2)') ispec,
     :     nucnam(ispec), thalf(ispec), iu(ispec), rx1, rx2, ea, eb, eg
        end if
*
        eat(ispec) = ea
        ebt(ispec) = eb
        egt(ispec) = eg
   50 continue
      if (nucnam(nspec)(:2) .eq. 'Sf') then
         nint = icutoff(eat, ebt, egt, 0, nspec-1)
         next = icutoff(eat, ebt, egt, 1, nspec-1)
      else
         nint = icutoff(eat, ebt, egt, 0, nspec)
         next = icutoff(eat, ebt, egt, 1, nspec)
      end if
      if (nspec .eq. 1) then
         nint = 1
         next = 1
      end if
      if (.not. dbatch) call pauseit
      do 70 jspec = 2, nspec
         n = 0
         do 60 ispec = 1, nspec
            if (nucnam(ispec)(:2) .eq. 'Sf') goto 60
            if (branch(ispec, jspec) .gt. 1.0E-6) then
               n = n + 1
               ibr(n, jspec) = ispec
            end if
   60    continue
         nbr(jspec) = n
   70 continue
*
      return
      end
*-----------------------------------------------------------------------
*
      subroutine chekab( nuke )
*
*-----------------------------------------------------------------------
*     routine:  chekab
*     author:   M. Cristy
*     date:     05/05/93
*     purpose:  if nuclide not found, checks whether "a" & "b" isomers
*               exist
*
      character*7 nuke, nukeab, nukea, nukeb, check
      character*6 thalf1, thalf2
      parameter (nnuke = 9)
      dimension nukeab(nnuke), nukea(nnuke), nukeb(nnuke),
     :          thalf1(nnuke), thalf2(nnuke)
      data nukeab /'Eu-150 ', 'In-110 ', 'Ir-186 ', 'Nb-89  ',
     .'Np-236 ', 'Re-182 ', 'Sb-120 ', 'Sb-128 ', 'Ta-178 '/
      data nukea /'Eu-150a', 'In-110a', 'Ir-186a', 'Nb-89a ',
     .'Np-236a', 'Re-182a', 'Sb-120a', 'Sb-128a', 'Ta-178a'/
      data nukeb /'Eu-150b', 'In-110b', 'Ir-186b', 'Nb-89b ',
     .'Np-236b', 'Re-182b', 'Sb-120b', 'Sb-128b', 'Ta-178b'/
      data thalf1/'12.62h', '69.1m', '15.8h', '66m',  '115E3y',
     .'12.7h', '15.89m', '10.4m', '9.31m'/
      data thalf2/'34.2y',  '4.9h',  '1.75h', '122m', '22.5h',
     .'64.0h', '5.76d',  '9.01h', '2.2h'/
      do 10 i = 1, nnuke
        if (nuke .eq. nukeab(i)) then
          write(*,9110) nuke, nukea(i), thalf1(i),nukeb(i),thalf2(i)
  100     write(*,'(a\)')' Input nuclide or <Enter> to quit)-> '
          read(*,'(bn, a7)') nuke
          nuke = check( nuke )
          if (len_trim(nuke) .eq. 0) stop
          return
        endif
   10 continue
      return
 9110 format( 4x,'Nuclide ',a,'has 2 isomers:'/20x,a,' with halflife ',a
     . ./16x,'and ',a,' with halflife ',a/4x,
     .'Re-input entire name with appropriate "a" or "b" designation',/)
      end
*-----------------------------------------------------------------------
*
      subroutine chekmn( nuke )
*
*-----------------------------------------------------------------------
*     routine: chekmn
*     author:  M. Cristy
*     date:    05/05/93, revised 8/16/93
*     purpose: if "m" isomer is requested, checks whether "n" isomer
*              also exists.
*
      character*7 nuke, nukem, nuken
      character*6 thalfm, thalfn
      character*1 meta
      dimension nukem(3), nuken(3), thalfm(3), thalfn(3)
      data nukem /'Ir-190m', 'Sb-124m', 'Tb-156m'/
      data nuken /'Ir-190n', 'Sb-124n', 'Tb-156n'/
      data thalfm/'1.2h', '93s',   '24.4h'/
      data thalfn/'3.1h', '20.2m', '5.0h'/
      data nnuke /3/
      do 20 i = 1, nnuke
        if (nuke .eq. nukem(i)) then
          write(*, 9110) nuke, nukem(i), thalfm(i), nuken(i), thalfn(i)
   10     write(*, 9120) nukem(i), nuken(i)
          read(*,'(a1)') meta
          if (meta.eq.' ' .or. meta.eq.'m' .or. meta.eq.'M') then
              return
            elseif (meta.eq.'n' .or. meta.eq.'N') then
              nuke = nuke(1:6) // 'n'
```

```
            return
          else
            goto 10
        endif
      endif
   20 continue
      return
 9110 format(4x,'Nuclide ',a,' has 2 metastable isomers:'/20x,
     :a,' with halflife ',a/16x,'and ', a,' with halflife ',a)
 9120 format(4x,'Input <Enter> to accept ',a,', or input "n" for ',a,
     :': ',/)
      end
*-----------------------------------------------------------------------
*
      subroutine closem
*
*-----------------------------------------------------------------------
      include 'iolist.cmn'
      close(idex)
      close(ingx)
      close(inhx)
      close(i31)
      close(i32)
      close(i33)
      close(i34)
      close(i35)
      close(i36)
      close(i37)
      close(olog)
      return
      end
*-----------------------------------------------------------------------
*
      subroutine depfrac (amad, d3, d4, d5)
*
*-----------------------------------------------------------------------
*
*     author:   k. f. eckerman
*     date:     08/20/93
*     purpose:  routine computes the deposition fractions in the three
*               regions of the lung: nasal-pharynx (d3), tracheo-bronchial
*               (d4) and pulmonary (d5).
*
      p(t) = 1.0 - 0.5*((((0.019527*t + 0.000344)*t + 0.115194)*t +
     :                    0.196854)*t + 1.0)**(-4)
      ppt(w) = -(w - ((0.010328*w + 0.802853)*w + 2.515517) /
     :              (((0.001308*w + 0.189269)*w + 1.432788)*w + 1.0))
      ww(x) = sqrt(-2.0*alog(x))
*
*     Nasal-pharynx region
      if (amad .eq. 0.0) then
         d3 = 0.30
      elseif (amad .le. 0.1) then
         d3 = 0.1
      elseif (amad .ge. 20) then
         d3 = 0.95
      elseif (amad .gt. 0.1 .and. amad .lt. 0.2) then
         u0 = -0.698970
         u1 = -1.00
         t0 = ppt(ww(0.05))
         t1 = ppt(ww(0.10))
         t = t0 + (t1 - t0) / (u1 - u0) * (alog10(amad) - u0)
         if (t .ge. 0.0) then
            d3 = p(t)
         else
            d3 = 1.0 - p(-t)
         end if
      elseif (amad .ge. 0.2 .and. amad .le. 10.0) then
         u0 = -0.698970
         u1 =  0.301030
         t0 = ppt(ww(0.05))
         t1 = ppt(ww(0.5))
         t = t0 + (t1 - t0) / (u1 - u0) * (alog10(amad) - u0)
         if (t .ge. 0.0) then
            d3 = p(t)
         else
            d3 = 1.0 - p(-t)
         end if
      elseif (amad .gt. 10.0 .and. amad .lt. 20.0) then
         u0 =  1.0
         u1 =  1.301030
         t0 = ppt(ww(0.875))
         t1 = ppt(ww(0.95))
         t = t0 + (t1 - t0) / (u1 - u0) * (alog10(amad) - u0)
         if (t .ge. 0.0) then
            d3 = p(t)
         else
            d3 = 1.0 - p(-t)
         end if
      end if
*     TB region
      if (amad .eq. 0.0) then
         d4 = 0.08
      elseif (amad .le. 0.1) then
         d4 = 0.17
      elseif (amad .ge. 20) then
         d4 = 0.035
      elseif (amad .gt. 0.1 .and. amad .lt. 0.2) then
         u0 = -0.698970
         u1 = -1.0
         t0 = ppt(ww(0.08))
         t1 = ppt(ww(0.17))
         t = t0 + (t1 - t0) / (u1 - u0) * (alog10(amad) - u0)
         if (t .ge. 0.0) then
            d4 = p(t)
         else
            d4 = 1.0 - p(-t)
         end if
      elseif (amad .ge. 0.2 .and. amad .le. 10.0) then
         d4 = 0.08
      elseif (amad .gt. 10.0 .and. amad .lt. 20.0) then
         u0 = 1.0
         u1 = 1.30130
         t0 = ppt(ww(0.08))
```

```
            t1 = ppt(ww(0.035))
            t = t0 + (t1 - t0) / (u1 - u0) * (alog10(amad) - u0)
            if (t .ge. 0.0) then
               d4 = p(t)
            else
               d4 = 1.0 - p(-t)
            end if
         end if
*        Pulmonary region
         if (amad .eq. 0.0) then
            d5 = 0.25
         elseif (amad .le. 0.1) then
            d5 = 0.6150
         elseif (amad .ge. 20.0) then
            d5 = 0.015
         elseif (amad .gt. 0.1 .and. amad .lt. 0.2) then
            u0 = -0.698970
            u1 = -1.00
            t0 = ppt(ww(0.5))
            t1 = ppt(ww(0.615))
            t = t0 + (t1 - t0) / (u1 - u0) * (alog10(amad) - u0)
            if (t .ge. 0.0) then
               d5 = p(t)
            else
               d5 = 1.0 - p(-t)
            end if
         elseif (amad .ge. 0.2 .and. amad .le. 10.0) then
            u0 = -0.698970
            u1 = 1.0
            t0 = ppt(ww(0.5))
            t1 = ppt(ww(0.05))
            t = t0 + (t1 - t0) / (u1 - u0) * (alog10(amad) - u0)
            if (t .ge. 0.0) then
               d5 = p(t)
            else
               d5 = 1.0 - p(-t)
            end if
         elseif (amad .gt. 10.0 .and. amad .lt. 20.0) then
            u0 = 1.0
            u1 = 1.301030
            t0 = ppt(ww(0.05))
            t1 = ppt(ww(0.015))
            t = t0 + (t1 - t0) / (u1 - u0) * (alog10(amad) - u0)
            if (t .ge. 0.0) then
               d5 = p(t)
            else
               d5 = 1.0 - p(-t)
            end if
         end if
   50    continue
         return
         end
```

```
*-----------------------------------------------------------------------
*
      subroutine dosecof(nuke, flo, class, fli, amad, ipath)
*
*-----------------------------------------------------------------------
*     routine:  dosecof
*     author:   k.f. eckerman
*     data:     05/20/96
*     purpose:  assemble the decay chain and extract the dose coefficients
*               for the chain members.
*-----------------------------------------------------------------------
*     input variables
*       nuke       parent of the chain in standard notation; e.g., Cs-137
*       flo        f_1 value for oral intakes (defaults to highest effective
*                  dose
*       class      inhalation clearance class for parent.  decay products
*                  will be of this class if possible or class with highest
*                  effective dose.
*       fli        f_1 value for inhalation.  coefficients are actually
*                  picked by class.
*       amad       amad for the aerosol in μm.
*       ipath(i)   pathway logical flag of size 9. 1 inhalation,
*                  2 ingestion, 3 submersion, 4 immersion, 5 ground surface,
*                  6 1 cm thick soil sample, 7 5 cm, 8 15 cm, 9 infinite.
*
*     output
*     output is through the common blocks in the include file dcfpak.cmn
*     as shown below.
*      common /dfacts/ organ(morg), df(mspec, mfact, morg), flinh(mspec),
*     :                florl(mspec), classo(mspec), iflag(mspec, mfact),
*     :                nint, next
*       organ      character*9 array of organ names.
*       df         dose factors array by chain member, pathway, and organ
*       flinh      array of f_1 values for inhalation by chain member
*       florl      array of f_1 values for ingestion by chain member
*       classo     array of class notation for inhalation by chain member
*       iflag      logical flags for dose factor by chain member and pathway
*       nint       length of chain for internal factors
*       next       length of chain for external factors
*
*      common /radat/ thalf(mspec), iu(mspec), nucnam(mspec),
*     :               branch(mspec, mspec), lmr(mspec),
*     :               ibr(mspec,mspec), nbr(mspec), nspec
*       thalf      half-life of chain members
*       iu         units of the half-lives
*       nucnam     names of the chain members
*       branch     branching fraction
*       lmr        decay constant in 1/d
*       ibr        branch pointer
*       nbr        number of branches for chain members
*       nspec      length of chain
*
      include 'pakparm.for'
      include 'dcfpak.cmn'
      include 'batch.cmn'
      character*7 nuke, nuclide
      character*1 clshld, class, cx
```

```
      integer ipt, ibinry, idep
      logical ipath(9)
      dimension dx(4, 25), f1hld(4), clshld(4), idep(4, 25, 2), ifi(7)
      include 'iolist.cmn'
      data ifi /i31, i32, i33, i34, i35, i36, i37/
      if (.not. dbatch) call cls
*
*     zero out the dose factor array
*
      call zerom
*
*     assemble the decay chain
*
      call chain (nuke)
*
*     first do the external coefficients for the next chain members
*
      do 20 ispec = 1, next
         nuke = nucnam(ispec)
         ipt = ibinry( nuke )
         if (ipt .ne. 0) then
            read(idex,'(a7,a8,a2,a6,3i5)', rec=ipt) nuke, t, ix, mode,
     :           inh, ing, iex
            if (iex .ne. 0) then
               do 10 ip = 3, 9
                 if (ipath(ip)) then
                    iflag(ispec,ip) = .true.
                    read(ifi(ip-2),'(a7, 1p25e9.0)', rec = iex)
     :                 nuclide, (df(ispec,ip, j), j = 1, 25)
                 end if
   10          continue
            end if
         end if
   20 continue
*
*     now do the inhalation and ingestion coefficients
*
      if (amad .ne. 1.0) call depfrac (amad, d3, d4, d5)
      do 200 ispec = 1, nint
         nuke = nucnam(ispec)
         ipt = ibinry( nuke )
         read(idex,'(a7,a8,a2,a6,3i5)', rec=ipt) nuke, t, ix, mode, inh,
     :        ing, iex
c
c        inhalation
c
         if (ipath(1).and. inh .ne. 0) then
           imax = 0
           dmax = 0.0
           irec = inh - 1
           iflag(ispec, 1) = .true.
           do 50 itry = 1, 4
              irec = irec + 1
              read(inhx,'(a7, 1x, a1, e8.0, 25e9.0)', rec=irec, end=60)
     :                    nuclide, cx, f1hld(itry),
     :                    (dx(itry, j), j = 1, 25)
*
*          if we have a particulate factor then read the next record
*          to get the contribution of the deposition in the NP and P
*          region to the organ dose.
*
           if (cx .eq. 'D' .or. cx .eq. 'W' .or. cx .eq. 'Y') then
              irec = irec + 1
              read(inhx, '(17x, 25(i5, i4))', rec = irec)
     :             (idep(itry, j, 1), idep(itry, j, 2), j = 1, 25)
           end if
           clshld(itry) = cx
           if (len_trim(nuclide) .gt. 0 .and. nuclide .ne. nuke) then
              goto 60
           elseif (len_trim(nuclide).eq.0 .or. nuclide.eq.nuke) then
              if (class .eq. clshld(itry) .and.
     :                    fii .eq. f1hld(itry)) then
*
*               calculate the factors for this amad if <> 1 µm
*
                if (amad.ne.1.0) call newdfs (itry,d3,d4,d5,dx,idep)
*
                do 30 j = 1, 25
                   df(ispec, 1, j) = dx(itry, j)
   30           continue
                f1inh(ispec) = f1hld(itry)
                classo(ispec) = clshld(itry)
                goto 80
              elseif (class .eq. clshld(itry)) then
*
*               calculate the factors for this amad if <> 1 µm
*
                if (amad.ne.1.0) call newdfs (itry,d3,d4,d5,dx,idep)
*
                do 40 j = 1, 25
                   df(ispec, 1, j) = dx(itry, j)
   40           continue
                f1inh(ispec) = f1hld(itry)
                classo(ispec) = clshld(itry)
                goto 80
              else
                if (dx(itry, 25) .gt. dmax) then
                   dmax = dx(itry, 25)
                   imax = itry
                end if
              end if
           else
             goto 60
           end if
   50      continue
*
*          calculate the factors for this amad if <> 1 µm
*
   60      if (amad.ne.1.0) call newdfs (imax, d3, d4, d5, dx, idep)
*
           do 70 j = 1, 25
              df(ispec, 1, j) = dx(imax, j)
   70      continue
           f1inh(ispec) = f1hld(imax)
```

```
          classo(ispec) = clshld(imax)
        end if
   80   continue
c
c       ingestion
c
        if (ipath(2) .and. ing .ne. 0) then
          imax = 0
          dmax = 0.0
          irec = ing - 1
          iflag(ispec, 2) = .true.
          do 100 itry = 1, 4
             irec = irec + 1
             read(ingx,'(a7, e8.0, 25e9.0)', rec = irec, end = 110)
     :                    nuclide, flhld(itry), (dx(itry, j), j = 1, 25)
             if (len_trim(nuclide) .gt. 0 .and. nuclide .ne. nuke) then
                goto 110
             elseif (len_trim(nuclide).eq.0 .or. nuclide.eq.nuke) then
              if (flo .eq. flhld(itry)) then
                  do 90 j = 1, 25
                     df(ispec, 2, j) = dx(itry, j)
   90             continue
                  fiorl(ispec) = flo
                  goto 130
              else
                  if (dx(itry, 25) .ge. dmax) then
                     dmax = dx(itry, 25)
                     imax = itry
                  end if
              end if
             else
              goto 110
             end if
  100      continue
  110      do 120 j = 1, 25
             df(ispec, 2, j) = dx(imax, j)
  120      continue
           fiorl(ispec) = flhld(imax)
        end if
  130   continue
  200 continue
      return
      end
*-----------------------------------------------------------------------
*
      subroutine fileini(fnpath, target, nlen, nulog, nuini, program)
*
*-----------------------------------------------------------------------
*
*  Author: K. F. Eckerman
*  Opens file 'program'.ini and finds the full file name, including path,
*  for the file target and its record length nlen.  See the ini file for
*  additional comments.
*  target = file name passed to this subprogram
*  fnstd = standard file name read from file 'program'.INI
*  fnpath = file name including path
*
      character*(*) fnpath, target, program
      character*12 fnini, fnstd, lcase
      integer nulog
      logical test
      parameter (maxfil = 40)
      fnini = program(1:len_trim(program)) // '.ini'
      inquire (file = fnini, exist = test)
      if (.not. test) goto 20
      open(nuIni, file=fnini, status='old')
      target = lcase(target)
      do 10 i = 1, maxfil
        read(nuIni, *, end=15) fnstd, fnpath, nlen
        if (fnstd(:3) .eq. 'EOF') goto 15
        if(lcase(fnstd) .eq. target) goto 20
   10 continue

   15 write(nuLog,9110)target(:len_trim(target)),fnini(:len_trim(fnini))
      write( *   ,9110)target(:len_trim(target)),fnini(:len_trim(fnini))
      close(nuIni)
      stop 1
   20 close(nuIni)
      return
 9110 format(' **** FATAL ERROR in function FileIni: Unable to find the
     :file ', a, ';'/6x,'check ',a,' for proper assignments ****')
      end
*
*-----------------------------------------------------------------------
*                                                                      *
      subroutine frward
*                                                                      *
*-----------------------------------------------------------------------
*  routine:  frward
*  author:   k. f. eckerman
*  date:     01/14/92
*  purpose:  read down a branch of a decay chain.
*
      include 'pakparm.for'
      include 'dcfpak.cmn'
      character*7 named
      real fhold
      integer iptb, iparb, ibrch, ipar, ipt
      logical eob, pob
      common/chaind/named(mspec), fhold(mspec), iptb(mspec),
     :              iparb(mspec), ipt, ibrch, ipar, eob, pob
      common/energy/ealpha(mspec), ebeta(mspec), egamma(mspec)
*
*     functions referenced.
*
      integer ibinry
      character*8 t, mode
      character*7 nuke, dl
      character*2 ix
      integer j
      include 'iolist.cmn'
*
*     get parent record.
*
```

```
      if (ipar .eq. 1) then
         nuke = nucnam(ipar)
         ipt = ibinry(nuke)
         if (ipt .eq. 0) then
            nspec = 0
            return
         endif
      endif
   10 if (ipt .lt. 999) then
         read(idex,'(a7,a8,a2,a6,3i5,3(i4,e11.0),3f7.0,e11.0,1x,a9)',
     :      rec=ipt) nuke, t, ix, mode, inh, ing, iex, id1, f1,
     :      id2, f2, id3, f3, ea, eb, eg, amu, endsf
      else
        id1 = 0
        f1 = 0.0
        id2 = 0
        f2 = 0.0
        id3 = 0
        f3 = 0.0
        nuke = 'sf'
        ea = 0.
        eb = 0.
        eg = 0.
        t = ' '
        ix = ' '
      end if
*
*     ids = 999 denotes "sf" which is not a daughter product, thus set
*     the ids to zero if "sf".
*
      if (id1 .eq. 999 .and. id2 .ne. 0) then
         if (id3 .eq. 0) then
            fhld = f1
            ihld = id1
            f1 = f2
            id1 = id2
            f2 = fhld
            id2 = ihld
         else
            fhld = f1
            ihld = id1
            f1 = f2
            id1 = id2
            f2 = f3
            id2 = id3
            f3 = fhld
            id3 = ihld
         end if
      end if
      if (id2 .eq. 999 .and. id3 .ne. 0) then
         ihld = id2
         fhld = f2
         id2 = id3
         f2 = f3
         id3 = ihld
         f3 = fhld
      end if
*
*     if processing a branch then check to see if d1 has already
*     been included in nucnam, if so only fix up branch(ipar,past) and
*     terminate chain, i.e., chain has converged.
*
      if (pob) then
         if (id1 .gt. 0 .and. id1 .lt. 999) then
            read(idex, 50, rec = id1) d1
         elseif (id1 .eq. 999) then
            d1 = 'Sf'
         end if
         do 15 j = 1, nspec - 1
            if (d1 .eq. nucnam(j)) goto 16
   15    continue
         goto 17
*
*        have already handled this daughter; chain has converged.
*        set end of chain and return.
*
   16    branch(ipar, j) = f1
         nucnam(ipar) = nuke
         thalf(ipar) = t
         ealpha(ipar) = ea
         ebeta(ipar) = eb
         egamma(ipar) = eg
         iu(ipar) = ix
         nspec = nspec + 1
         return
      end if
*
*     need to treat this chain member.
*
   17 nucnam(ipar) = nuke
      thalf(ipar) = t
      ealpha(ipar) = ea
      ebeta(ipar) = eb
      egamma(ipar) = eg
      iu(ipar) = ix
      nspec = nspec + 1
      branch(ipar, nspec) = f1
*
*     no further daughters in chain - set end of chain
*
      if (id1 .ne. 0) then
*       further daughters, treat id1 and check for possible branches.
        ipt = id1
*
        if (id2 .ne. 0 ) then
*
*         set end of branch to false, increment branch counter, store
*         pointer of parent, and record number of second or third daughter
*         while following current chain.  routine recver will direct
*         recovery of branches.
*
          eob = .false.
          ibrch = ibrch + 1
          iptb(ibrch) = id2
```

```
          fhold(ibrch) = f2
          iparb(ibrch) = ipar
          if (id2 .ne. 999) then
             read(idex, 50, rec = id2) named(ibrch)
          else
             named(ibrch) ='Sf'
          end if
        endif
*       third daughter, branch info held as above.
        if (id3 .ne. 0) then
          eob = .false.
          ibrch = ibrch + 1
          iptb(ibrch) = id3
          fhold(ibrch) = f3
          iparb(ibrch) = ipar
          if (id3 .ne. 999) then
             read(idex, 50, rec = id3) named(ibrch)
          else
             named(ibrch) = 'Sf'
          end if
        endif
        ipar = nspec
        if (ipt .ne. 999) goto 10
      endif
      return
*
   50 format (a7,a8,a2,a8,i7,i5,i6,i4,3(i4,e11.0),f7.0,2f8.0,a10)
      end
*-----------------------------------------------------------------------
*
      subroutine newdfs (i, d3, d4, d5, dx, idep)
*
*-----------------------------------------------------------------------
*     routine:  newdfs
*     author:   k.f. eckerman
*     date:     10/23/93
*     purpose:  compute h_e and e for d3, d4, d5 depositions
*
      dimension dx(4, 25), idep(4, 25, 2), dsort(12), isort(12)
      data isort / 1, 4, 7, 8, 9, 10, 11, 12, 16, 19, 21, 23/
*
*     calculate the inhalation coefficient for this new depositions
*
      do 10 j = 1, 23
         f = d3 * float(idep(i, j, 1)) / 0.30 +
     :       d5 * float(idep(i, j, 2)) / 0.25 +
     :       d4 * (float(100 - idep(i,j,1) - idep(i,j,2))) / 0.08
         dx(i, j) = 0.01 * f * dx(i, j)
   10 continue
*
*     compute the effective dose equivalent (weighting factors of icrp-26)
*     and the effective dose (weighting factors of icrp-60) for the new
*     depostions.
*
      he = 0.25 * max(dx(i, 15), dx(i, 20)) + 0.15 * dx(i, 5) +
     :     0.12 * dx(i, 13) + 0.12 * dx(i, 17) + 0.03 * dx(i, 22) +
     :     0.03 * dx(i, 3)
      do 20 j = 1, 12
         dsort(j) = dx(i, isort(j))
   20 continue
      call sortem (dsort, 12)
      he = he + 0.06 * (dsort(12) + dsort(11) + dsort(10) + dsort(9) +
     :     dsort (8))
      e = 0.20 * max(dx(i,15), dx(i,20)) + 0.05 * dx(i,5) +
     :    0.12 * dx(i,10) + 0.12 * dx(i,17) + 0.12 * dx(i, 13) +
     :    0.12 * dx(i,7) + 0.05 * dx(i,2) + 0.05 * dx(i, 12) +
     :    0.05 * dx(i,6) + 0.05 * dx(i,22) + 0.01 * dx(i, 3) +
     :    0.01 * dx(i,18)
      hr = (14. * dx(i,1) + 1400. * dx(i, 4) + 640. * dx(i, 8) +
     :     310. * dx(i,11) +28000. * dx(i, 14) + 100. * dx(i,16) +
     :     180. * dx(i,19) + 20. * dx(i, 21) + 80. * dx(i,23)) /
     :     30744.
      e = e + 0.05 * hr
      dx(i, 24) = he
      dx(i, 25) = e
      return
      end
*
*-----------------------------------------------------------------------
*
      subroutine openem
*
*-----------------------------------------------------------------------
*     routine:  openem
*     author:   k.f. eckerman
*     date:     10/23/93
*     purpose:  open index and dose coefficient files.
*
      character*64 fpath
      character*12 target
      character*8 prog
      include 'iolist.cmn'
      open(olog, file = 'df_read.log')
      prog = 'dcfpak'
      target = 'snlindex.ndx'
      call fileini(fpath, target, nlen, olog, 41, prog)
      open(idex, file=fpath, access='direct', recl=nlen, form=
     :    'formatted', status='old')
      target = 'ingestsf.dfs'
      call fileini(fpath, target, nlen, olog, 41, prog)
      open(ingx, file=fpath, access='direct', recl=nlen, form=
     :    'formatted', status='old')
      target = 'inhalesf.dfs'
      call fileini(fpath, target, nlen, olog, 41, prog)
      open(inhx, file=fpath, access='direct', recl=nlen, form=
     :    'formatted', status='old')
      target = 'submrsin.dfs'
      call fileini(fpath, target, nlen, olog, 41, prog)
      open(i31, file=fpath, access='direct', recl=nlen, form=
     :    'formatted', status='old')
      target = 'imersion.dfs'
      call fileini(fpath, target, nlen, olog, 41, prog)
      open(i32, file=fpath, access='direct', recl=nlen, form=
     :    'formatted', status='old')
```

```
      target='grsurf00.dfs'
      call fileini(fpath, target, nlen, olog, 41, prog)
      open(i33, file=fpath, access='direct', recl=nlen, form=
     :    'formatted', status='old')
      target = 'grvol_01.dfs'
      call fileini(fpath, target, nlen, olog, 41, prog)
      open(i34, file=fpath, access='direct', recl=nlen, form=
     :    'formatted', status='old')
      target = 'grvol_05.dfs'
      call fileini(fpath, target, nlen, olog, 41, prog)
      open(i35, file=fpath, access='direct', recl=nlen, form=
     :    'formatted', status='old')
      target = 'grvol_15.dfs'
      call fileini(fpath, target, nlen, olog, 41, prog)
      open(i36, file=fpath, access='direct', recl=nlen, form=
     :    'formatted', status='old')
      target = 'grvolinf.dfs'
      call fileini(fpath, target, nlen, olog, 41, prog)
      open(i37, file=fpath, access='direct', recl=nlen, form=
     :    'formatted', status='old')
      return
      end
*-----------------------------------------------------------------------
*
      subroutine order
*
*-----------------------------------------------------------------------
*
*   routine:  order
*   author:   k. f. eckerman
*   date:     04/06/89
*   purpose:  order the chain so daughter index > parents.
*
      include 'pakparm.for'
      include 'dcfpak.cmn'
      common/energy/ealpha(mspec), ebeta(mspec), egamma(mspec)
      character*8 thold
      character*7 nuke
      character*2 ix
      real rsave, csave
      integer i, j, ip, jp, ipass, move
      dimension rsave(mspec), csave(mspec)
      include 'iolist.cmn'
*
*     move # of elements to move
*
      ipass = 0
  100 move = 0
      ipass = ipass + 1
      if (ipass .gt. 4*nspec) then
         write(olog,'('' Failure in routine order: greater than'',i3,
     :    '' passes for '', a7, ''.'')') ipass, nucnam(1)
         write(*,'('' Failure in routine order: greater than'',i3,
     :    '' passes for '', a7, ''.'')') ipass, nucnam(1)
         stop 1
      endif
*
      do 10 i = 1, nspec
         do 10 j = 1, i-1
            if (branch(i, j) .ne. 0.) then
               ip = i
               jp = j
               move = 1
               go to 15
            endif
   10 continue
*
*     if no elements to move then return
*
   15 if (move .eq. 0) return
      nuke = nucnam(ip)
      thold = thalf(ip)
      ea = ealpha(ip)
      eb = ebeta(ip)
      eg = egamma(ip)
      ix = iu(ip)
      do 20 j = 1, nspec
         rsave(j) = branch(ip, j)
   20 continue
      do 30 i = ip - 1, jp, -1
         nucnam(i + 1) = nucnam(i)
         thalf(i + 1) = thalf(i)
         ealpha(i + 1) = ealpha(i)
         ebeta(i + 1) = ebeta(i)
         egamma(i + 1) = egamma(i)
         iu(i + 1) = iu(i)
         do 30 j = 1, nspec
            branch(i + 1, j) = branch(i, j)
   30 continue
      nucnam(jp) = nuke
      thalf(jp) = thold
      iu(jp) = ix
      ealpha(jp) = ea
      ebeta(jp) = eb
      egamma(jp) = eg
      do 40 j = 1, nspec
         branch(jp, j) = rsave(j)
   40 continue
      do 50 i = 1, nspec
         csave(i) = branch(i, ip)
   50 continue
      do 60 j = ip - 1, jp, -1
      do 60 i = 1, nspec
         branch(i, j + 1) = branch(i, j)
   60 continue
      do 70 i = 1, nspec
         branch(i, jp) = csave(i)
   70 continue
*
*     Yes we do have backward point gotos in FORTRAN, sorry!
*
      goto 100
*
      end
```

```
*-----------------------------------------------------------------------
*
      subroutine path
*
*-----------------------------------------------------------------------
*
*     author:  K.F. Eckerman
*     date:    04/06/89
*     purpose: initialize mpath and max matrices
*              Adopted from A. Birchall, Health Phys. 50, 3, 389-397, 1986.
*
      include 'pakparm.for'
      include 'dcfpak.cmn'
      integer max, mpath
      common/calcul/ max(mspec), mpath(mspec, mspec)
*     initializes mpath and max matrics.
      do 10 i = 1, nspec
        max(i) = 0
        do 10 j = 1, nspec
        mpath(i,j) = 0
   10 continue
      do 20 j = 2, nspec
      do 20 i = 1, j - 1
        if (branch(i, j) .ne. 0.) then
          max(j) = max(j) + 1
          mpath(max(j), j) = i
        end if
   20 continue
      return
      end
*-----------------------------------------------------------------------
*
      subroutine pauseit
*
*-----------------------------------------------------------------------
*     routine: pause
*     author:  K.F. Eckerman
*     date:    10/23/93
*     purpose: pause w/o line feed
*
      character*1 dumy
      write(*,'(a\)')' Hit <Enter> to continue.'
      read(*,'(bn, a1)') dumy
      return
      end
```

```
*-----------------------------------------------------------------------
*
      subroutine printm
*                                                                      *
*----------------------------------------------------------------------*
*     routine:  printm
*     author:   k. f. eckerman with modifications by j. c. ryman
*     date:     01/15/92
*     purpose:  print the decay chain
*
      include 'pakparm.for'
      include 'dcfpak.cmn'
*
*     local variables.
      character*7 nuke, rlist
      integer i, j, k, nlist, jlist
      dimension jlist(mspec), nuke(mspec), rlist(mspec)
      include 'iolist.cmn'
      write(*, 8001)
      write(olog, 8001)
      if (nspec .eq. 1) then
         write(*, 8002) nspec, nucnam(1), thalf(1), iu(1)
         write(olog, 8002) nspec, nucnam(1), thalf(1), iu(1)
      else
         do 20 i = 1, nspec
            if(nucnam(i)(:2) .eq. 'Sf') goto 20
            nlist = 0
            do 10 j = 1, nspec
               if (i .ne. j .and. branch(i,j) .ne. 0.0) then
                  nlist = nlist + 1
                  jlist(nlist) = j
                  nuke(nlist) = nucnam(j)
                  write (rlist(nlist), '(1pe7.1)') branch(i,j)
                  do 5 k = 4, 6
                     rlist(nlist)(k:k) = rlist(nlist)(k+1:k+1)
    5             continue
                  rlist(nlist)(7:7) = ' '
               endif
   10       continue
            write(*, 8002) i, nucnam(i), thalf(i), iu(i),
     :                     (rlist(j), jlist(j), nuke(j), j = 1, nlist)
            write(olog, 8002) i, nucnam(i), thalf(i), iu(i),
     :                     (rlist(j), jlist(j), nuke(j), j = 1, nlist)

   20    continue
         if (nspec .gt. 5) pause
      endif
*
*     clear screen and return
*
      if (nspec .gt. 5) call cls
      return
*
*     formats
*
 8001 format(' ',3x,'Nuclide  Halflife    f1',7x,'Nuclide   f2',7x,
     : 'Nuclide   f3',7x,'Nuclide')
```

```
 8002 format(' ',i2,1x,a7,1x,a8,a2,:,3(1x,a6,'->',i2,1x,a7))
      end
*-----------------------------------------------------------------------
*
      subroutine recver
*
*-----------------------------------------------------------------------
*   routine:  recver
*   author:   k. f. eckerman
*   date:     04/06/89
*   purpose:  recover info on branches in the chain that were detected
*             by frward and direct the reading of the new branch.
*
      include 'pakparm.for'
      include 'dcfpak.cmn'
      character*7 named
      real fhold
      integer iptb, iparb, ibrch, ipar, ipt
      logical eob, pob
      common/chaind/named(mspec), fhold(mspec), iptb(mspec),
     :              iparb(mspec), ipt, ibrch, ipar, eob, pob
* local variables.
      character*7 nuke
      integer i
      include 'iolist.cmn'
*
*     no branches to treat, set end of branch to true and return.
*
    1 if (ibrch .eq. 0) then
         eob = .true.
*      elseif (iptb(ibrch) .eq. 999) then
*         eob = .true.
      else
*
*        consider remaining branches. recover parent's
*        index at branch (ipar) and daughter's record number (ipt).
*        decrement branch counter and return.
*
         pob = .true.
         ipar = iparb(ibrch)
         ipt = iptb(ibrch)
         nuke = named(ibrch)
*
*        need to check to see of the daughter of the branch has already
*        a member of the chain.
*
         do 10 i = 1, nspec - 1
            if (nuke .eq. nucnam(i)) goto 15
   10    continue
         nucnam(nspec) = nuke
         branch(ipar, nspec) = fhold(ibrch)
         ibrch = ibrch - 1
         ipar = nspec
         return
*
*        if already a chain member set r, decrement the branch counter
*        and look for another branch to process.
*
   15    branch(ipar,i) = fhold(ibrch)
*
         ibrch = ibrch - 1
         go to 1
*
      endif
*
      return
      end
*-----------------------------------------------------------------------
*
      subroutine sortem (ar, n)
*
*-----------------------------------------------------------------------
*     routine:  sortem
*     author:   k.f. eckerman
*     date:     10/23/93
*     purpose:  sort array a into descending numerical order
*
      dimension ar(*)
      do 20 j = 2, n
         a = ar(j)
         do 11 i = j-1, 1, -1
            if(ar(i) .le. a) goto 12
            ar(i+1) = ar(i)
   11    continue
         i = 0
   12    ar(i+1) = a
   20 continue
      return
      end
*-----------------------------------------------------------------------
*
      subroutine nukeok (nuke, ok)
*
*-----------------------------------------------------------------------
      character*7 nuke, check
      logical ok
*
*     function check ensure proper format of user input
*
      nuke = check( nuke )
      call chekmn( nuke )
*
*     find the nuclide in the index file
*
      ipt = ibinry( nuke )
      if (ipt .eq. 0) then
         call chekab( nuke )
         ipt = ibinry( nuke )
      end if
*
*     if pointer ipt is zero, the nuclide is not in the index file
*     else we have a valid nuclide to process
*
      if (ipt .eq. 0) then
```

```
         ok = .false.
      else
         ok = .true.
      end if
      return
      end
*-----------------------------------------------------------------------
*
      subroutine zerom
*
*-----------------------------------------------------------------------
*     routine:  zerom
*     author:   k.f. eckerman
*     date:     10/23/93
*     purpose:  set the dose coefficient arrays to zero
*
      include 'pakparm.for'
      include 'dcfpak.cmn'
      logical ifalse
      dimension dzero(morg*mspec*mfact), ifalse(mspec*mfact)
      equivalence (df(1,1,1), dzero(1)), (iflag(1,1), ifalse(1))
      parameter(zero = 0.0)
      do 10 i= 1, morg * mspec * mfact
         dzero(i) = zero
   10 continue
      do 20 i = 1, mspec * mfact
         ifalse(i) = .false.
   20 continue
      return
      end
*-----------------------------------------------------------------------
*
*     2. Screen Routines
*
*-----------------------------------------------------------------------
*
      subroutine cls
*
*-----------------------------------------------------------------------
*
*     author:   k. f. eckerman
*     date:     12/08/93
*     routine to clear screen
      write(*,*) char(27),'[2J'
      return
      end
*
*-----------------------------------------------------------------------
*
       subroutine curright(icol)
*
*-----------------------------------------------------------------------
*
*     author:   k. f. eckerman
*     date:     12/08/93
*     move the cursor icol columns right on the display.
*
      character*2  col
      write(col, '(i2.2)') icol
      write(*,'(a\)') ' ' // char(27) // '[' // col //'C'
      return
      end
*-----------------------------------------------------------------------
*
       subroutine curpos(irow, icol)
*
*-----------------------------------------------------------------------
*
*     author:   k. f. eckerman
*     date:     12/08/93
*     move cursor to the indicated row and column.
*
      character*2  row, col
      write(row, '(i2.2)') irow
      if (icol .ne. 0) then
         write(col, '(i2.2)') icol
      else
         col ='01'
      end if
      write(*,'(a\)') ' ' // char(27) // '[' // row // ';' // col // 'H'
      return
      end
*-----------------------------------------------------------------------
*
*     3. Function Routines
*
*-----------------------------------------------------------------------
*
      character*(*) function check(nuke)
*
*-----------------------------------------------------------------------
*     function: check
*     author:   r.j. westfall
*     date:     07/20/89
*     purpose:  convert chemical symbol in nuclide name to proper
*               notation, e.g.; Kr-85m, etc.
*
      character*(*) nuke
      character*7 ltrim
*
*     remove any leading blanks from nuke
*
      nuke = ltrim( nuke )
*
*     ensure first character is upper case.
*
      if (nuke(:1) .ge. 'a' .and. nuke(:1) .le. 'z')
     :    nuke = char(ichar(nuke(:1)) - 32) // nuke(2:7)
*
*     ensure second character, if present, is lower case.
*
      if (nuke(2:2) .ge. 'A' .and. nuke(2:2) .le. 'Z')
     :    nuke = nuke(:1) // char(ichar(nuke(2:2)) + 32) // nuke(3:7)
*
```

```
*     ensure metastable notation, if present, is lower case.
*
      do 30 j = 4, 7
         if (nuke(j:j) .ge. 'A' .and. nuke(j:j) .le. 'Z')
     :   nuke = nuke(:j-1) // char(ichar(nuke(j:j)) + 32) // nuke(j+1:)
   30 continue
      check = nuke
      end
*-----------------------------------------------------------------------
*
      double precision function expf1 (lm, t)
*
*-----------------------------------------------------------------------
*
*     author:   k. f. eckerman
*     date:     10/04/94
*     purpose:  routine to compute [1.0 - exp(-lm * t)] / lm.
*
      double precision lm, t, lmt, one, two, expfun, eps
      logical first
      parameter (one = 1.0d0, two = 2.0d0)
      data first/ .true./
      if (first) then
         eps = one
   10    eps = eps / two
         if (eps + one .gt. one) goto 10
         eps = dsqrt(eps)
         first = .false.
      end if
      lmt = lm * t
      if (lmt .lt. eps) then
         expf1 = t * (one - lmt / two)
      else
         expf1 = (one - expfun(-lmt)) / lm
      end if
      return
      end
*
*-----------------------------------------------------------------------
*
      double precision function expfun (t)
*
*-----------------------------------------------------------------------
*
*     author:   k. f. eckerman
*     date:     10/04/94
*     purpose:  routine to compute exp (t).
*
      double precision t, zero, upval
      parameter(zero = 0.0d0, upval = 180.d0)
      if (t .lt. -upval) then
         expfun = zero
      else
         expfun = dexp(t)
      end if
      return
      end
```

```
*-----------------------------------------------------------------------
*
      integer function ibinry ( target )
*
*-----------------------------------------------------------------------
*     function: ibinry
*     author:   k.f. eckerman
*     date:     06/20/88
*     purpose:  locate record sort by target key
*
      integer nstart, nlast, left, right, try
      character*7 target, a1
      logical first
      save first, nstart, nlast
      include 'iolist.cmn'
*
*     initialization.
*
      data first /.true./
      if (first) then
         read (idex, '(2i4)', rec = 1) nstart, nlast
         first = .false.
      end if
      left = nstart
      right = nlast
*
*     begin attempts to find target.
*
   10 try = int((left + right) / 2)
      read (idex, '(a7)', rec = try) a1
      if (a1 .lt. target) then
         left = try + 1
      elseif (a1 .gt. target) then
         right = try - 1
      else
         ibinry = try
         return
      end if
*
*
*     continue search unless left > right then set ibinry to zero and
*     let calling deal with the unidentified target.
*
      if (left .lt. right + 1) then
         goto 10
      else
         ibinry = 0
         return
      end if
      end
*-----------------------------------------------------------------------*
*
      integer function icutoff(eat, ebt, egt, igamma, nspec)
*
*-----------------------------------------------------------------------*
      dimension eat(*), ebt(*), egt(*)
      ea = eat(nspec)
```

```
      eb = ebt(nspec)
      eg = egt(nspec)
      if (ea .gt. 0. .and. igamma .eq. 0) then
         do 10 i = nspec-1, 1, -1
            if (eat(i) .lt. 0.99 * ea) then
               ia = i + 1
               goto 15
            end if
   10    continue
         ia = 1
      else
         ia = 0
      end if
   15 if (eb .gt. 0.) then
         do 20 i = nspec-1, 1, -1
            if (ebt(i) .lt. 0.99 * eb) then
               ib = i + 1
               goto 25
            end if
   20    continue
          ib = 1
      else
         ib = 0
      end if
   25 if (eg .gt. 0.) then
         do 30 i = nspec-1, 1, -1
            if (egt(i) .lt. 0.99 * eg) then
               ig = i + 1
               goto 35
            end if
   30    continue
         ig = 1
      else
         ig = 0
      end if
   35 icutoff = max0(ig, max0(ia, ib))
      return
      end
*-----------------------------------------------------------------------
*
      character*(*) function lcase (a)
*
*-----------------------------------------------------------------------
*
*     author:   k. f. eckerman
*     date:     10/04/94
*     purpose:  convert character variable a to lower case.
*
      character*(*) a
      lcase = a
      do 10 i = 1, len_trim(lcase)
         ix = ichar(lcase(i:i))
         if (ix .gt. 64 .and. ix .lt. 91) then
            lcase(i:i) = char(ix + 32)
         end if
   10 continue
      return
      end
*-----------------------------------------------------------------------
*
      character*(*) function ltrim(a)
*
*-----------------------------------------------------------------------
*     function: ltrim
*     author:  K.F. Eckerman
*     date:    10/23/93
*     purpose: trim leading blanks from string a
*
      character*(*) a
      logical ok
      ok = .false.
   10 if (a(1:1) .ne. ' ') ok = .true.
      if (.not. ok) then
         n = len_trim(a) - 1
         do 20 i = 1, n
            a(i:i) = a(i+1:i+1)
   20    continue
         a(n+1:n+1) = ' '
         goto 10
      end if
      ltrim = a
      return
      end
*-----------------------------------------------------------------------
*
      double precision function timest (t, ix)
*
*-----------------------------------------------------------------------
*
*     author:   k. f. eckerman
*     date:     10/04/94
*     purpose:  function returns time in days given time string t and its
*               units ix.
*     function arguments.
      character*2 ix
      character*8 t
      double precision tp
*
      read(t,'(E8.0)') tp
      if (ix .eq. 'us') then
         tp = tp / 8.64d+10
      elseif (ix .eq. 'ms') then
         tp = tp / 8.64d+07
      elseif (ix .eq. 's ') then
         tp = tp / 8.64d+04
      elseif (ix .eq. 'm ') then
         tp = tp / 1.44d+03
      elseif (ix .eq. 'h ') then
         tp = tp / 24.d0
      elseif (ix .eq. 'y ') then
         tp = tp * 365.25d0
      endif
*
*
```

```
      timest = tp
      return
      end
*-----------------------------------------------------------------------
*
      character*(*) function ucase (a)
*
*-----------------------------------------------------------------------
*
*     author:   k. f. eckerman
*     date:     10/04/94
*     purpose:  convert character variable a to upper case.
*
      character*(*) a
      ucase = a
      do 10 i = 1, len_trim(ucase)
         ix = ichar(ucase(i:i))
         if (ix .gt. 96 .and. ix .lt. 123) then
            ucase(i:i) = char(ix - 32)
         end if
   10 continue
      return
      end
c-----------------------------------------------------------------------
c
c      integer function len_trim(a)
c
c-----------------------------------------------------------------------
c     author:   k.f. eckerman
c     date:     06/25/96
c     purpose:  determine the trim length of a character variable a
c               in the manner of Microsoft len_trim function. if the
c               len_trim function is not supported by the FORTRAN
c               compiler then active this routine by removal of the
c               'c's on the statements below and in the above name.
c      character*(*) a
c      n = len(a)
c      do 10 i = n, 1, -1
c         if (a(i:i) .ne. ' ' .and. ichar(a(i:i)) .ne. 0) then
c            len_trim = i
c            return
c         end if
c   10 continue
c      len_trim = 0
c      end
```

A3. ILLUSTRATIVE PROGRAM READEM.FOR

```
      program readem
*
* READEM illustrates the manner the electronic files of dose coefficients
* for Sandia National Laboratory can be accessed in FORTRAN.  The procedure
* uses the index file DFEXTINT.NDX to coordinate the reading of the
* coefficients from their respective files.
* The coefficient files are:
*   ICRPINGS.DAT - FGR 11/ICRP-30 ingestion intakes.
*   ICRPINHS.DAT - FGR 11/ICRP-30 inhalation intakes.
*   FGR12F31.DAT - FGR 12 Table III.1 for air submersion.
*   FGR12F32.DAT - FGR 12 Table III.2 for water immersion.
*   FGR12F33.DAT - FGR 12 Table III.3 for ground surface contamination.
*   FGR12F34.DAT - FGR 12 Table III.4 contaminated soil slab 1 cm thick.
*   FGR12F35.DAT - FGR 12 Table III.5 contaminated soil slab 5 cm thick.
*   FGR12F36.DAT - FGR 12 Table III.6 contaminated soil slab 15 cm thick.
*   FGR12F37.DAT - FGR 12 Table III.7 contaminated soil infinite thickness.
*
* All files are formatted direct access files. The RECL specification in
* the FORTRAN open statement is read from the file DCFPAK.INI. See that file
* for further discussion regarding specifications for different compliers.
*
* The file FGR1112.IDX contains the following information:
* The first record gives the record numbers of the first and last data
* record,format (2i4) which are 2 and 839.  The format of records 2 through
* 839 is (a7,a8,a2,a6,3i5,3(i4,e11.0),3f7.0,e11.0,1x,a9)
*
* Variable        Description             Format
* nuke            Name of nuclide           a7
* t12             Halflife                  a8
* ix              Halflife units            a2
* mode            Decay Modes               a6
* inh             Record # for inhalation   i5
* ing             Record # for ingestion    i5
* iex             Record # for external     i5
* id(1)           Record # of daughter      i4
* f(1)            Branching fraction        e11
* id(2)           Record # of daughter      i4
* f(2)            Branching fraction        e11
* id(3)           Record # of daughter      i4
* f(3)            Branching fraction        e11
* Ealpha          Emitted alpha energy      f7.0
* Eelectron       Emitted electron energy   f7.0
* Ephoton         Emitted photon energy     f7.0
* AMU             Atomic mass               e11
* ENDSF           Date of ENDSF             a9
*
*     author:  k.f. eckerman
*     date:    06/04/96 : 06/27/96
*
      include 'pakparm.for'
      include 'dcfpak.cmn'
      include 'batch.cmn'
      character*7 nuke
      character*5 amadx
```

```
      character*1 class, ucase
      logical ipath(9), ok
      include 'iolist.cmn'
      dbatch = .false.
      numarg = nargs()
      if (numarg .gt. 1) dbatch = .true.
*
*     this loop sets all pathway flags to true; i.e., we want dose
*     cofficients for all pathways.
*
      do 1 i = 1, 9
    1 ipath(i) = .true.
*
      call cls
      write(*,'(1x,'' View dose coefficients'')')
      write(*,'(3x,'' K.F. Eckerman and R.W. Leggett'')')
      write(*,'(3x,'' Oak Ridge National Laboratory'')')
      write(*,'(3x,'' Oak Ridge, TN 37831-6383'',/)')
*
*     open direct access files and write text to screen
*
      call openem
*
*     label 10 is the subject of a backward pointing goto
*
   10 write(*,'(a\)')' Input nuclide (e.g.; Ba-137m or <Enter> to quit)-
     .> '
      read(*,'(bn, a7)') nuke
      if (len_trim(nuke) .ne. 0) then
*
*        call nukeok to determine if nuke in data bases
*
         call nukeok (nuke, ok)

         if (.not. ok) then
            write(*,'(1x, (a),'' is not in index file!'')')
     :                 nuke(:len_trim(nuke))
            write(*,*)
            write(*,*)
            call pauseit
         else
*
*           if no inhalation class is indicated then the class with the
*           higest effective dose will be assumed.
*
            class = ' '
            write(*,'(a\)')' Clearance class (D, W, Y, V or " ")-> '
            read(*,'(bn, a1)') class
            class = ucase (class)
*
*           read in amad as a character and then do a internal read
*
            write(*,'(a\)')' Input AMAD (default 1 µm) -> '
            read(*, '(bn, a5)') amadx
            if (len_trim(amadx) .eq. 0) then
               amad = 1.0
            else
               read(amadx, '(f5.0)') amad
            end if
*
*           lets get some factors
*
            floral = 0.0
            flinhi = 0.0
            call dosecof(nuke, floral, class, flinhi, amad, ipath)
*
*           call tablem to print factors to screen
*
            call tablem (amad)
*
         endif
*
*        compute activity of chain members as at user specified times
*
         call cls
         write(*,*) ' Activity and Integrated Activity of Chain Members'
   20      write(*,'(a\)')
     :     ' Input time of interest (d); zero to quit --> '
           read(*,'(bn, e10.0)') t
           if (t .eq. 0.) goto 40
           write(*,'(5x,''Nuclide  T1/2        A(t)/Ao   intA/Ao(d)'')')
*
*          do over the maximum length of the retained chain members
*
           do 30 ispec = 1, max(nint, next)
              call birch(ispec, t, a, aint)
              write(*,'(5x, a7, 1x, a8, a2, 1p2e12.5)') nucnam(ispec),
     :             thalf(ispec), iu(ispec), a, aint
   30      continue
           goto 20
*
*        clear screen and goto 10 to get another nuclide.
*
   40    call cls
         go to 10
      endif
*
*     close all files
*
      call closem
      end
*-----------------------------------------------------------------------
*
      subroutine tablem (amad)
*
*-----------------------------------------------------------------------
*     routine:  tablem
*     author:   k.f. eckerman
*     data:     05/20/96
*     purpose:  write dose coefficients to screen.
*
      include 'pakparm.for'
      include 'dcfpak.cmn'
      character*60 head
```

```
      character*20 titles(mfact)
      character*18 units(mfact)
      character*8 t, ltrim, buffer
      character*5 buff5
      data titles / 'Inhalation          ', 'Ingestion          ',
     :              'Submersion          ', 'Immersion          ',
     :              'Ground Surface      ', '1 cm Soil Slab     ',
     :              '5 cm Soil Slab      ', '15 cm Soil Slab    ',
     :              'Infinite Thick Soil ' /
      data units / '(Sv/Bq)             ', '(Sv/Bq)             ',
     :             '(Sv per Bq s m^-3)', '(Sv per Bq s m^-3)',
     :             '(Sv per Bq s m^-2)', '(Sv per Bq s m^-3)',
     :             '(Sv per Bq s m^-3)', '(Sv per Bq s m^-3)',
     :             '(Sv per Bq s m^-3)' /
      do 100 ifact = 1, mfact
         if (ifact .le. 2) then
            nupper = nint
         else
            nupper = next
         end if
         do 50 ispec = 1, nupper
            if (iflag(ispec, ifact)) then
               call cls
               head = ' Dose Coefficients ' // titles(ifact)(:len_trim(
     :                 titles(ifact))) // ' ' // units(ifact)
               ip = (80-len_trim(head))/2
               call curpos(2, ip)
               write(*,*) head(:len_trim(head))
               write(*,*)
               t = ltrim(Thalf(ispec))
               call curright(6)
               if (ifact .eq. 1) then
                 write(buffer, '(1pe8.1)') fiinh(ispec)
                 write(buff5, '(f5.2)') amad
                 write(*,*) nucnam(ispec) //
     :                      ' T1/2 = ' // t(:len_trim(t)) // ' ' //
     :                      iu(ispec) // ' Class: ' // classo(ispec)
     :                      // ' f_1 =' // buffer // ' AMAD (µm) ='
     :                      // buff5
               elseif (ifact .eq. 2) then
                 write(buffer, '(1pe8.1)') fiorl(ispec)
                 write(*,*) nucnam(ispec) //
     :                      ' T1/2 = ' // t(:len_trim(t)) // ' ' //
     :                      iu(ispec) // ' f_1 =' // buffer
               else
                 write(*,*) nucnam(ispec) //
     :                      ' T1/2 = ' // t(:len_trim(t)) // ' ' //
     :                      iu(ispec)
               end if

               write(*,*)
               write(*,'(/,t7,''Organ'',t20,''h_T'',t32,''Organ'',t45,
     :                 ''h_T'',t57,''Organ'',t70,''h_T'',/,
     :                t7, ''----------'',t18,''--------'',
     :                t32,''----------'',t43,''--------'',
     :                t57,''----------'',t68,''--------'')')
               write(*,'(3(6x,a9,2x,1pe8.2))') (organ(j),
     :                   df(ispec, ifact, j), j = 1, 25)
               call curpos (22, 60)
               write(*,'(''Member: '',i2,'' of '',i2)') ispec, nupper
               call pauseit
            end if
   50    continue
  100 continue
      return
      end
```

INTERNAL DISTRIBUTION

1. B. A. Berven
2. J. S. Bogard
3-7. K. F. Eckerman
8. C. E. Easterly
9. R. N. Hamm
10. G. D. Kerr
11. D. C. Kocher
12. R. W. Leggett
13. P. Y. Lu
14. C. A. Little
15. D. A. McLaughlin
16. J. C. Miller
17. S. T. Purucker
18. D. E. Reichle
19. P. S. Rohwer
20. J. C. Ryman
21. A. L. Sjoreen
22. R. C. Ward
23. Central Research Library
24. ORNL Y-12 Research Library
25. Laboratory Records Department
26. Laboratory Records Department (RC)
27. ORNL Patent Office

EXTERNAL DISTRIBUTION

28. Assistant Manager of Energy Research and Development, U.S. Department of Energy, Oak Ridge Operations Office, P.O. Box 2001, Oak Ridge, TN 37831-8600

29. B. B. Boecker, Inhalation Toxicology Research Institute, Lovelace Biomedical and Environmental Research Institute, P.O. Box 5890, Albuquerque, NM 87185

30. W. E. Bolch, 205 Nuclear Science Center, University of Florida, Gainesville, FL 32611

31. A. Bouville, National Cancer Institute, Radiation Effects Branch, Executive Plaza North, Suite 530, Bethesda, MD 20892

32-42. D. Clauss , Sandia National Laboratory, P.O. Box 5800, MS 0748, Albuquerque, NM 87185-0748

43. F. J. Congel, Incident Response Division, Office for Analysis and Evaluation of Operational Data, U.S. Nuclear Regulatory Commission, Washington, DC 20555

44. F. Harper, Sandia National Laboratory, P.O. Box 5800, MS-0748, Albuquerque, NM 87185-0748

45. C. B. Nelson, Office of Radiation and Indoor Air, U. S. Environmental Protection Agency, 401 M. Street, S. W., Mail Code 6602J, Washington, DC 20555

46. H. T. Peterson, Jr., Air, Water and Radiation Division, Office of Environmental Guidance, U. S. Department of Energy, 1000 Independence Ave., SW, Washington, DC 20585

47. A. C. B. Richardson, Deputy Director for Federal Guidance Criteria and Standards Division, United States Environmental Protection Agency, Washington, DC 20460

48. G. S. Roessler, Route 1, Box 139H, Elysian, MN 56028

49. G. Runkle, U. S. Department of Energy, Albuquerque Operations, Environmental Safety and Health Division, Albuquerque, NM 87115

50. M. Smith, U.S. EPA/NAREL, 540 S. Morris Ave., Montgomery, AL 36115

51. A. Wallo, Air, Water and Radiation Division (EH-232), Office of Environmental Guidance, U. S. Department of Energy, 1000 Independence Ave., SW, Washington, DC 20585

52. W. Whicker, Professor, Department of Radiology and Health Sciences, Colorado State University, Fort Collins, CO 80523

53. S. S. Yaniv, Office of Nuclear Regulatory Research, U. S. Nuclear Regulatory Commission, Washington, DC 20555

54. M. Young, Sandia National Laboratory, P.O. Box 5800, MS-0748, Albuquerque, NM 87185-0748

55-56. Office of Scientific and Technical Information, P.O. Box 62, Oak Ridge, TN 37831