NUREG/CR-5512, Vol. 2

Residual Radioactive Contamination From Decommissioning

User's Manual

Draft Report

Sandia National Laboratories

U.S. Nuclear Regulatory Commission
Office of Nuclear Regulatory Research
Washington, DC 20555-0001

AVAILABILITY NOTICE

Availability of Reference Materials Cited in NRC Publications

NRC publications in the NUREG series, NRC regulations, and *Title 10, Energy,* of the *Code of Federal Regulations,* may be purchased from one of the following sources:

1. The Superintendent of Documents
   U.S. Government Printing Office
   P.O. Box 37082
   Washington, DC 20402–9328
   <http://www.access.gpo.gov/su_docs>
   202–512–1800

2. The National Technical Information Service
   Springfield, VA 22161–0002
   <http://www.ntis.gov/ordernow>
   703–487–4650

The NUREG series comprises (1) brochures (NUREG/BR-XXXX), (2) proceedings of conferences (NUREG/CP-XXXX), (3) reports resulting from international agreements (NUREG/IA-XXXX), (4) technical and administrative reports and books [(NUREG-XXXX) or (NUREG/CR-XXXX)], and (5) compilations of legal decisions and orders of the Commission and Atomic and Safety Licensing Boards and of Office Directors' decisions under Section 2.206 of NRC's regulations (NUREG-XXXX).

A single copy of each NRC draft report is available free, to the extent of supply, upon written request as follows:

Address: Office of the Chief Information Officer
Reproduction and Distribution
Services Section
U.S. Nuclear Regulatory Commission
Washington, DC 20555–0001
E–mail: <DISTRIBUTION@nrc.gov>
Facsimile: 301–415–2289

A portion of NRC regulatory and technical information is available at NRC's World Wide Web site:

<http://www.nrc.gov>

All NRC documents released to the public are available for inspection or copying for a fee, in paper, microfiche, or, in some cases, diskette, from the Public Document Room (PDR):

NRC Public Document Room
2120 L Street, N.W., Lower Level
Washington, DC 20555–0001
<http://www.nrc.gov/NRC/PDR/pdr1.htm>
1–800–397–4209 or locally 202–634–3273

Microfiche of most NRC documents made publicly available since January 1981 may be found in the Local Public Document Rooms (LPDRs) located in the vicinity of nuclear power plants. The locations of the LPDRs may be obtained from the PDR (see previous paragraph) or through:

<http://www.nrc.gov/NRC/NUREGS/SR1350/V9/lpdr/html>

Publicly released documents include, to name a few, NUREG-series reports; *Federal Register* notices; applicant, licensee, and vendor documents and correspondence; NRC correspondence and internal memoranda; bulletins and information notices; inspection and investigation reports; licensee event reports; and Commission papers and their attachments.

Documents available from public and special technical libraries include all open literature items, such as books, journal articles, and transactions, *Federal Register* notices, Federal and State legislation, and congressional reports. Such documents as theses, dissertations, foreign reports and translations, and non-NRC conference proceedings may be purchased from their sponsoring organization.

Copies of industry codes and standards used in a substantive manner in the NRC regulatory process are maintained at the NRC Library, Two White Flint North, 11545 Rockville Pike, Rockville, MD 20852–2738. These standards are available in the library for reference use by the public. Codes and standards are usually copyrighted and may be purchased from the originating organization or, if they are American National Standards, from—

American National Standards Institute
11 West 42nd Street
New York, NY 10036–8002
<http://www.ansi.org>
212–642–4900

DISCLAIMER

This report was prepared as 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, expressed or implied, or assumes any legal liability or responsibility for any third party's use, or the results of such use, of any information, apparatus, product, or process disclosed in this report, or represents that its use by such third party would not infringe privately owned rights.

NUREG/CR-5512, Vol. 2

Residual Radioactive Contamination From Decommissioning

User's Manual

Draft Report

Manuscript Completed:April 1999
Date Published: May 1999

Prepared by
M.A. Wernig, Interna Environmental Services, Inc.
A.M. Tomasi, F.A. Duran, Sandia National Laboratories
C.D. Updegraff, GRAM, Inc.

Interna Environmental Services, Inc.
1650 University Blvd. NE Ste 300
Albuquerque, NM 87112

GRAM, Inc.
8500 Menual Blvd. NE Ste B-335
Albuquerque, NM 87112

Under contract to:
Sandia National Laboratories
Albuquerque, NM 87185

Prepared for
Division of Risk Analysis and Application
Office of Nuclear Reactor Research
U.S. Nuclear Regulatory Commission
Washington, DC 20555-0001
NRC Job Code W6227

Abstract

The Decontamination and Decommissioning (DandD) software package developed by Sandia National Laboratories for the Nuclear Regulatory Commission (NRC) provides a user-friendly analytical tool to address the technical dose criteria contained in NRC's Radiological Criteria for License Termination rule (10 CFR Part 20 Subpart E; NRC, 1997). Specifically, DandD embodies the NRC's screening methodology to allow licensees to convert residual radioactivity contamination levels at their site to annual dose, in a way consistent with both 10 CFR Part 20 and the corresponding implementation guidance currently under development by NRC. The screening methodology employs reasonably conservative scenarios, fate and transport models, and default parameter values that have been developed to allow the NRC to qualitatively estimate the risk of terminating a license given only information about the level of contamination. A licensee has the option of specifying only the level of contamination and running the code with the default parameter values or, in the case where site specific information is available, applying site-specific parameter values and calculating dose. DandD can evaluate dose for four different scenarios: residential, building occupancy, building renovation, or drinking water. Note that only two scenarios, residential and building occupancy, can currently be used for screening using the default parameter values in the software. The screening methodology and DandD are part of a larger decision framework that allows and encourages licensees to optimize decisions regarding alternative actions at their site, including collection of additional data and information [NRC, 1998]. This document provides instructions for the application and use of the DandD software.

Table of Contents

Abstract iii

Foreword vii

1. Introduction 1-1
   - 1.1 Overview 1-1
   - 1.2 System Requirements 1-2
   - 1.3 On-Line Help 1-2
   - 1.4 Reports 1-2
   - 1.5 Organization 1-3
   - 1.6 Installation Instructions 1-3
     - 1.6.1 Nomenclature 1-3
     - 1.6.2 Preparation 1-3
     - 1.6.3 Windows 3.1 Installation 1-3
     - 1.6.4 Windows 95/NT Installation 1-5

2. Basic Program Usage 2-1
   - 2.1 Session Overview 2-1
     - 2.1.1 Sessions and Session Groups 2-1
     - 2.1.2 Session Groups: Directories and Files 2-1
     - 2.1.3 Using Session Groups 2-1
   - 2.2 Running A Test Case 2-2
   - 2.3 Defining New Sessions 2-2
   - 2.4 Main Program Window 2-3
     - 2.4.1 File Dialog Box 2-3
     - 2.4.2 File Menu 2-4
     - 2.4.3 Group Menu 2-5
   - 2.5 Main Session Window 2-6
     - 2.5.1 Session Information 2-6
     - 2.5.2 Scenario Controls 2-6
   - 2.6 Radionuclide Selection 2-7
     - 2.6.1 Entering a Radionuclide Mixture 2-7
     - 2.6.2 Changing Activity Input Units 2-8
     - 2.6.3 Importing Concentrations 2-8
   - 2.7 Basic Parameter Definition 2-8
   - 2.8 Editing Tables 2-9
   - 2.9 Executing Sessions/Scenarios 2-9
     - 2.9.1 Session Execution Status 2-10
   - 2.10 Text Reports 2-10
     - 2.10.1 Selecting/Deselecting Text Reports for Automatic Generation 2-10
     - 2.10.2 Viewing Text reports 2-10
     - 2.10.3 Defining and Editing Text Reports 2-11
     - 2.10.4 Text Report Filenames 2-12
     - 2.10.5 Using a Text Editor or Word Processor 2-12
     - 2.10.6 Report Groups 2-12
   - 2.11 Graphics Reports 2-12
     - 2.11.1 Dose Over Time 2-12
     - 2.11.2 Dose Percentages 2-13
     - 2.11.3 Controlling Graphics with Time History Parameters 2-13
     - 2.11.4 Graph Window 2-13

3. Advanced Topics 3-1
3.1 Manipulating Graphics 3-1
3.1.1 Introduction 3-1
3.1.2 The Main Graphics Window 3-1
3.1.3 Types of Charts 3-2
3.1.4 Tabbed Dialog 3-4
3.1.5 Rotate Dialog 3-6
3.1.6 Working with the Toolbar 3-6
3.1.7 The Palette Bar 3-7
3.1.8 The Pattern Bar 3-7
3.1.9 Zoom Capabilities 3-8
3.1.10 Legends 3-8
3.1.11 Working with the Data Editor 3-8

4.0 REFERENCES 4-1

Appendix A Differences Between Volume 1 and Release 1.0 of DandD A-1

Appendix B Sample Problems B-1

Appendix C Data Tables C-1

Appendix D Input Parameter Defaults D-1

Appendix E Text File Formats E-1

Appendix F FORTRAN Source Code F-1

Foreword

This technical contractor report, NUREG/CR-5512, Volume 2, was prepared by Sandia National Laboratory under their DOE Interagency Work Order (JCN W6227) with the Radiation Protection and Health Effects Branch, Office of Nuclear Regulatory Research, U.S. Nuclear Regulatory Commission. The report is a user's manual for the DandD software, which implements the modeling methodology described in Volume 1 of NUREG/CR-5512 "Residual Radioactive Contamination From Decommissioning: Technical Basis for Translating Contamination Levels to Annual Total Effective Dose Equivalent" issued October 1992. Modifications and corrections that have been made to the methodology and incorporated in the final version of the software are described in Appendix A. A listing of the FORTRAN source code is provided in Appendix F.

The purpose of the DandD software is to provide a simple screening approach for demonstrating compliance with 10 CFR 20, Subpart E. The software supports both a simple screening using generic scenarios and default parameters, as well as allowing input of site-specific parameters and modification of scenarios by changing or eliminating pathways. This software may be used to demonstrate compliance with the dose criterion in 10 CFR 20, Subpart E, as described in draft Regulatory Guide DG-4006, "Demonstrating Compliance with the Radiological Criteria for License Termination," and draft NUREG-1549, "Decision Methods for Dose Assessment to Comply with Radiological Criteria for License Termination."

This NUREG/CR report is not a substitute for NRC regulations, and compliance is not required. The approaches and/or methods described in this NUREG/CR are provided for information only. Publication of this report does not necessarily constitute NRC approval or agreement with the information contained herein. Use of product or trade names is for identification purposes only and does not constitute endorsement by the NRC or Sandia National Laboratory.

Cheryl Trottier

Cheryl A. Trottier, Chief
Radiation Protection, Environmental Risk
& Waste Management Branch
Division of Risk Analysis and Application
Office of Nuclear Regulatory Research

1. Introduction

1.1 Overview

The general objective of the DandD software, is to provide a user-friendly and somewhat automated interface to the Nuclear Regulatory Commission's (NRC) dose assessment and screening methodology for site assessment against the Radiological Criteria for License Termination Rule in 10 CFR Part 20 Subpart E [NRC, 1997]. DandD assists NRC licensees who have requested termination of their license and who, in some cases, must decontaminate lands and structures as part of the decommissioning process. The software does this by providing a tool that allows licensees to translate residual radioactive contamination levels at their site into total effective dose equivalent (TEDE) by analyzing and modeling the set of NRC-prescribed scenarios of future land-use. DandD contains models of the transport and exposure pathways associated with each of the scenarios, requiring only information on source concentration from the user. Using DandD, and within the context of the decision methodology described in draft NUREG-1549, the user may supply site-specific parameter values if available and defensible, may modify or eliminate pathways, and may propose alternative critical groups and/or scenarios.

Specifically, DandD is the software implementation of NRC's screening methodology transport and exposure models for assessing human health and safety against the dose requirements set forth in 10 CFR Part 20 Subpart E [NRC, 1997]. For this discussion, screening refers to the release of a site where little or no site-specific information is known or used, other than level of contamination. To provide useful and defensible screening level calculations, the NRC has developed reasonably conservative scenarios, pathway models, and parameter values, and has implemented these in DandD. "Reasonably conservative" implies that the calculated doses are much more likely to be overestimates of the actual dose rather than accurate estimates or underestimates, but at the same time are not necessarily worst case estimates. As a result, the scenarios and models implemented in DandD are relatively simple. To perform these screening calculations, the DandD software automates the scenarios, models, mathematical formulations, and assumptions documented in NUREG/CR-5512, Volume 1 [Kennedy and Strenge, 1992] with a few corrections and enhancements. The generic modeling approach defines radiation exposure scenarios to address residual radioactive contamination inside buildings, in soils and in ground water. For buildings, two scenarios are presented. These scenarios relate both volume and surface contamination levels to estimates of the annual TEDE received during a year of exposure with the conditions defined in the scenarios.

For the simplest level of analysis (previously referred to as Level 1 screening), the user is required to provide a minimum amount of site-specific information. In general, only information about contaminant concentration is required for this level of analysis. This level of analysis is automated in DandD, and therefore provides certain licensees a simple and cost-effective method for demonstrating compliance using a minimum amount of information. This level of analysis implements the generic scenarios and models from NUREG/CR-5512, Volume 1, and uses deterministic values for all model parameters that have also been defined to be reasonably conservative. The default parameter values in DandD Version 1.0 have been defined through a systematic process of assessing the variability of each parameter across the U.S. and then defining default values that produce generic dose estimates that are unlikely to be exceeded at any real site[1]. In summary, the default models and parameter values are intended to estimate the upper range of the dose that the average member of the critical group could receive at any site given the contaminant level at that site.

If a licensee has site-specific information for certain parameters, they may choose to replace the default parameter values with alternative values, and employ the default transport and exposure models. This level of analysis (previously referred to as Level 2 screening) is easily conducted with DandD. Licensees are not required to conduct the "Level 1" screening calculations prior to proposing changes to parameter values if they have such information to do so.

The default parameter values for the NUREG/CR-5512 modeling (which are implemented in DandD) are based on probability distributions representing the variability across the country. As a consequence, the licensee would likely need little supporting information to defend significant changes to the physical parameter values. For example, the probability distributions used in defining the default values for radionuclide sorption in soils for the NUREG/CR-5512 residential scenario models is

[1] Parameter Analysis to be published in NUREG/CR-5512 Volume 3

based on the variety of all possible soil types and geochemical conditions. In order to provide a defensible screening process where a license could be terminated based only on residual contamination data, the parameter analysis produces default values for some of the sorption coefficients that are representative of the lower measured values. Therefore, many sites would be able to defend a significantly higher sorption coefficient and input this value into DandD. This approach of moving away from the "reasonably conservative" values used in the NUREG/CR-5512 modeling could be used by all sites until the point that further reduction in simulated dose would require model changes. This would necessarily require the licensee to step away from using DandD. At that point, new model parameter values would have to be developed and defended by the licensee. Model changes should lead to less conservative models and lower doses with each iteration, because the NUREG/CR-5512 models are designed to be inherently conservative.

DandD is intended to be implemented within the structure of NRC's decommissioning decision framework documented in draft NUREG-1549 [NRC, 1998]. This process has been defined to allow licensees to define the most cost-effective decommissioning and license termination strategy by evaluating alternative actions at their site, including possible reductions in uncertainty that would reduce overall remediation costs. As such, because of nature of the DandD models and default parameter values, the NRC does not expect licensees to define concentration clean-up levels based on preliminary DandD dose calculations that fail to meet 10 CFR Part 20 dose criteria; rather, licensees are encouraged to evaluate the cost of added information and the value it adds in better defining remedial actions.

The results of the modeling in DandD is used to produce reports, in NRC-defined and accepted text and graphics formats, that will allow the NRC to efficiently assess compliance with the 10 CFR Part 20 dose criteria and to determine if more detailed modeling should be required.

1.2 System Requirements

The DandD software requires a minimum configuration of:

- an IBM compatible PC with an 80386 or higher processor (80486 or higher recommended)
- Microsoft Windows 3.1
- a VGA monitor
- a mouse or other pointing device
- 4 megabytes of Random Access Memory (RAM)
- 5MB of free disk space

1.3 On-Line Help

DandD was designed to give the user an on-line user manual that reduces the need to refer to a hard-copy manual. It provides on-line documentation and software guidance using the Windows Help system. The Windows Help facility is a hypertext system that allows the user to easily jump from one topic to another and back again. Hypertext is basically the same as regular text with one exception: hypertext contains connections within the text to other documents. These hypertext links, called hyperlinks, connect to other documents in the help system and appear as colored, underlined text. By clicking on the hyperlink text, the program will display a different help topic document. When DandD is run for the first time, it will display an introductory help topic which the user should read.

The system includes a Search command to help locate particular topics, a Bookmark command which can flag topics of interest, an Annotate feature that lets the user add their own comments to the help text, a Copy command that copies important information into a word processor, and a Print command to print help text. If the user is not familiar with the Windows Help system they should select the **Help** menu option **How To Use Help** for instructions.

The user can get program help at any time by pressing Help buttons that are displayed on most program windows, by pressing the F1 function key, or by using the Help menu. It can be difficult to view both program windows and the Help system window at the same time.

1.4 Reports

DandD produces several types of reports including the FORTRAN output file, text reports and graphics data. The FORTRAN output file contains all the input and output from the FORTRAN modeling codes that implement the methodology described in NUREG/CR-5512, Volume 1. Text reports are a user-defined combination of reformatted sections of the FORTRAN output file. The program has a limited report definition capability that is used to create different report formats for other regulatory agencies.

The graphics data is a dose history produced by the FORTRAN modeling codes. It reports doses at specific

calculation intervals or timesteps over the entire period of a scenario execution. The doses reported are total dose for each radionuclide and progeny, total dose for each radionuclide and progeny by pathway, total dose for each pathway, and the total dose at that timestep. Graphical displays are created from this dose history file. Graphics may be imported into editors that manipulate Windows bitmaps.

Once a scenario has been executed text reports may be generated and viewed. When the user defines a new session a set of text reports are defined for each scenario and automatically generated each time that scenario is executed. The user may choose which set of reports are used for a particular group of sessions and which reports are automatically generated for each scenario.

1.5 Organization

This report consists of four sections. They provide instructions for using the code and all of its features. Section 2 presents basic program usage commands and Section 3 provides guidance on using the advanced features of the code. Appendix A describes the differences between NUREG/CR-5512, Vol. 1, [Kennedy and Strenge, 1992] and this Volume as they were implemented in the code. Appendix B provides some sample problems, Appendix C contains the data tables that are used as input to DandD, and Appendix D shows the default parameter values for each scenario. Appendix E describes the text file formats used by the code. Appendix F contains the FORTRAN source code for the DandD calculations.

1.6 Installation Instructions

NOTE: The following instructions apply to the installation of DandD from floppy disks. The DandD setup program is also available for downloading from the Internet. Installation instructions for the downloaded version are in a file called README.TXT in the same folder as the setup program.

Four installation disks are distributed with the DandD software. The first two are labeled *Microsoft Win32S Setup, Disk 1 of 2* and *Disk 2 of 2,* and are used to install the Win32S memory management software, if necessary. The third and fourth diskettes, labeled *DandD Version 1.0, Disk 1 of 2* and *Disk 2 of 2*, are used to install the DandD program and the graphics software needed to run the program. Section 1.6.3 describes how to install the software under the 16-bit Windows operating system and Section 1.6.4 tells how to do it under the 32-bit Windows 95/NT operating systems.

1.6.1 Nomenclature

<Windows drive>	The drive that the Windows directory is on, usually c:.
<source floppy drive>	The 3 ½" source floppy drive, probably a: or b:.
<Win dir>	The directory where Windows is installed, usually Windows.

1.6.2 Preparation

Check to make sure that the following settings are contained in the config.sys file which should be in the root directory. If the number of buffers and files are already set to a higher number than below, do not decrease them to these values. This file contains commands that specify part of your machine's configuration.

BUFFERS=30	This command reserves memory storage areas called "buffers" which are used for transferring information to and from disks.
FILES=50	This command gives the MS-DOS operating system access to a specific number of files at one time.

If the above settings need to be changed, the config.sys file will need to be edited and the machine rebooted before installing Win32S or DandD.

Use the File Manager **Search** or Windows 95/NT Explorer **Find** function to see if the following files exist on the system and write down their directory location prior to starting the installation procedure.

win32s.ini
pkzip.exe
cfx2*.*

1.6.3 Windows 3.1 Installation

This section describes how to install the Win32S and DandD software under the Windows 3.1 and Windows for Workgroups 3.11 operating systems.

1.6.3.1 Installing the Win32S Memory Management Software

The DandD software requires a memory management

software package called Win32S to be installed on the system when running under the Windows 3.1 or 3.11 operating system. Win32S allows a 32-bit program to run in a 16-bit Windows environment. To see if it needs to be installed, search for a file called win32s.ini using the File Manager. If this file exists, Win32S might not need to be installed.

If Win32S is already installed, check to see if the version needs updating by doing the following:

1. In an MS-DOS prompt window, type in:

   cd <Windows drive>\<Win dir>\system

2. Type in:

   type win32s.ini

3. Note the version number from the line that contains Version= in the [Win32s] section.

If the version number specified here is less than 1.20.123.0, the Win32S installation provided with the DandD software should be run to install a newer version of Win32S. Before installing the new version, it is a good idea to make a backup of the current version first. Follow the instructions contained in the next section to do this, or skip to the section following it to proceed with installation of the Win32S software.

1.6.3.1.1 Backing Up An Existing Version of Win32S

To make a backup copy of the existing Win32S software perform the following steps:

1. Place the disk labeled *Microsoft Win32S Setup Disk 1 of 2* into the floppy disk drive.

2. In an MS-DOS prompt window, type in:

   **cd **
   mkdir w32tmp

Note: If the second command above fails because a directory of this name already exists, type in a different name and substitute it in the following commands.

3. Then type in:

   cd w32tmp
   copy <source floppy drive>\backup32.bat

4. If the **pkzip** program does not exist on your computer, type in:

   copy <source floppy drive>\pkzip.exe

5. Place an empty, formatted disk into the floppy disk drive.

6. Type in:

   backup32 <Windows drive> <source floppy drive>

   for example: **backup32 c: a:**

7. Then delete the backup and pkzip programs by typing:

   del backup32.bat
   del pkzip.exe

Now run the Win32S installation program as outlined in the next section.

1.6.3.1.2 Installing Win32S

1. Start Windows, if it is not already running.

2. Place the diskette labeled *Microsoft Win32S Setup Disk 1 of 2* into the floppy disk drive.

3. From the **File** menu of the *Windows Program Manager* select **Run**, type in:

   <source floppy drive>\setup

   for example: **a:\setup**

Note: An alternative method to step 3 in the above procedure is to start the Windows File Manager, double-click on the source floppy drive, and then double-click on the setup.exe file or its icon.

Follow the installation program instructions. After the Win32S installation program is done, it will restart Windows. Now the DandD installation program can be run as described in the next section.

1.6.3.2 Installing DandD and The Graphics Library

Once Win32S has been installed, the DandD program and its associated graphics library must be installed. To run the DandD installation program do the following:

1. Start Windows, if it is not already running.

2. Place the diskette labeled *DandD Version 1.0 Disk 1 of 2* into the floppy disk drive.

3. From the **File** menu of the Program Manager select **Run**, type in:

   <source floppy drive>\setup

   for example: **a:\setup**

Note: An alternative method to step 3, above, is to start the *Windows File Manager*, double-click on the source floppy drive, and then double-click on the **setup.exe** file or its icon.

4. A Welcome message will be displayed that asks if the installation should continue.

5. It asks for a destination directory. The default is c:\dandd, although a different directory name may be substituted. If the directory name already exists, it will ask for confirmation that the software should be installed in that directory.

6. It will ask if backups of existing files should be made. Definitely click on YES if DandD has been previously installed or if the system has a software package called ChartFX on it (cfx*.* files from Section 1.6.2). If the answer was yes, it will ask for a backup destination directory.

7. The DandD program files will now be installed. When the installation is complete, it will ask if the DandD program icons should be installed. Click on YES and select a program group where the icon should be placed. The default is a new program group called DandD.

1.6.4 Windows 95/NT Installation

DandD does not require the Win32S files to run under Windows 95/NT and the Win32S diskettes should <u>not</u> be installed because Windows 95/NT have their own version of these files. The current version of the installation package requires that all directory defaults be used, otherwise the DandD program will not initialize properly.

1. Place the diskette labeled *DandD Version 1.0 Disk 1 of 2* into the floppy disk drive.

2. Click on the **Start** button on the taskbar located at the bottom of the screen.

3. Click on **Run** and type in:

   <source floppy drive>\setup

   for example: **a:\setup**

Note: An alternative method to step 3, above, is to start the Windows Explorer, click on the source floppy drive, and then double-click on the **setup.exe** file or its icon. Or, the user can simply use the **Add/Remove Programs** feature of Windows 95/NT to "find" the setup program and initiate it.

4. A Welcome message will be displayed that asks if the installation should continue.

5. It asks for a destination directory. The default is c:\Program Files\Dandd and should not be changed.

6. It will ask if backups of existing files should be made. Definitely click on YES if DandD has been previously installed or if the system has a software package called ChartFX on it (cfx*.* files from Section 1.6.2). If the answer was yes, it will ask for a backup destination directory.

7. The DandD program files will now be installed. When the installation is complete, it will ask if a shortcut should be installed in the Programs menu. Click on YES.

8. Do <u>not</u> start DandD the first time from the shortcut created in step 7. Start Windows Explorer, double-click on the Program Files directory and double-click on the DandD directory. Then double-click on the NRC application icon in this directory. This should start the DandD program and display a disclaimer window.

9. To create an on-screen shortcut, close the DandD program and click-and-drag the NRC application icon to the desktop area.

2. Basic Program Usage

2.1 Session Overview

This section describes some of the basic concepts behind DandD's use of sessions and the directories and files associated with a session.

2.1.1 Sessions and Session Groups

DandD is organized in terms of sessions. A session consists of one or more selected scenarios; the associated input parameters needed to run the scenarios; and, once the scenarios have been run, the resulting output data. DandD stores information about a session in a session document.

DandD stores sessions in groups. Each session group has a file that contains the list of sessions in that group (identified with a .sgp extension) and a directory that stores sessions. The group directory is named by dropping the .sgp extension from the group filename. Each session has a subdirectory within the group directory that contains its associated files, such as text reports.

When a session is first opened, the Main Session Window is displayed. This window will be displayed during the entire run of a session; closing this window will close the session. The left hand side of the window contains global session information and the right side contains scenario specific information.

There is a default session group that is used when the program starts up for the first time; using this group for all sessions is fine. Other session groups may be created if the number of sessions gets large, or they are naturally grouped in some way; for example, multiple sessions for several different sites. Another reason to create different session groups is to spread session files over the available disk space. Groups may be stored on different network drives, hard disks, or even a floppy disk (although access times may be slowed considerably).

2.1.2 Session Groups: Directories and Files

Within the session group directory there is a subdirectory called sessions that contains all the session documents for the group. Each session has a file number assigned to it at creation. This zero-based four digit number is used to create a unique session filename and a unique subdirectory for each session within the session group directory. For example, the 5th session in a group will have a session file named

<session group directory>\sessions\sess0004.ses

and an associated directory

<session group directory>\0004.

The session file (.ses) contains the information needed to execute the scenarios.

Files produced by the execution of each scenario that are marked as being saved are stored within the session subdirectory. Each scenario has a prefix that is attached to all files produced for that scenario; occ for Building Occupancy, ren for Building Renovation, dw for Drinking Water and res for Residential. Each type of file is further identified by an extension; .out for the modeling code output file, .rep for text reports, and .his for the dose history file. Report files (see Section 2.10) also have a two digit report number attached to them; for example, occ00.rep and occ01.rep are the first and second text reports generated for the Building Occupancy scenario.

DandD uses an initialization file called nrc.ini. The state of the program is saved in the initialization file each time it exits; and each time it starts up, the state from the last execution is restored. Part of the state that is saved is the name of the current session group. DandD will attempt to open this session group each time it starts. If the session group does not exist, the default session group will be opened instead. The name of the current session group file is displayed on the right side of the status bar. The full directory path to the current session can be located using the Program Information (Section 2.4.2.9) menu option of the main application File menu.

2.1.3 Using Session Groups

Session groups can only be manipulated from the main program window, i.e. when there is no open session. The Session Group submenu of the Groups main menu contains all the commands associated with session groups. To create a new session group, select the New menu option. A Windows File Dialog will be displayed for entering a session group filename. The location of the session group can be changed by selecting different drives and/or directory paths. To open or delete an existing session group, select the Open or Delete menu options. Deleting a session group will delete not only the

group file, but the entire group directory including all the session files and session directories.

The following options on the main program window File (Section 2.4.2) menu apply to session groups:

Run Batch:
: Used to open and execute each session in the group in succession.

Write Disk:
: Produce a disk for the NRC that contains data from the currently loaded session group.

Read Disk:
: Reads a disk that was produced by the Write Disk function into the current session group.

2.2 Running A Test Case

This section describes the normal sequence of actions taken to open and run a session. Load the session group called testcase.sgp using the Groups, Session Group, Open menu option from the Main Program Window. Position the mouse pointer on Groups, click the mouse button, then position the pointer on Session Group, click the mouse button again, then position the pointer on Open, and click the mouse button.

The Open Session Group Dialog window will then appear. Double click on testcase.sgp.

To open a session, click on File on the Main Program Window and select Open. The Open Session File window will display. Select a session title and double click on its name.

The Main Session Window will be displayed. This window will remain open during the entire session and will be updated as changes in the status of the session definition occurs. Closing this window will close the current session.

Click on the Radionuclide Mixture button for the Residential scenario to view the radionuclide mixture. A window will be displayed that contains a scrollable table of all of the parent radionuclides, and a list of the radionuclides that have been selected for the test case mixture. The initial concentration for each radionuclide in the mixture is displayed to the right of the selected radionuclide. Click on Cancel when finished viewing the radionuclide mixture.

Click on the Parameters check box for the Residential scenario to view the scenario specific parameters. The Residential Parameters window will display. Scroll through the list of parameters using the arrow keys or the scroll bar. Note that scrolling through the list will cause a description of the current parameter to be shown at the bottom of the dialog box. Click on Cancel when finished.

Now click on the Execute Scenarios pushbutton on the left hand side of the Main Session Window to run all of the selected scenarios, which in this case is just the Residential scenario A new window will be displayed that allows changing the types of reports that will be created and selecting other execution options. Do not change any of the current settings and click on OK to start running the session. Another window is displayed that will report the status of the session execution. The name of the currently executing scenario and the time elapsed in the session execution are displayed at the top of the window. Messages reporting the status of the executing modeling codes are also shown. When the session is finished, a message will be displayed. Click on Yes. Review the messages in the window by scrolling through them. Click on Done when finished.

On the Main Session Window, click on Text Reports to review the results. The Text Reports Selection window will display. Select the report named NRC Report by clicking on its name in the box on the right side to highlight it, and then click on View (or just double click on the report name) to view the report. When finished, click on the File menu in the viewing window and select the Close menu option.

Click on Graphics Reports on the Main Session Window to see a graphical display of the time step data. A window will be displayed for selecting the desired number of time steps, radionuclides, and pathways. Click on OK when finished with the selection and a graph will be displayed. Click on File and Close to return to the main session window. Click on the File menu Exit option (or use the keystrokes Alt-F X) in the Main Session Window to exit the program.

2.3 Defining New Sessions

This section describes the normal sequence of actions taken to create a session. First, reload the default session group called sessions.sgp using the Groups, Session Group, Open menu options on the Main Program Window. To start a new session, click on File on the main menu and select New. This will display a window that can be used to uniquely identify the session. Enter a title and a description before continuing. When these have been entered, click on the OK button and the Main Session Window will be displayed.

Selecting a scenario will activate its associated actions and allow it to be run during a session execution. Select or deselect a scenario by clicking on the name of the scenario in the Main Session Window with the left mouse button, or by pressing the Enter key when the name is highlighted. A selected scenario has a check mark in its associated check box. A deselected scenario has no check mark and its associated actions are disabled, as indicated by the grayed text. All scenarios can be selected or deselected by clicking on the Clear/Set Selections button on the left side of the main session window.

A radionuclide mixture must be entered before a scenario may be executed. Enter a radionuclide mixture by clicking on the Radionuclide Mixture check box for a specific scenario, or by pressing the Enter key when the check box is highlighted. This will bring up a window that allows selecting the radionuclides and editing the initial activities. Although each scenario has its own radionuclide mixture, the Building Occupancy and Building Renovation scenarios share the same initial mixture and concentrations and the Drinking Water and Residential scenarios share the same initial mixture and concentrations. After the initial mixture has been defined, the mixture for each scenario must be edited separately.

Once a radionuclide mixture has been entered a scenario may be executed. Parameters for each scenario have default values that produce generic dose estimates. DandD allows variable parameters to be edited to replace default parameter values with site-specific values. Edit variable parameters by clicking on the Parameters check box for a specific scenario, or by pressing the Enter key when the check box is highlighted. This will bring up a window that allows editing the parameters.

The Drinking Water and Residential scenarios have other tables that may be edited beyond these basic parameters by clicking on the Tables checkbox.

Execute a specific scenario by clicking on the Execute check box for that scenario on the Main Session Window, or by pressing the Enter key when the check box is highlighted. Execute a session (all selected scenarios) by clicking on the Execute Scenarios button on the left side of the main session window.

Once a scenario has been executed, text reports may be generated and viewed. When defining a new session, a set of text reports are defined for each scenario and automatically generated each time that scenario is run. The set of reports used for a particular group of sessions and the reports automatically generated for each scenario are selectable by the user. View text reports by clicking on the Text Reports check box for a specific scenario, or by pressing the Enter key when the check box is highlighted.

Continue defining the session using the Main Session Window and other subsidiary windows Help is available by clicking on the help button contained in the window or using the Help system topics to get program help. When finished defining the session click on File and Save to save the session. Exit the program by clicking on the File menu Exit option (or use the keystrokes Alt-F, X).

2.4 Main Program Window

The main program window for DandD will be displayed each time DandD is started. From this window an existing session may be opened or a new session started.

The title bar is located along the top of a window which contains the name of the application. The title bar may be used to move and resize the main program window. A menu bar is located below the title bar of the main application window. This menu will change depending on which subordinate program window is active. A toolbar is located below the menu bar, which consists of a series of buttons that represent common actions that are executed during the course of program execution. A status bar is located at the bottom of the main application window. This area is used to show one-line descriptions of the function of menu options and screen controls, and messages about actions currently being executed. Each of these and the File, Group and Help menus are discussed in this section. Window attributes, such as font and color, that are modifiable using the standard Windows programs such as Control Panel or Desktop are reflected in all DandD program windows.

2.4.1 File Dialog Box

Any activities that require file manipulation will open a File Dialog box window. The File Dialog box allows selecting an existing file, specifying a new filename or selecting an existing directory. This window is visible when selecting a file related activity such as the New, Open, and Delete commands.

File Name

Type or select a filename. This box lists files with the extension selected in the List Files of Type box. A filename can contain up to eight

characters and an extension of up to three characters.

List Files of Type

Select a file type by filename extension.

Drives
Select a drive.

Directories
Select a directory.

Network...
Choose this button to connect to a network drive, assigning it a new drive letter.

2.4.2 File Menu

The main program window File menu is used to manipulate the session files and global options. It offers the following commands which are explained further in this section.

2.4.2.1 New Command

Use this command to create a new session document in DandD.

2.4.2.2 Open Command

Use this command to open an existing session document. Only one session document can be open at a time.

2.4.2.3 Delete Command

Use this command to delete an existing session document and all associated session files, including modeling code output, history files, text reports and any intermediate program-generated files.

2.4.2.4 Run Batch

Use this command to execute all sessions in the current session group. The status window for executing a single session will be displayed throughout the batch run displaying information about the execution of each individual session.

2.4.2.5 Write Disk

Use this command to write the current session and report group to a floppy disk. All session documents and associated session files will be written to the disk. This command is used to submit data to the NRC in the proper format. It asks which drive the files should be copied to and then starts copying the files.

2.4.2.6 Read Disk

Use this command to read the session and report groups from a floppy disk that was created using the Write Disk command (see previous section). All session documents and associated session files will be read from the disk and placed onto the hard disk. This command is used to import session groups from other users.

NOTE: This command will overlay any files of the same name that already exist on the system. The recommended procedure is to use the Change Work Directory command (see next section) prior to running this command.

NOTE: This command may only be used to recover files created with the current version of DandD.

2.4.2.7 Change Work Directory

Use this command to change the program working directory. The default working directory is either the working directory at the end of the last execution of the program or the directory from which the program is currently being executed. The working directory is where the tables of data used by the modeling codes must be located. If these tables are not in the working directory, they should be copied into it from the default directory's tables sub-directory.

2.4.2.8 Turn On/Off Logging

Use this command to turn the program logging flag on or off. When logging is turned on, DandD will record program execution and error messages in a log file in the current working directory called DandD.log. A check mark displayed to the left of the menu option text indicates that logging is turned on. Information from this log file can be used by the system support personnel to locate errors.

2.4.2.9 Program Information

Use this command to show current program information. This information includes program version, paths to the current working directory, session and report groups files, and registered user information.

2.4.2.10 Print Setup Command

Use this command to select a printer and a printer connection. This command opens a Print Setup dialog box for specifying the printer and its connection.

2.4.2.11 Exit Command

Use this command (or the Close command on the Application Control menu) to exit the DandD program. DandD prompts to save open documents with unsaved changes.

2.4.3 Group Menu

The main program window Group menu contains two submenus which allow the user to manipulate session and report groups. It offers the following commands which are explained further in this section.

2.4.3.1 Session Group Menu

2.4.3.1.1 New Command

This command creates a new session group. Use the File Dialog Box to select a location and name for the session group file. Session group files end with a .sgp file extension. When a new session group is created, a directory is created in which session documents and other session files are stored for the group.

2.4.3.1.2 Open Command

This command opens an existing session group. Use the File Dialog Box to select the name of an existing session group file. Session group files end with a .sgp file extension. The name of the currently loaded session group appears in the status line message bar.

2.4.3.1.3 Delete Command

This command allows deleting an existing session group. Use the File Dialog Box to select the session group filename. Session group files end with a .sgp file extension. When a session group is deleted, the entire group directory is deleted and all the session documents are deleted.

2.4.3.1.4 Write as Text Command

This command allows saving all session files from the current group as text files (see Appendix E for a description of the file format). It is normally used in conjunction with the Read from text command (see next section) to transfer an entire session group to another directory or machine. It can also be used create a new set of sessions with different parameters by using a text editor. When it is executed, it will create the text files in a subdirectory called ascii under the current session subdirectory with a file extension of .asc. The following procedure should be used when moving files:

1. Open the session group to be moved.
2. Run the Write as text command.
3. Click on yes to verify overwriting of existing text files in this subdirectory.
4. Create a new session group using the Session, Group, New command (see Section 2.4.3.1.1).
5. Copy the files from the first session group's ascii subdirectory to the new session group's ascii subdirectory.
6. Run the Read from text command.

2.4.3.1.5 Read from Text Command

This command allows retrieving session files that were saved as text files using the Write as text command (see previous section). Caution should be used when running this command because it overlays all session files in the current session group. Use the following procedure to import the text files:

1. Create a new session group using the Session, Group, New command (see Section 2.4.3.1.1).
2. Copy the files from their source to the new session group's ascii subdirectory.
3. Run the Read from text command.
4. Select the session group name that was created in step 1, above, from the File Dialog box.
5. Click on yes to confirm creating the session group from the text files.

2.4.3.2 Report Group Menu

2.4.3.2.1 New Command

This command creates a new report group. Use the File Dialog Box to select a location and name for the report group file. Report group files end with a .rgp file extension.

2.4.3.2.2 Open Command

This command opens an existing report group. Use the File Dialog Box to select the name of an existing report group file. Report group files end with a .rgp file extension. The name of the currently loaded report group appears in the status line message bar.

2.4.3.2.3 Delete Command

This command allows deleting an existing report group. Use the File Dialog Box to select the report group filename. Report group files end with a .rgp file extension.

2.4.3.2.4 Define Text Reports

See Section 2.10.3 for information about this command.

2.5 Main Session Window

The Main Session Window will be displayed after starting a new session or opening an existing session. This window will remain open during the entire session and will be updated as changes in the status of the session definition occurs. Closing this window will close the current session.

The left side of the window shows information about the session (Section 2.5.1) and the right side shows information about each scenario (Section 2.5.2). All of the basic actions performed during the definition of a session are controlled from this window.

Activate either side of the window by clicking the left mouse button on a blank portion of that side of the window or toggle back and forth between sides using the F6 function key. Once activated use the tab and/or arrow keys to move to entry or check boxes. To perform the actions associated with a check box press Enter when the checkbox is highlighted or click on it with the left mouse button. On the scenario specific side of the window the Tab key will move from scenario to scenario, the arrow keys move through the actions associated with that scenario.

2.5.1 Session Information

The left side of the main session window contains information about the session, along with the windows controls that perform actions that affect all scenarios.

2.5.1.1 Basic Information

Title:
: Used to identify the session

*Description:
: Descriptive text for session, for example site information

Directory:
: Identifies the directory where saved files associated with the session are stored

Created:
: Date and time of creation of this session

*By:
: Person who created the session, initialized to registered user information

*From:
: Organization of person who created the session, initialized to registered user

Last Modified:
: Date and time session was last modified

Last Executed:
: Date and time session was last executed

* - These fields can be edited

2.5.1.2 Session Actions

Save Reports
: If this checkbox is marked, the modeling code output file and text reports will be saved in the session directory.

Save History
: If this checkbox is marked, the modeling code dose history file will be saved in the session directory.

Clear (Set) Selections
: Toggles on and off the selection status of all the scenarios.

Execute Scenarios
: Executes all selected scenarios that have radionuclide mixtures specified for them.

Copy Text Reports
: Copies all saved text reports to another directory or disk.

Generate Reports
: Generates all selected text reports from the output of the last execution.

2.5.2 Scenario Controls

The right side of the main session window contains

windows controls that perform actions that affect each individual scenario. These are described below.

2.5.2.1 Selecting Scenarios

Selecting a scenario will activate its associated actions and allow it to be run during a session execution. Select or deselect a scenario by clicking on the name of the scenario with the left mouse button, or by pressing the Enter key when the name is highlighted. A selected scenario has an "X" in its associated check box. A deselected scenario has no "X" and its associated actions are disabled, as indicated by the grayed text. Select or deselect all scenarios by clicking on the Clear/Set Selections button on the left side of the main session window.

2.5.2.2 Entering Radionuclide Mixtures

A radionuclide mixture must be entered before a scenario may be executed (see Section 2.6). Enter a radionuclide mixture by clicking on the Radionuclide Mixture check box for a specific scenario, or by pressing the Enter key when the check box is highlighted. Although each scenario has its own radionuclide mixture, the Building Occupancy and Building Renovation scenarios share the same initial mixture and concentrations and the Drinking Water and Residential scenarios share the same initial mixture and concentrations. After the initial mixture has been defined, the mixture for each scenario must be edited separately.

2.5.2.3 Editing Variable Parameters

Once a radionuclide mixture has been entered, a scenario may be executed. Parameters for each scenario have default values that produce generic dose estimates. DandD allows variable parameters to be edited to replace default parameter values with site-specific values. Edit variable parameters by clicking on the Parameters check box for a specific scenario or by pressing the Enter key when the check box is highlighted (see Section 2.7).

2.5.2.4 Editing Tables

The Drinking Water and Residential scenarios have other parameters that may be edited beyond the basic ones and they are contained in tables. Edit the tables by clicking on the Tables check box for a specific scenario or by pressing the Enter key when the check box is highlighted (see Section 2.8).

2.5.2.5 Executing a Scenario or Session

Execute a specific scenario by clicking on the Execute check box for that scenario or by pressing the Enter key when the check box is highlighted. Execute a session (all selected scenarios) by clicking on the Execute Scenarios button on the left side of the main session window (see Section 2.9).

2.5.2.6 Viewing Text Reports

Once a scenario has been executed, text reports may be generated and viewed. When defining a new session a set of text reports are defined for each scenario and automatically generated each time that scenario is run. Which set of reports are used for a particular group of sessions and which reports are automatically generated for each scenario may be chosen by the user. View text reports by clicking on the Text Reports check box for a specific scenario or by pressing the Enter key when the check box is highlighted (see Section 2.10).

2.5.2.7 Defining and Viewing Graphics

Graphical reports are also produced and displayed by DandD. Define and view graphics reports by clicking on the Graphics Reports check box for a specific scenario or by pressing the Enter key when the check box is highlighted (see Section 2.11). Graphics may be imported into other applications that accept Windows bitmaps

2.6 Radionuclide Selection

2.6.1 Entering a Radionuclide Mixture

The Radionuclide Mixture dialog box allows entering the parent radionuclide mixture for a selected scenario. Up to 20 parent radionuclides may be selected. The parent radionuclides listed are from NUREG/CR-5512, Volume 1 [Kennedy and Strenge, 1992], as amended, and are listed in Appendix C, Table C.1.

Select or de-select radionuclides from the list of all radionuclides by clicking the left mouse button on an entry in the list. Use the scroll bar on the right side of the list to position the cursor on the desired radionuclide or use the arrow keys. When the left mouse button is clicked once while positioned on the radionuclide then that entry is selected, highlighted, added to the Selected Radionuclides list on the left and to the scroll list on the right side of the All Radionuclides box. If it is clicked again then that entry will be deselected and removed from both lists.

A list box search is also provided to go directly to radionuclides beginning with a particular number or letter. To use the search facility, press the tab key until the list of radionuclides is highlighted. Then press the first number or letter of the radionuclide repeatedly until the particular radionuclide entry is under the cursor, then press the Insert key to select or deselect it.

As radionuclides are selected for a mixture, entries will be added to the Selected Radionuclides scroll box. The values in the entry boxes in this list are the initial activities used for parent radionuclides by the modeling codes. These are initially set to a default of one unit. When selecting a radionuclide, the entry box for its initial activity is automatically highlighted. Edit this entry to change the initial activity value and then press Tab to move back to the list of all radionuclides to select the next radionuclide. The arrow keys can be used to move through the entry box list to highlight and replace activity values or use the scrollbar and mouse. Before entering the initial activity values the input units may be changed (see Section 2.6.2) for activities by pressing the button labeled Change Units.

Parent radionuclides selected for a mixture are shown in the scroll list on the right side of the All Radionuclides box and the number of radionuclides in the mixture is displayed above that. Highlighting an entry on this list will position the list of all radionuclides at that entry and select the corresponding entry in the initial activity entry list. This is useful for editing an existing radionuclide mixture.

Although each scenario has its own radionuclide mixture, the Building Occupancy and Building Renovation scenarios share the same initial mixture and concentrations, and the Drinking Water and Residential scenarios share the same initial mixture and concentrations. After the initial mixture has been defined, each scenario mixture must be edited separately.

When finished entering the radionuclide mixture click on the OK button or press the Enter key to keep the changes. Click on the Cancel button or press the Escape key to discard the changes.

Notice that the radionuclide lists contain U_Nat for natural Uranium and Th_Nat for natural Thorium. Natural Uranium consists of three radionuclide chains - ^{234}U, ^{235}U, and ^{238}U. The initial activity entered for natural Uranium is distributed among the parent radionuclides as follows: 48.9% ^{235}U, 2.25% ^{234}U, and 48.9% ^{238}U (see footnote f in Table E.6 of NUREG/CR-5512, Volume 1 [Kennedy and Strenge, 1992]). Natural Thorium is the same as ^{232}Th+C.

2.6.2 Changing Activity Input Units

Input units are the activity units that are used to enter the initial activities for the parent radionuclide mixture. DandD converts activities entered in these units into activities in the default units for each scenario. These conversions are shown on the radionuclide mixture dialog next to the initial activity entry box list. Clicking on the Change Units box will cause a window to appear.

The currently selected input units will be highlighted. After selecting the input units, click on the OK button or press the Enter key to keep the changes. Click on the Cancel button or press the Escape key to discard the changes.

2.6.3 Importing Concentrations

Both the Drinking Water and Residential scenarios include an import concentration capability. With this capability the two scenarios use a data file for inputting soil and/or groundwater radioactivity concentrations. Click on the Import... button to bring up a Windows File Dialog box to select the data file that contains the input concentrations. For a description of the expected format of the data file and error messages see Appendix E.

2.7 Basic Parameter Definition

Once a radionuclide mixture has been entered a scenario may be executed. Parameters for each scenario have default values that produce generic dose estimates for Level 1 screening. DandD allows variable parameters to be edited to replace default parameter values with site-specific values for Level 2 screening. Edit variable parameters by clicking on the Parameters check box for a specific scenario, or by pressing the Enter key when the check box is highlighted.

Use the Parameters window to edit variable parameters such as the time step information for the execution of a scenario or to add site specific information to the generic modeling.

Each parameter for the scenario is shown with name, its range of allowable values, default value, and units. The edit boxes contain the current value and, initially, these are all set to the parameter defaults. A description of the currently selected parameter is shown at the bottom of the dialog box. Use the up and down arrow keys to move through the list to highlight and edit the parameter values

or use the scrollbar and mouse.

All scenario parameters, including constant parameters, are shown in the parameter list. Constant parameters have "Constant" in the range column and their associated edit boxes are disabled. Use the Parameter Type dropdown list to select the type of parameters to edit for the Residential scenario. Default values for the parameters are listed in Appendix D.

After entering the parameter values click on the OK button or press the Enter key to keep the changes. Click on the Cancel button or press the Escape key to discard the changes.

2.8 Editing Tables

Some tables that are listed by element may be edited for the Drinking Water and Residential scenarios. For these tables, a table entry may be edited if an isotope of the element is present in the parent radionuclide mixture, or found in the progeny mixture of those parent radionuclides.

Only one entry for each element that may be edited will be presented, even though there may be multiple radionuclides that are isotopes of that element. Default values are assigned to each table entry from the associated table in Appendix D (Tables D.19 through D.22). Access these tables from the main session window by pressing the Tables checkbox or by selecting the Scenarios menu option Edit Tables.

The table in the Drinking Water Partition Coefficients Window may be edited for both the Drinking Water and Residential scenarios. It contains the partition coefficients (Kds) which are used to develop the leach-rate constants between the surface-soil layer, the unsaturated soil layer, and the aquifer in the three-box water-use model.

Use the TableType dropdown list to select the type of table to edit for the Residential scenario. The following tables may be edited for the Residential scenario:

Fish Bioaccumulation Factors: The transfer of activity from water to edible parts of fish is defined by fish bioaccumulation factors. These factors are the ratio of radionuclide concentrations in fish to the same radionuclide concentrations in water.

Soil-To-Plant Concentration Factors: These describe the amount of activity from contaminated soil that is concentrated in food crops and animal feed crops.

Animal Product Transfer Factors: These describe the amount of activity from soil that is transferred into animal products.

Use the View/Print button to bring up the list of parameters in a text viewing window. From this window the entire list of parameters and their current values may be printed. After entering the table values, click on the OK button or press the Enter key to keep the changes. Click on the Cancel button or press the Escape key to discard the changes.

2.9 Executing Sessions/Scenarios

Execute Session window allows choosing the modeling code execution options. These options are applied to <u>all</u> scenarios that are selected for the current execution.

Each scenario that will be executed is shown with an estimate of the size of the output that will be produced by its execution. The names of scenarios that are not selected will be grayed out, and those scenarios that are selected, but have not had a radionuclide mixture specified for them yet, will be followed by the text Not Edited.

A radio button group allows specifying whether or not data for graphics should be produced. Notice that the size of the output produced by a session will change depending on whether graphics data is included or not. Other options for the modeling code are toggled on or off by selecting/deselecting the name of the option in the options list. Notice that the text of the description will change by toggling these options on and off.

The following modeling code options are implemented:

Combine Progeny: If this flag is turned on, implicit progeny doses will be included in the parent progeny dose when reported. Otherwise, implicit progeny doses will be reported separately.

Distribute Progeny: This flag affects how initial activities of equilibrium chains selected in the radionuclide mixture are entered. If this flag is turned on, the initial activity entered for the parent radionuclide will be distributed across all of the progeny. Otherwise, the initial activity entered for the parent radionuclide should be

the activity of the parent by itself. Activities for the progeny initially in equilibrium with the parent will be calculated by the modeling codes. Equilibrium chains are identified by a +C following the radionuclide name.

After selecting the modeling code options for an execution, click on the OK button or press the Enter key to execute the session. To cancel the execution, click on the Cancel button or press the Escape key.

2.9.1 Session Execution Status

The Session Execution Status window will be displayed during a session execution. This window allows monitoring the status of the currently running session.

The window shows the name of the session and scenario currently executing, along with the time that has elapsed. A box, labeled Modeling Code Messages, shows a log of the messages that are coming from the modeling codes.

Execution of the modeling codes may be stopped at any time by clicking on the Abort button with the hourglass pointer. A Message Box will then appear asking for verification.

After execution of the session, the option to review the modeling code messages will be presented. To return directly to the main session window click on the No button. Click on the Yes button to review these messages and the Abort button will turn into a Done button. Press the Done button on the Session Execution Status window to return to the main session window.

2.10 Text Reports

The modeling code output file contains all the input and output produced by the modeling codes that implement the methodology. Text reports are a user-defined combination of reformatted sections of the modeling code output file.

Selected text reports are automatically generated after each execution of the associated scenario. When a new report type is defined for a scenario it is initially selected. If necessary, a report type may be deselected to conserve system resources. This should not normally be required.

2.10.1 Selecting/Deselecting Text Reports for Automatic Generation

After clicking on the Text Reports checkbox for a scenario, the Text Reports window is displayed. All of the reports defined by the user for the scenario are listed in the list on the left and all of the selected reports are in the list on the right. The arrow buttons between the lists permit selecting or deselecting a report. To select a report, highlight the report name on the All Defined Reports list and click on the arrow button pointing to the right. To deselect a report, highlight its name in the selected list and click the arrow button pointing left. The output from the modeling codes is always generated and is listed as the first text report labeled "FORTRAN output."

2.10.2 Viewing Text reports

To view a selected text report, highlight its name in the selected report list and click on the View Report button or double-click on its name in the selected report list. A viewing window will be displayed with the text of the report displayed.

Viewing Window

The viewing window allows viewing the text of a report but it does not allow editing. Use the scrollbar to scroll through the text or use the keyboard. The viewing window responds to the following keystrokes:

Home
: Moves to the beginning of the text file

End
: Moves to the end of the text file

Page Up
: Moves text up one page

Page Down
: Moves text down one page

Up Arrow
: Moves text up one line

Down Arrow
: Moves text down one line

The viewing window menu contains the following File menu commands:

Close
: Closes the viewing window.

Save As

Saves the text file to the specified filename.

Print
Print the report.

Print Preview
Shows how the printed output will look.

2.10.3 Defining and Editing Text Reports

To define a new text report or edit an existing one, click on the Define Reports button. A dialog window will be displayed that allows editing, deleting, or creating a new report type for a scenario.

2.10.3.1 Defining Report Types

The list of existing report types defined for the current report group is displayed. Create a new report type by pushing the New button. Edit an existing report by highlighting the report type name on the list and pushing the Edit button or by double clicking on the report name. Delete a report type by highlighting the report type name on the list and clicking the Delete button.

2.10.3.2 Editing a Report Type

A text report contains sections that define the input to and output from an execution of the modeling codes for a particular scenario. Input and output sections for the report may be chosen by checking the checkboxes labeled Include Input or Include Output. A list of subsections are available for each of these sections. The input subsections will be the same for each scenario. The following are the input subsections available for all scenarios.

2.10.3.2.1 Input Section

Execution Options
Shows a description of the options that were selected for execution. For example, this subsection describes the status of the Combine Progeny flag.

Initial Activities
Shows the initial activities entered for each parent radionuclide in the mixture.

Code-generated Activities
Shows the initial activities generated by the modeling codes for all radionuclides (including the progeny) in the mixture.

Edited Parameters
Shows all variable parameters that have been changed from their default values.

All Parameters
Shows all variable parameters; a default value is shown if the parameter has not been changed.

2.10.3.2.2 Output Section

The output subsections will be different for each scenario. The following are the output subsections available for each of the scenarios.

Building Occupancy, Building Renovation and Residential:

Max Occurrence
Shows the maximum TEDE and the year it occurred.

Max Pathway Doses
Shows a table of the dose contributions and percentages of each exposure pathway to the maximum annual TEDE.

Max Radionuclide Doses
Shows a table of the dose contributions and percentages of each radionuclide to the maximum annual TEDE.

Max Radionuclide Pathway Doses
Shows a table of the dose contributions and percentages of each radionuclide to each exposure pathway.

Drinking Water:

Max Occurrence
Shows the maximum TEDE and the year it occurred.

Max Radionuclide Doses
Shows a table of the dose contributions and percentages of each radionuclide to the maximum annual TEDE.

Organ Doses
Shows organ doses and critical radionuclides at maximum TEDE.

2.10.3.2.3 Setting Output Dose Units

Output units are the units of dose that will be used in the

report. DandD converts doses output in the modeling codes default units to the selected output units for the doses in this report. The currently selected output units will be highlighted. Change the units by clicking on the unit name in the list.

After editing the report type click on the OK button or press the Enter key to keep the changes. Click on the Cancel button or press the Escape key to discard the changes.

2.10.4 Text Report Filenames

Text reports are stored in different files depending on user selection. See Section 2.1.2 for the topic on where text files are stored.

2.10.5 Using a Text Editor or Word Processor

The original text reports generated by DandD should not be edited, but may be imported into a text editor or word processor to produce a new document. Graphics files may also be imported into a word processor that has the capability of manipulating Windows bitmap files (.bmp files). Use the Save As function from the Viewing window (Section 2.10.2) to save a report as a text file.

2.10.6 Report Groups

DandD stores report types in groups. There is a default report group that is used when the program starts up for the first time; using this group for all report types is fine. Other report groups may be created if the number of reports gets large or they are naturally grouped in some way; for example, multiple report types for reporting to different regulatory agencies. Each report group has a file that contains the list of report types in that group. This file is identified with an .rgp extension.

Report groups can only be manipulated from the main application window, i.e. when there is no open session. The Report Group submenu (Section 2.4.3.2) of the Groups main menu contains all the commands associated with report groups. To create a new report group, select the New menu option. A Windows Open File Dialog will be displayed in which a report group filename is entered. The location of the report groups may be changed by selecting different drives and/or directory paths. To open or delete an existing report group, select the Open or Delete menu options. Select the report group name using the Windows Open File Dialog.

The reports selected for a session will still be selected when report groups are switched. Reports are selected by count, so if the session has the first and third report types selected in one report group they will be selected in all report groups. This means that selected report types may not exist in a particular report group; these are just ignored.

DandD uses an initialization file, nrc.ini. The state of the program is saved in the initialization file each time it exits; and each time it starts up the state from the last execution is restored. Part of the state of the program that is saved is the current report group. DandD will attempt to open this report group each time it starts. If the report group does not exist anymore, the default report group will be opened instead. The name of the current report group is displayed on the right side of the status line. The full path to the current report group in the program information window may be displayed using the Program Information menu option of the main application File menu.

2.11 Graphics Reports

When clicking on the Graphics Reports checkbox for a scenario, the Graph Choices window is displayed. Graphics produced by DandD show the time history of doses for radionuclides, exposure pathways and/or total annual maximum TEDE or percentages of those doses at a particular time step. Select the desired type of graph by clicking the left mouse button on the associated Dose Graph radio button; Over Time for doses over time or Percentage for dose percentages at a single time step. The dose over time graphs are only available if there is more than one time step produced in the history file. The modeling codes will produce a history file that contains doses at intervals specified by the variable parameters described in Section 2.11.3.

2.11.1 Dose Over Time

Dose Over Time graphs show radionuclide, pathway and/or TEDE doses over time. Select the desired doses to graph directly by selecting the name on the radionuclide or pathway dose lists. Or click on a checkbox to select multiple doses, such as all the Parent Radionuclides. Up to 25 sets of doses can be shown on a graph; these sets may be any combination of radionuclides, exposure pathways and/or total annual TEDE doses.

The range of the time history to be graphed may be selected by entering the first and last time steps into the

edit boxes at the top of the window. Initially, the time step graph range is set to include all time steps. The time that DandD takes to produce a graph can vary widely depending upon what is selected, such as a large number of time steps or radionuclides. Limit the number of time steps graphed within the range by graphing every nth time step. Two or more time steps must be included for graphing time histories.

2.11.2 Dose Percentages

Percentage graphs show all the radionuclide or all the pathway doses as a percentage of the total TEDE at a particular time step. Select All Radionuclides or All Pathways doses by clicking the left mouse button on the associated radio button. Time steps may be entered at which to show the percentages. The default is the time step at which the maximum TEDE occurred.

When finished click on the OK button or press the Enter key to generate the specified graph. Click on Cancel or press the Escape key to cancel this action.

2.11.3 Controlling Graphics with Time History Parameters

The range and number of time steps produced in the modeling code history file may be controlled by using the time step variable parameters defined for each scenario. The time step variable parameters are Scenario Period, Start Time, End Time, DT and Skip.

Scenario Period
: The duration of the calculation period in days.

Start Time
: The start time of the calculations in number of days from today. Use this parameter to start the modeling codes some time in the future.

End Time
: The ending time of the calculations specified as number of days from the start time. Use this parameter to change the length of time the scenario is run.

DT
: The delta time step size. This parameter sets the number of days between dose calculations. Changing this parameter affects the number of output values in the dose history file. Use this parameter in combination with the end time of the scenario to produce a more detailed dose history for graphical displays. The default for this parameter is one year or 365.25 days and care should be taken when changing this to make sure that the value chosen creates even time steps in relation to the start and end times.

Skip
: Doses will be written to the history file every n time steps. Use this parameter to limit the number of time steps written to the dose history file for graphical displays.

2.11.4 Graph Window

This section describes the menu options available while using the Graph window. See Chapter 3 for a detailed discussion of the graphics capability of the DandD software.

2.11.4.1 File Menu

Close
: Closes the current graph window.

Export
: Saves the current graph in Chart FX format.

Import
: Loads a graph from a Chart FX formatted file.

Print
: Print the current graph.

Print Setup
: Selects a printer and printer connection.

Page Setup
: Sets up the pages for printing.

2.11.4.2 Edit Menu

Clipboard Copy Bitmap
: Copies the current graph window to the Clipboard as a bitmap.

Clipboard Copy Data
: Saves the current graph in Chart FX format.

Regraph
: Allows user to reselect the graph type and graph data.

UseTemplate

Loads graphing preferences from a template file.

Save Template

Saves graphing preferences in a template file.

3. Advanced Topics

3.1 Manipulating Graphics

3.1.1 Introduction

A product called Chart FX™ [Software FX, Inc., 1994] was incorporated into DandD for processing graphics and charts. A chart is a graphic representation of numeric data. Chart FX offers approximately 50 chart types (including 2D and 3D), although some of these types are not enabled in DandD. See Section 3.1.3 for a description of the charts provided. Besides the types of charts provided, users can add custom formatting to obtain the kind of charts they need. Options that can be accessed from the main graphics window are:

Menu Bar. Used to do file manipulation and printing. It is a standard Windows menu and allows the user to execute different commands from the Main Graphics window. See Section 3.1.2 for a description of the functions available.

Tabbed Dialog. Allows access to all features of the graphics library including those found in the Toolbar. A complete description of each tab is found in Section 3.1.4.

Rotate Dialog. Allows a 3-D chart to be rotated around the x-y axis. See Section 3.1.5 for more information.

Toolbar. Gives the user access to most features using a pointing device such as a mouse. It is generally used for common commands and features for all types of charts. For more information refer to "Working with the Toolbar" in Section 3.1.6.

Palette Bar. Permits the user to change any color used in a chart, including the backgrounds and colors used in the data markers of the series. Chart FX provides two palette bars; one embedded in the Toolbar (Palette Icons) and the standard palette bar which is shown at the top of the chart control. For more information refer to "The Palette Bar" in Section 3.1.7.

Pattern Bar. Similar to the Palette Bar. Instead of dragging colors to the desired area of the chart, the user can drag a specific pattern to a series of the chart to change it from a solid color to another pattern. For more information refer to "The Pattern Bar" in Section 3.1.8.

Data Editor. Gives the user the ability to manually change any of the values or legends passed to Chart FX, so the chart can be modified as needed. For more information refer to "Working with the Data Editor" in Section 3.1.11.

3.1.2 The Main Graphics Window

The Main Graphics window uses the default graph type that is displayed when the user selects the Graphics Reports function from the Main Session window. The functions available on the Menu Bar are described in this section.

3.1.2.1 File Menu

The File Menu provides the following functions:

Close. Closes the current graph window and returns to the Main Session window.

Export. Saves the current graph in Chart FX format.

Import. Loads a graph from a Chart FX formatted file.

Print. Prints the current graph.

Print Setup. Selects a printer and printer connection.

Page Setup. Sets up the pages for printing.

3.1.2.2 Edit Menu

The Edit Menu provides the following functions:

Clipboard Copy

Bitmap. Copies the current graph window to the Windows Clipboard as a bitmap.

Data. Saves the current graph in Chart FX format to the clipboard.

Regraph. Allows the user to reselect the graph type and graph data.

Use Template. Loads graphing preferences from a template file.

Save Template. Saves graphing preferences in a template file.

3.1.3 Types of Charts

Chart FX supports different types of charts. In this section each of these types and their variations will be presented. Instead of providing charts with formats included, Chart FX lets the user create a standard chart type and then add custom formatting to that chart. The basic charts supported are:

Area. Shows the relative importance of values over a period of time.

Bar. Shows variations over a period of time. It includes Column and Horizontal Bar charts.

Cubes. Shows the relative importance of values over a period of time. Instead of bars it has cubes as data markers.

Doughnut. Shows the relationship or proportions of parts to a whole. Same as a pie chart but with a hole in the middle.

Line. Shows trends or changes in data over a period of time.

Pareto. Often used in statistical areas. It has bars showing a variation over a period of time and a cumulative curve indicating the proportion accumulated by each column in the chart.

Pie. Shows the relationship of parts to a whole. This is the default chart for DandD when plotting a single time step.

Polar. Shows the relative importance of values over a period of time.

Scatter. A point chart that shows the relationship or degree of relationship between numeric values in several series, or plots two groups of numbers as one series of x-y coordinates. This is the default chart for DandD when plotting a time history of the doses.

Spline. Similar to Line, except the lines curve to pass through each point or data marker. This type is also known as fit-to-curve.

Surface. Shows a "rubber sheet" stretched over a 3D bar chart. It is useful for finding the best combinations between two sets of data. Shows relationships between large amounts of data that otherwise would be difficult to see.

3.1.3.1 Area Chart

Shows the relative importance of values over a period of time. This chart has several variations in the way it can be displayed and supports special 3D effects, Zoom In and rotation capabilities. The variations include the following:

Simple Area. Shown with drop down lines in each vertex. each series is viewed "as is."

3D Area. Same as Simple Area, except the drawing is three-dimensional.

Stacked Area. Shown with drop down lines at each vertex. Each series is plotted on top of the previous one.

3D Stacked Area. Same as Stacked Area, except the drawing is three-dimensional.

100% Stacked Area. Shown with drop down lines at each vertex. Each series is plotted on top of the previous one, calculating that all series will add to 100%.

3D 100% Stacked Area. Same as 100% stacked area, except the drawing is three-dimensional.

3.1.3.2 Bar Chart

Shows variation over a period of time and has several variations in the way it can be displayed. Bar charts can be displayed in both vertical (column) and horizontal formats and support special 3D effects, Zoom In, and rotation capabilities.

Simple Column Chart. Shows the data markers with a space between categories.

Color Each Bar. Shows the data markers in different colors (or patterns) with a space between categories.

Stacked Bar. Each series is positioned above the previous one, adding its value to the stacked bar.

100% Stacked. Each series is positioned above the previous one, adding its value to the stacked bar and calculating the percentage that each series represents of the whole part.

Joined Bars. Same as Simple Column chart, only there is no gap between series (or bars).

3D Bars. Same as Simple Column chart, except the bars are drawn three-dimensional.

3D Stacked. Same as Stacked, except the bars are drawn three-dimensional.

3D Joined Bars. Same as Joined Bars chart except the bars are drawn three-dimensional.

3D 100% Stacked. Same as 100% stacked, except the bars are drawn three-dimensional.

3D Clustered Bars. Every series has its own row in the chart.

3D Clustered/Joined. Same as 3D clustered except the bars are drawn together, with no space between categories.

3D Show-Zero. This type of chart is used when there are negative (less than zero) values in the chart. The starting point of all bars is zero. Therefore, the chart will have bars that go up and down (depending on their value.

Simple Horizontal Bar. Shows the data markers with a space between categories.

Stacked Horizontal Bar. Each series is positioned above the previous one, adding its value to the stacked bar.

100% Stacked Horizontal. Each series is positioned to the right of the previous one, adding its value to the stacked bar and calculating the percentage that each series represents of the whole.

3.1.3.3 Cube Chart

Refer to Bar Charts in Section 3.1.3.2 for the different variations supported in Cube charts. Cube Charts support Zoom In, Special 3D effects and Rotation.

2D Cube. This type of chart is very similar to a bar chart, but instead of a bar, a cube is displayed.

3D Cube. Same as 2D cube but in 3D mode.

3.1.3.4 Doughnut Chart

Shows the relationship or proportions of parts to a whole. Same as a pie chart but with a hole in the middle. Doughnut charts do not support special 3D effects or rotation capabilities.

Standard Doughnut. Same as a pie chart but with a hole in the middle of the chart.

Exploding Doughnut. The user can drag slices out of the doughnut.

3.1.3.5 Line Chart

Shows trends or changes in data over a period of time. Variations of this chart are:

Lines Only. Specifies that points or data markers are not shown in the chart.

Lines and Data Markers. Specifies that points or data markers are going to be shown in the chart. These points can be displayed in several forms. These forms are; rectangles, circles, triangles, marbles and cubes (3D-only).

Show Values. Specifies that at each vertex or data marker, the corresponding value of the point will be shown above the data marker.

3D Lines. Specifies that each line will be drawn as a three-dimensional figure.

Step Lines. The lines are painted as steps in every point. Also supported in 2D mode and applicable to scatter charts.

3D Clustered Lines. Specifies that each line will be drawn as a three-dimensional figure and each series have its row.

3.1.3.6 Pareto Chart

Often used in statistics, this has bars showing a variation over a period of time and a cumulative curve indicating the proportion accumulated by each column in the chart. These charts support special 3D effects, zoom, and rotation capabilities.

2D Pareto. Shows the cumulative line and percentage axis on the right of the drawing.

3D Pareto. Same as 2D, except the drawing is three-dimensional. The bars can be displayed in the same or different colors.

2D Pareto Joined. Shows the cumulative line and percentage axis on the right of the drawing, and the bars appear together with no space between categories.

3.1.3.7 Pie Chart

Shows the relationship or proportions of parts to a whole. X axis Rotation is also supported in this type of chart. This chart has several variations, in terms of how it can be displayed. These variations are:

Standard Pie. Shows the relationship or proportions of parts to a whole as a pie.

2D Exploding Pie. The user can drag one or more slices out of the pie.

3D Pie. Same as standard Pie, except it is a three-dimensional drawing.

3D Exploding Pie. The user can drag one or more slices out of the pie.

3.1.3.8 Polar Chart

3D Polar charts support Zoom In, Special 3D effects and rotation.

2D Polar. The user can control lines and its colors and even the marker size.

3D Polar. Same as 2D polar but in 3D mode.

3.1.3.9 Scatter Charts

A point chart that shows the relationship or degree of relationship between numeric values in several series, or plots two groups of numbers as one series of x-y coordinates. These charts support special 3D effects, Zoom In and rotation capabilities.

2D Scatter. Shows the relationship or degree of relationship between numeric values in several series.

Scatter with lines. Scatter charts can also be shown with connecting lines.

3D Scatter. Same as 2D Scatter, except points are shown in 3D. Uses the cube type point.

3D Clustered Scatter. Same as 3D Scatter except each series is positioned in a different row.

3.1.3.10 Spline Chart

Similar to a Line chart, except that the lines curve to pass through each point or data marker. This chart has several variations, in terms of how it can be displayed. These variations are:

Curve and Data Markers. Specifies that points or data markers are going to be shown in the chart. These points can be displayed in several forms. These forms include rectangles, circles, triangles, marbles and cubes (3D-only).

Curve Only. Specifies that points or data markers are not shown in the chart.

Show Values. Specifies that at each vertex or data marker, the corresponding value of the point will be shown above the data marker.

3D Curve. Specifies that each curve will be drawn as a three-dimensional figure.

3D Clustered Curves. Specifies that each series will have its own row. This option can be accessed from the toolbar. For an example refer to the 3D Clustered Lines sample in Section 3.1.3.5.

3.1.3.11 Surface Chart

Useful for finding the best combinations between two sets of data. Shows relationships between large amounts of data that otherwise would be difficult to see. Surface charts also support Zoom In and Rotation.

2D Surface. Same as 3D but displayed in 2D mode.

3D Surface. Colors are assigned depending on each point height (y value) in the chart).

3.1.4 Tabbed Dialog

Chart FX provides a Tabbed Dialog which allows access to all features of the graphics library, including those found in the Toolbar. This section contains a complete description of each tab. The dialogs provided are General, Series, Scale, 3D View, and Titles. The 3D View and Titles Tabs are the same as explained in the Toolbar section of this manual.

3.1.4.1 General Tab

The General Tab Dialog allows the user to set and customize general settings for the chart. Note that all settings apply to the entire chart including all of its series. The functions contained in this dialog are as follows:

3.1.4.1.1 General Settings Section

Gallery Type. For changing chart types. Same as

Toolbar Gallery Icon.

Cluster. To (de)activate 3D clustered charts. Same as Toolbar Cluster Button.

3.1.4.1.2 Appearance Section

Point Type. Selects marker shape.

Point size slider. To select marker size for all of the series in the chart.

Marker Volume. To control the volume that every marker occupies; sliding to the right will cause the markers to appear together (i.e. joined bars).

Stacked Icon. To select stacked style of the chart.

Grid Icon. To select grid styles in the chart.

Color Scheme. To select Solid Colors, Colors and B&W patterns.

Color Lines. Make 2D lines appear in same color as the series. (Polar, Spline and Line charts).

Show Lines. Make Lines appear in polar and scatter charts.

3.1.4.2 Series Tab

This section describes the Series Tab Dialog, which allows the user to set and customize settings for each series independently. All the features located in this tab are proprietary to the series selected in the series combo box (only when the Multiple Types or Multiple Shapes check boxes are checked). The functions contained in this section are:

3.1.4.2.1 Type Section

Series Combo box. This item allows the user to select the series to apply the changes to. When series legends are set, those labels will appear in this combo. Note that each series will contain a small colored rectangle indicating the actual color of selected series.

Show Values. This feature can be set either generally or independently to a certain series. When checked, Chart FX will show the appropriate value above the marker (in all chart types). If you have a Multiseries (i.e. 2 series) chart and want to have only one with the values shown, just check the MultiType option without changing the chart types for any of the series and check the show values check box for the appropriate series.

Borders. This feature can also be applied to all the chart or independently to a certain series. The effect of checking this button is that the borders surrounding markers (i.e. bar chart borders) will appear. This feature is very useful when working with conic or cylindrical charts and you want to get rid off the borders.

Multiple Types. Check this box if you want to perform a multiple type chart and with the series combo box and gallery combo select the type for each series accordingly.

Gallery Type. To be used only when Multiple Types is checked.

Show Points. This feature can be set either generally or independently to a certain series. When checked Chart FX will show the point. If you have a Multiseries (i.e. 2 series line chart) chart and want to show points for only one series, just check the MultiType option without changing the chart types for any of the series and check the show points check box for the appropriate series.

Area Lines. This feature will (de)activate lines shown in area charts.

3.1.4.2.2 Line's Shape Section

Series Colors Icon. Select the appropriate color for the series selected in the series combo box.

Multiple Shapes. Check this box when you want a Multiseries chart (i.e. 2 series) with different conic or cylindrical shapes. When checked you can select the appropriate series and apply a different form to its marker.

Line Thickness Edit. Type the thickness you want for 3D Line charts or use the slider located underneath this edit field to change the line thickness. Please note that the slider will serve as line thickness control when selecting a series that has a line type. When using the slider you will see a sample right below the slider indicating the thickness that you are applying.

Base selector slider. When having a BAR, CUBE, HI-LOW-CLOSE chart this slider will control number of sides in the base of the cylindrical shape applied to such series.

Conic check box. When checked you will obtain a conic form to BAR, CUBES or HI-LOW-CLOSE charts. In the sample provided this check box is hidden since the

chart type selected does not support conic charts. Nevertheless, when having any of the types mentioned above this check box will appear automatically.

3.1.4.3 Scale Tab

Axis selection. Check the appropriate axis that you want to customize. The X axis is only available on scatter charts.

Minimum/Maximum. Type the desired minimum and maximum for the axis selected. If the number if greater than maximum or less than minimum the chart will be clipped.

Scale Unit. Chart FX automatically selects the scale unit used in any of the axes. This scale unit is usually used when displaying big numbers in any of the axes. (i.e. 1/1.000.000 which means that every tick of the axis represents 1 million units).

Automatic/Fixed Gap. Chart FX automatically selects the gap used in any of the axes. This gap refers on the tick scale for the axis (i.e. if you have a maximum in the Y axis of 100 and want Chart FX to tick this y axis every 10 you will type this number in this edit field).

Show Zero axis. When checked and having negative values in the chart, the zero axis will be shown and the markers will be shown up and down depending on their value.

Linear or Logarithmic. The user can choose between a linear or logarithmic scale. When using log scale the base used can be changed (Default Log base used is base 10).

3.1.5 Rotate Dialog

Chart FX provides an intuitive way to rotate any kind of chart. By accessing the rotation dialog, the user can set any x or y axis angle to view that chart in different perspectives. The rotation dialog can be accessed via the menu or toolbar.

3.1.5.1 Rotation

To apply the rotation angles follow these steps:

1. Access the rotation dialog
2. Check the Full 3D View box
3. Drag the marbles to desired angles or enter the desired angles in the fields.
4. Verify the desired view shown in the sample rotation frame
5. Click on the "Apply" Button to modify the actual angles in the background chart while remaining in the rotation dialog, or OK button to apply these angles and exit the rotation dialog.

To turn off the rotation angles, follow these steps:

1. Access the rotation dialog
2. Uncheck the Full 3D View box
3. Click on OK to exit the dialog.

3.1.5.2 Controlling the 3D Depth

The Depth of the 3D chart can be controlled with the sliding control located underneath the rotation preview. When it is dragged to the right it will increase the Depth (in comparison to the chart width). Sliding it to the left will decrease the chart depth.

3.1.5.3 Shadows

This dialog contains the shadows check box which allows the user to turn on/off the shadow for the markers in 3D mode. Turning it off will cause the back side of the markers to paint in the same color assigned to the front of the marker. Default = On.

3.1.6 Working with the Toolbar

Another feature that can be accessed from the main graphics window is the Toolbar where the user can modify the chart characteristics with a single click of the mouse. This tool provides the user with the most commonly used commands in a chart, such as Chart Gallery, interaction with the clipboard, 3D effects, rotation, etc.

Chart FX 3.0 Toolbar has been enhanced to support floating capabilities. To make the toolbar float in the main graphics Window the user must do a double click in any part of it and drag the frame to adjust the desired size for the toolbar. To return the toolbar to its default position and display (Top), just drag it to the top of the main graphics window and it will return to its original position.

The Toolbar provides commands for 5 major areas which are described in the following sections.

File Commands. These options provide single click command execution to handle files (including printing) and interact with the clipboard.

Gallery Commands. Allows the user to change from one chart type to another.

Embedded Palette. Allows the user to select a color and drag & drop it in any part of the chart (Series Colors and backgrounds.

3D Effects. Provides the user with an interface to handle 3D effects including rotation and clustered charts.

Legends and Grids. Allows the user to show or hide chart legends and grids.

Utilities. Provides the user with an interface to handle title modification, fonts, and other important utilities.

ToolTips. Controls how help text is displayed when positioned on a Toolbar icon.

3.1.6.1 Gallery Commands

Chart FX Toolbar provides the Types Graphical Icon combo box to select the different chart types supported in the library. This control works as a normal Windows combo box, just click on the icon which opens the Gallery and select the desired chart types from the listed types.

3.1.6.2 Embedded Palette

Chart FX Toolbar provides the Palette Graphical Icon combo box to select a color and drag & drop it in any part of the chart (Series Colors and backgrounds). Open the combo and select the desired color by clicking the left mouse button and holding it down, then dragging it to a desired location in the chart and drop it. Chart FX will automatically apply that color to the selected section. See Section 3.1.7 for a discussion of the Palettes.

3.1.6.3 ToolTips

Chart FX 3.0 supports ToolTips in its Toolbar, which displays the text in the corresponding button of the bar. This text can be displayed in a balloon help or inside a rectangle. This can be customized or deactivate this feature by pressing the right mouse button in the background of the toolbar (not over a button) and a menu will appear to select the desired setting.

3.1.7 The Palette Bar

Chart FX provides the user with a customizable Palette Bar to change the colors used in the chart area. The colors of the following items in a chart may be changed:

- Colors used in the series of a chart.
- Color used in the background area surrounding a chart.
- Color used in the wall of a 3D chart.
- Color used in the background containing a chart.
- Color used in the legends of an horizontal bar chart.

3.1.7.1 Working with the Palette Bar

To change the colors of these items in a chart, click on and drag the desired color to any portion or area of the item and drop it.

3.1.7.2 Customizing the Palette Bar

Although a default Palette Bar is shown in the main graphics window, the user may add other colors to this tool. To change any color of the Palette Bar, double-click any of the colors shown and the MS Windows colors dialog will appear. The user may choose any color from this palette, press the OK button, and automatically the Palette Bar will add that color to the position that was double-clicked.

Note: The changes made in the default palette are not saved unless the user executes the Save Template command (See Section 3.1.2.2).

3.1.8 The Pattern Bar

Chart FX provides a customizable Pattern Bar so the user can switch between patterns or solid colors to be used in the series of a chart. These patterns are also useful when printing the chart on a black and white printer. These changes between solid colors and patterns are referred to by Chart FX as Color Schemes. To change the patterns used in a chart, click and drag the desired pattern to any series of the chart.

3.1.8.1 Working with the Pattern Bar

A pattern has 2 colors:

Foreground Color. The color used to draw the pattern itself.

Background Color. The color used to draw behind the pattern.

The user can change both colors used in any series by dragging the color from the Palette Bar and dropping it on the desired series by doing the following:

1. If a color is dropped over a series, it changes the foreground color of the pattern. If this drop is made while pressing the <shift> key it changes the background color of the pattern.

2. To support Color Patterns the Color Pattern scheme must be activated in the Tabbed Dialog (Section 3.1.4.1). If the black and white Pattern scheme used a foreground color of black then the background color is white.

3.1.8.2 Customizing the Pattern Bar

Although a default pattern is shown in the Pattern Editor window, the user can change this and any other pattern. To change any pattern of this tool, double-click any of the patterns shown in the Pattern Bar and the Pattern Editor will appear. The pattern can be changed by adding or removing points from the pattern.

The changes made in the default Pattern Bar are not saved unless the user executes the Save Template command (See Section 3.1.2.2).

3.1.9 Zoom Capabilities

In certain applications or contexts, the user may want to zoom in on a section of the chart to view more detailed information. Instead of specifying a zoom factor to the chart, the user is able to select a region to be zoomed and Chart FX will take care of displaying that part of the chart in the rectangle that contains the chart. This feature is available through the zoom button located in the default toolbar.

Clicking on the toolbar Zoom In Button allows the user to select a region in the chart to be zoomed. Click the left mouse button on the upper left corner of the area to be enlarged, drag the pointer to the lower right of the area and release the left mouse button. In order to deactivate the zoom factor, press the zoom button again and the chart will return to its normal view.

3.1.10 Legends

Chart FX allows the user to change the text of legends (Series and Points) using the Data Editor (Section 3.1.11). Since these legends are separate, the user can decide which one to display and where to display them.

3.1.10.1 Dockable Legends

Series and point legends can be placed anywhere in the chart window. These can be positioned in usual parts in the chart (top, left, bottom or right) or the user can decide where to place them in a fixed position inside the chart. The term *dockable* means that the user can interact with these legends by double-clicking them or dragging them to a desired position in the chart, and the legend will try to best fit into the space into which its being dragged. For example, double-clicking the legend will cause it to convert to a floatable tool which can be dragged anywhere in the chart window.

Another example of the dockable options of the legends is when the legend is captured anywhere but on its caption and dragged near any standard location in the chart. It will fit into that location automatically. This feature is also available in the default Toolbar.

3.1.10.2 Menus on Demand

This feature allows the user access to a selection by right clicking inside any of the legends and accesses all of the necessary features to customize and position the legends anywhere inside the chart window.

3.1.10.3 Color Customizable

All the Drag & Drop features of the Palette Bar are also available to the legends. This means that the user can select a color from the Palette Bar and drop it in any of the legends (Points & Series) to change its default color.

3.1.10.4 Legend Internal Options

When accessing the options dialog for the appropriate legends the user can set the following:

Draw Border. When legend is fixed in a specific position in the chart window.

Draw Splitter. To resize the legend when it is dockable in a standard position. (Left, Right, Top, Left).

3D Frame. Surrounding the Legend.

Skip Empty. Legend will show only those points that have text assigned.

Multiline. Multiline legends (Word Break)

3.1.11 Working with the Data Editor

The Data Editor allows the user to alter the values plotted in the chart and to change the legends used on the x and y axes.

3.1.11.1 Editing Values

To edit any of the cells containing a value, double-click the left mouse button over the value. After the value is edited, move up and down using the up and down arrow keys on the keyboard. Move right and left by using the tab and <shift> tab keys.

3.1.11.2 Editing Legends

To edit any of the legends used in the chart, double-click the left mouse button on the desired text to edit it. After changing any of the values, the chart will be redrawn.

4.0 REFERENCES

Kennedy, Jr., W.E., and D.L. Strenge, 1992. *Residual Radioactive Contamination From Decommissioning: Technical Basis for Translating Contamination Levels to Annual Effective Dose Equivalent,* Vol. 1, NUREG/CR-5512, PNL-7994, U.S. Nuclear Regulatory Commission, Washington, DC.

Software FX, Inc., 1994, *ChartFX - User's and Programmer's Manual, Version 3*

Appendix A Differences Between Volume 1 and Release 1.0 of DandD

A.1 Introduction

NUREG/CR-5512, Volume 1, "Residual Radioactive Contamination From Decommissioning: Technical Basis for Translating Contamination Levels to Annual Effective Dose," [Kennedy and Strenge, 1992], provides the technical basis, associated equations, and data tables for translating residual contamination levels to annual dose for decommissioned sites. The NUREG/CR-5512 modeling is intended to be a screening tool for assessing potential doses from decommissioned sites based on a philosophy of moving from simple, prudently conservative, calculations toward more realistic simulations, as necessary. This modeling is part of a process that includes three increasingly precise methods for calculating screening model dose conversion factors (DCFs) for evaluating exposure to residual radioactivity in soil and structures. Level 1 screening uses very simple assumptions and default parameters that are intended to be prudently, but not excessively, conservative. Level 2 screening allows the user to adjust certain parameters to more closely approximate conditions at the site being evaluated. Level 3 modeling is very site specific and is beyond the scope of the screening methodology described in NUREG/CR-5512, Volume 1. For each screening level four scenarios are analyzed, including building occupancy, building renovation, drinking water, and residential.

In 1993, Sandia National Laboratories (SNL) developed codes to perform calculations using the NUREG/CR-5512 methodology to support the NRC's draft regulatory guide on release criteria for decommissioning, NUREG-1500 [Daily et al., 1994]. Subsequent to performing these calculations, SNL has developed DandD, a user-friendly software application that implements the NUREG/CR-5512 methodology. DandD incorporates the codes developed for the NUREG-1500 calculations within a graphical user interface. In the process of supporting the NUREG-1500 calculations and implementing and testing the DandD software, several problems and issues related to the original methodology in NUREG/CR-5512 have been identified. Consequently, several changes or corrections have been made for implementation of the NUREG/CR-5512 methodology in DandD. Generally, these changes address the following problems or issues:

(1) revisions to the equations published in NUREG/CR-5512, Volume 1;

(2) errors found in the data tables published in NUREG/CR-5512, Volume 1; and

(3) assumptions and default parameters used in the original NUREG/CR-5512 methodology that have been found to be inconsistent with the iterative screening philosophy of the methodology.

This appendix documents these problems and the corrections/solutions to these problems that have been implemented for Release 1.0 of DandD. Section A.2 describes equation nomenclature. Section A.3 of this report describes (1) the changes made to the equations originally published in NUREG/CR-5512, Volume 1, and (2) changes made as a result of the implementation and testing of DandD to address assumptions that resulted in inconsistencies between the intent of the NUREG/CR-5512 methodology and the originally published equations. Section A.4 identifies the corrections made to the data tables published in NUREG/CR-5512, Volume 1, to develop the data files for the Release 1.0 of the DandD code. Changes in the default parameter values are to be documented in Appendix D of this users manual and in the parameter analysis report to be published subsequently.

A.2 Nomenclature

The mathematical models originally described in Volume 1 of this report involve many equations with numerous parameters. Because of the complexity of the equations, a system of nomenclature was developed to represent symbols used for the parameters. This system of nomenclature is summarized in Section 2 of Volume 1. For ease of reference to understand the changes to the mathematical formulations that are presented in this appendix, the system of nomenclature is presented here in Table A.1. The system of nomenclature includes a definition of units used to permit a full dimensional analysis.

In general, terms beginning with "D" are dose or dose rate factors; "TEDE" are annual total effective dose equivalents; "AF" are ingestion-pathway committed effective dose equivalent factors; "C" are concentrations (per unit mass, volume, or area), or total activity of a radionuclide, as appropriate; and "λ" are radioactive decay rate constants. General subscripts encountered include "i" or "j" for parent or decay-chain-member radionuclides; "s" for soil; "w" for water; and "v" for

food crops (agricultural pathways).

In addition to the parameters listed in tables, a special notation is used for radioactive decay calculations. Decay operators are represented by A{}, S{}, R{}, and G{}, as defined in Appendix B of Volume 1.

A{} = Changes in parent and progeny activities or concentrations over time (i.e. radioactive decay and ingrowth).

S{} = time integrals of activity or concentration.

R{} = accumulation of deposited activity over a time period.

G{} = deposition, accumulation, and time-integration of a constant deposition rate (used for deposition from irrigation water onto plants).

The operations are performed on an initial array of chain member activities or concentrations for a specific time period. For example, the decay calculation is represented as follows:

$$A\{C_{\bullet},t_{\bullet}\}$$

where A{} = the operation of decay calculation (in appropriate units)

$C_{\bullet}$ = the array of chain members, activities, or concentrations (in appropriate units).

$t_{\bullet}$ = time period over which the decay occurs (in time units)

For additional information on nomenclature, the reader is referred to Volume 1 of this report.

Table A.1 Parameter Summary

Parameter	Description
A_{stj}	Concentration factor of radionuclide progeny j in soil for the current year per initial unit concentration of chain parent radionuclide in soil.
A_{wtj}	Average concentration factor of radionuclide progeny j in water for the current year per initial unit concentration of chain parent radionuclide in soil.
Af_{dj}	Committed effective dose equivalent per unit average concentration of chain member j (as a parent) in water from drinking water contaminated by radionuclide chain member.
AF_{fj}	Committed effective dose equivalent per unit average concentration of chain member j (as a parent) in water from aquatic food products contaminated by radionuclide chain member j.
AF_{fj}	Committed effective dose equivalent per unit concentration of chain member j (as a parent) in soil at the start of the growing season from agricultural products contaminated by radionuclide chain member j in soil.
AF_{wj}	Committed effective dose equivalent per unit average concentration of chain member j (as a parent) in water from agricultural products contaminated by irrigation water for radionuclide j.
C_{1j}	Total activity in box 1 of the three box ground-water model for radionuclide chain member j.
C_{2j}	Total activity in box 2 of either the three box or the multiple-layer-unsaturated-zone box ground-water model for radionuclide chain member j.
C_{3j}	Total activity in box 3 of the three box ground-water model for radionuclide chain member j.
$C_{1,n}$	Total activity in box 1 of the three box ground-water model for radionuclide chain member n.
$C_{2,n}$	Total activity in box 2 of either the three box or the multiple-layer-unsaturated-zone box ground-water model for radionuclide chain member n.
$C_{3,n}$	Total activity in box 3 of the three box ground-water model for radionuclide chain member n.
C_{kj}	Total activity in box k of the three box ground-water model for radionuclide chain member j.
C_{lj}	Total activity in box l o fthe multiple-layer-unsaturated-zone ground-water box model for radionuclide chain member j.
$C_{l-1,j}$	Total activity in box l-1 of the multiple-layer-unsaturated-zone ground-water box model for radionuclide chain member j.
$C_{l,n}$	Total activity in box l of the multiple-layer-unsaturated-zone ground-water box model for radionuclide chain member n.

Table A.1 Parameter Summary

Parameter	Description
$C_{m+1,j}$	Total activity in box m+1 (box immediately above the aquifer box) of the multiple-layer-unsaturated-zone ground-water box model for radionuclide chain member j.

Table A.1 Parameter Summary (Continued)

Parameter	Description
$C_{m+2,j}$	Total activity in box m+2 (aquifer box) of the multiple-layer-unsaturated-zone ground-water box model for radionuclide chain member j.
$C_{m+2,n}$	Total activity in box m+2 (aquifer box) of the multiple-layer-unsaturated-zone ground-water box model for radionuclide chain member n.
C_{rfCc}	Carbon-14 average concentration factor in forage crop at time of forage feeding resulting from resuspension and root uptake from soil for an average unit concentration of carbon-14 in ground water.
C_{rgCc}	Carbon-14 average concentration factor in stored grain at time of grain feeding resulting from resuspension and root uptake from soil for an average unit concentration of carbon-14 in ground water.
C_{rhCc}	Carbon-14 average concentration factor in stored hay at time of stored hay feeding resulting from resuspension and root uptake from soil for an average unit concentration of carbon-14 in ground water.
C_s	Initial activity in soil for a chain parent radionuclide.
C_{si}	Initial activity in soil for chain i parent radionuclide.
$C_{si}(0)$	Initial activity in soil for chain i parent radionuclide.
C_{swjt}	Activity concentration factor for transfer from soil to water for radionuclide chain member j at time t.
C_{ti}	Total activity in soil for chain i parent radionuclide.
C_{wfCc}	Carbon-14 concentration factor in fresh forage at time of animal consumption for unit initial concentration of carbon-14 in ground water.
C_{wfCd}	Carbon-14 concentration factor in soil at time of animal consumption for unit initial concentration of carbon-14 in ground water.
C_{wgCc}	Carbon-14 concentration factor in stored grain at time of animal consumption for unit initial concentration of carbon-14 in ground water.
C_{whCd}	Carbon-14 concentration factor in stored hay at time of animal consumption for unit initial concentration of carbon-14 in ground water.
C_{wwCc}	Carbon-14 concentration factor in water at time of animal consumption for unit initial concentration of carbon-14 in ground water.
CDG	Dust loading factor for gardening activities.
CDI	Dust loading factor for indoor exposure periods.
CDO	Dust loading factor for outdoor exposure periods.
DAR_i	Committed effective dose equivalent from ingestion of aquatic foods for radionuclide chain i.
$DEXR_i$	External dose for radionuclide chain i.
$DFER_j$	External dose rate factor for radionuclide chain member j from contamination uniformly distributed in the top 15 cm of residential soil.
DFG_j	Committed effective dose equivalent from ingestion of radionuclide chain member j.
DFH_j	Committed effective dose equivalent from inhalation of radionuclide chain member j from contaminated air.
DFO_{jo}	Committed effective dose equivalent to organ o from ingestion of radionuclide chain member j.
DGR	Committed effective dose equivalent from ingestion of home-grown food and animal products for a radionuclide chain.

Table A.1 Parameter Summary (Continued)

Parameter	Description
DGR_i	Committed effective dose equivalent from ingestion of home-grown food and animal products for radionuclide chain i.
DHR_i	Committed effective dose equivalent from inhalation of airborne soil and house dust for radionuclide chain i.
DIET	Fraction of annual diet derived from home-grown foods.
DIRR	Committed effective dose equivalent from ingestion of home-grown foods and animal products produced with irrigation water for a radionuclide chain.
$DIRR_i$	Committed effective dose equivalent from ingestion of home-grown foods and animal products produced with irrigation water fro radionuclide chain i.
DSR_i	Committed effective dose equivalent from secondary ingestion of soil for radionuclide chain i.
DWR_i	Committed effective dose equivalent from ingestion of drinking water and irrigated food for radionuclide chain i (also used for committed effective dose equivalent from ingestion of drinking water for radionuclide chain member j after modification of Eqn. 5.74).
F_r	Fraction of water removed from the aquifer box that is deposited on the surface layer by irrigation.
GR	Effective transfer rate for ingestion of soil and dust transferred to the mouth during the residential scenario.
$HOCDE_i$	Highest annual organ committed dose equivalent for radionuclide chain i from ingestion of drinking water.
$HOCDE_m$	Highest annual organ committed dose equivalent for a mixture of radionuclides from ingestion of drinking water.
J_i	Number of radionuclides in decay chain for radionuclide chain i.
L_{12j}	Rate constant for movement of radionuclide j between boxes 1 and 2 in the ground-water box model.
L_{23j}	Rate constant for movement of radionuclide j between boxes 2 and 3 in the ground-water three box model.
$\acute{L}_{23j}$	Rate constant for movement of radionuclide j between boxes 2 and 3 in the multiple-layer-unsaturated-zone ground-water box model.
$\acute{L}_{l-1,j}$	Rate constant for movement of radionuclide j between boxes l-1 and 1 in the multiple-layer-unsaturated-zone ground-water box model.
$\acute{L}_{l,l+1,j}$	Rate constant for movement of radionuclide j between boxes l and l+1 in the multiple-layer-unsaturated-zone ground-water box model.
$\acute{L}_{m+1,m+2,j}$	Rate constant for movement of radionuclide j between the box immediately above the aquifer and the aquifer box in the multiple-layer-unsaturated-zone ground-water box model.
M	Number of radionuclide chains.
P_d	Floor dust loading factor for residential scenario.
PF_{sji}	Pathway transfer factor providing uptake by humans per unit concentration in soil for chain member j caused by unit activity of radionuclide chain member i.
PF_{wji}	Pathway transfer factor providing uptake by humans per unit concentration in water for chain member j caused by unit activity of radionuclide chain member i.
$PPTF_{awC}$	Partial pathway transfer factor of animal product for carbon-14 due to irrigation.
RF_r	Resuspension factor for indoor activity in the residential scenario.
Q_d	Soil dry weight consumption rate as a fraction of the fresh forage consumption rate.
Q_f	Consumption rate of wet weight fresh forage by animal.
Q_g	Consumption rate of wet weight stored grain by animal.
Q_h	Consumption rate of wet weight stored hay by animal.
Q_w	Consumption rate of water by animal.

Table A.1 Parameter Summary (Continued)

Parameter	Description
$S\{A_{stj},t_{tg}\}$	Time integral of total soil activity for radionuclide chain member j over the gardening period from time t to time $t+t_{tg}$.
$S\{A_{stj},t_{tr}\}$	Time integral of soil total activity for radionuclide chain member j over the residential period from time t to time $t+t_{tr}$.
$S_{d3}\{C_{kj},t_y\}$	Time integral of drinking water scenario aquifer total activity for radionuclide chain member j over the time period t_y (one year).
SA_{TaC}	Specific activity equivalence of animal product and the specific activity of the total feed intake.
SFI	Shielding factor by which external dose rate is reduced during periods of indoor residence.
SFO	Optional shielding factor by which external dose rate is reduced during periods of outdoor residence.
$TEDED_i$	Annual total effective dose equivalent for the drinking water scenario for radionuclide chain i.
U_w	Ingestion rate of drinking water by humans.
V_{aq}	Aquifer volume.
V_g	Volumetric breathing rate for gardening activities for residential scenario.
V_{Ir}	Volume of water infiltrating through contaminated area in a year for the residential scenario water-use model.
V_{dd}	Volume of water used for domestic purposes during a year of the drinking water scenario.
V_{dr}	Volume of water used for domestic purposes during a year of the residential scenario.
V_{irr}	Volume of water used for irrigation during a year of the residential scenario.
V_{Td}	Total aquifer volume for the drinking water scenario.
V_{Tr}	Total aquifer volume for the residential scenario.
V_r	Volumetric breathing rate for indoor activities for the residential scenario.
V_{sw}	Volume of water in the surface-water pond used in production of aquatic foods.
V_x	Volumetric breathing rate for outdoor activities for the residential scenario.
W_f	Factor to convert mass of fresh forage from a wet-weight to a dry-weight basis.
c_{aq}	Concentration of radionuclides in the aquifer.
c_{sw}	Concentration of radionuclides in the surface-water pond.
$d_{n,j}$	Fraction of radionuclide n that decays to radionuclide j.
dgr_j	Partial committed effective dose equivalent fro ingestion of home-grown food and animal products for radionuclide chain member j.
$dirr_j$	Partial committed effective dose equivalent from ingestion of home-grown food and animal products produced with irrigated water for radionuclide chain member j.
f_{Ca}	Fraction of animal product that is carbon.
f_{Cf}	Fraction of wet fresh forage that is carbon.
f_{Cd}	Fraction of dry soil that is carbon.
f_{Cg}	Fraction of wet stored grain that is carbon.
f_{Ch}	Fraction of wet stored hay that is carbon.
m	Number of boxes in the unsaturated zone in the multiple-layer-unsaturated-zone ground-water box model.
t	Time.
t_{ca}	Time period over which animal product is consumed by humans.
t_{ff}	Time period over which animal is fed fresh forage.
t_{fg}	Time period over which animal is fed stored grain.
t_{fh}	Time period over which animal is fed stored hay.
t_{fw}	Time period over which animal consumes water.

Appendix B. Sample Problems

B.1 Introduction

When DandD is installed, a session group is copied that contains test case sessions that have been defined and documented. Running through a test case will help the user become familiar with the program and confirm that DandD is producing the same results as the documented test cases. For instructions on running a test case see Section 2.2.

The facilities described in the test cases were taken from the "Draft Generic Environmental Impact Statement in Support of Rulemaking on Radiological Criteria for Decommissioning of NRC-Licensed Nuclear Facilities" [NUREG-1496, 1994]. The baseline test cases assume the standard methodology presented in [Kennedy and Strenge, 1992] and default values listed in Appendix D. Additional test cases that go beyond the defaults are also included and contain a brief discussion of how the changes affect the results. All of the test cases only include the Residential scenario because it exercises all exposure pathways and is the most complex. The execution option Combine Progeny was turned on for all test cases. The results for some cases do not show a table for the radionuclide component of the total dose because there are no progeny for the parent radionuclide.

B.2 Reference Facilities

The test cases use reference facilities considered to be sufficiently representative of facilities licensed by NRC. They are divided into fuel cycle and non-fuel-cycle groups. A brief description of each type of facility is provided along with the parameters needed to set up the test cases for the DandD software and the results that were calculated.

Fuel cycle facilities include:

- power reactors
- non-power (research and test) reactors
- uranium fuel fabrication plants
- uranium hexafluoride conversion facilities
- uranium mills
- independent spent fuel storage installations (ISFSI)

Non-fuel-cycle facilities include:

- sealed source manufacturers
- research and development laboratories
- rare metal refineries

B.3 Nuclear Fuel Cycle Facilities

Nuclear fuel cycle facilities are those facilities involved in any of the steps leading to or resulting from the generation of electricity by controlled nuclear fission of uranium. The nuclear fuel cycle process consists of several steps in which uranium ore is processed into fuel elements. The facilities involved in these steps include uranium mills, uranium hexafluoride conversion plants, enrichment plants, and uranium fuel fabrication plants. The next step in the fuel cycle is the generation of electricity by nuclear power plants. The end of the fuel cycle consists of steps in which fuel removed from the reactor is stored and then disposed of in some manner. The facility considered here involved in this step is independent spent fuel storage installations (ISFSI's).

B.3.1 Nuclear Power Reactors

There are two major types of nuclear power reactors in the United States, pressurized water reactors (PWRs) and boiling water reactors (BWRs). The analysis described in Appendix C of [NUREG-1496, 1994] indicates that, within the common variations of contamination levels in nuclear power plants, the contaminants and contamination of a PWR approximate the levels for a BWR. Therefore, the reference power reactor in this case is a large PWR (1095 Mwe). The reference power reactor comprises five major structures: the reactor containment, the auxiliary building, the fuel handling building, the turbine building, and the control building. For the reference facility, it is assumed that there are areas of contaminated soil resulting from on-site spills.

The test case file names for this group are preceded by NPP and the contaminants consist of ^{60}Co, ^{90}Sr, and ^{137}Cs with initial activities of 1 pCi/gram. The results shown are for the residential scenario only and have the Combine Progeny switch turned on.

B.3.1.1 NPP Baseline Test Case

This test case is set up by selecting the three radionuclides shown below and using all default parameters.

Selected Radionuclides	Initial Activity
^{60}Co	1 pCi/gram
^{90}Sr	1 pCi/gram

^{137}Cs	1 pCi/gram

Results:

The peak dose of 7.12E+001 TEDE (mrem) occurred 3.00 years after license termination.

Table B.1 Pathway Component of Maximum Annual Dose for NPP Baseline Test Case

Pathway	TEDE (mrem)	Percentage
External	7.67E+000	10.77
Inhalation	9.51E-005	0.00
Agricult	6.05E+001	84.96
Soil	3.20E-003	0.00
Drinking	8.12E-002	0.11
Irrigated	4.84E-001	0.68
Aquatic	2.47E+000	3.47
Total	7.12E+001	100.00

Table B.2 Radionuclide Component of Maximum Annual Dose for NPP Baseline Test Case

Radionuclide	TEDE (mrem)	Percentage
^{60}Co	6.87E+000	9.64
^{90}Sr	5.68E+001	79.80
^{90}Y	2.60E+000	3.65
^{137}Cs	4.93E+000	6.92
Total	7.12E+001	100.00

B.3.1.2 NPP1 Contamination Under a House

This test case assumes that all of the contamination lies directly under a 2000 ft^2 house and turns off all pathways except for external exposure inside the house, soil ingestion, and inhalation inside the house.

Selected Radionuclides	Initial Activity
^{60}Co	1 pCi/gram
^{90}Sr	1 pCi/gram
^{137}Cs	1 pCi/gram

This test case is set up by selecting the three radionuclides specified above and changing the following parameters:

Time Outdoors	0.0000	days/year
Time Gardening	0.0000	days/year
Water Ingestion	0.0000	L/d
Cultivated Area	186.0000	m^2
Garden Diet	0.0000	
Diet - Fish	0.0000	kg/y

All of these parameters, except for the last two, are changed by clicking on the Parameters check box in the residential section of the main session window. To change the Garden Diet and Diet - Fish parameters select Scenarios, Edit Other Parameters, Residential, and Human Diet from the menu bar or click on the Parameters checkbox in the residential section of the main session window, click on the Parameter Type drop down list, and then click on Human Diet.

The basis for the changes in the default parameters are as follows:

The Time Outdoors is set to zero on the assumption that when a person is outdoors they are not exposed to contamination because the contaminated area is under the house

- The area is based upon a 2000 ft^2 (186 m^2) house
- The Time Gardening is set to zero because there is no contamination present in the garden
- The Water Ingestion rate is set to zero to turn off the drinking water pathway which assumes that the water comes from a municipal water supply
- The Garden Diet parameter is set to zero to turn off the irrigated and agricultural pathways because there is no contamination in the garden
- The Diet - Fish parameter is set to zero to turn off the aquatic pathway because there is no contamination in the pond

Results:

The peak dose of 5.79E+000 TEDE (mrem) occurred 1.00 year after license termination.

Table B.3 Pathway Component of Maximum Annual Dose for NPP1 Test Case

Pathway	TEDE (mrem)	Percentage
External	5.87E+000	99.95
Inhalation	1.47E-005	[illegible]
Agricult	0.00E+000	[illegible]
Soil	2.71E-003	0.05
Drinking	0.00E+000	0.00

Irrigated	0.00E+000	0.00
Aquatic	0.00E+000	0.00
Total	5.79E+000	100.00

Table B.4 Radionuclide Component of Maximum Annual Dose for NPP1 Test Case

Radionuclide	TEDE (mrem)	Percentage
^{60}Co	4.67E+000	80.76
^{90}Sr	1.95E-003	0.03
^{90}Y	8.15E-003	0.14
^{137}Cs	1.10E+000	19.07
Total	5.79E+000	100.00

B.3.1.3 NPP2 Contamination In the Garden

This test case assumes that the contamination is only in the garden. Therefore:

- The Time Indoors is set to zero to turn off exposure from inside the house
- The area is based upon a 2000 ft^2 ($186m^2$) garden
- The Garden Diet parameter is proportionately scaled against the default site area of 3800 m^2 as follows:
  .33* 186/3800 = .0162
- The Time Outdoors assumes that a person has the same behavior as in the default case and spends 104.44 days spread over 3800 m^2 of ground, but only 4.89% of that time is spent on contaminated ground: 186/3800 * 104.44 = 5.1121.

Selected Radionuclides	Initial Activity
^{60}Co	1 pCi/gram
^{90}Sr	1 pCi/gram
^{137}Cs	1 pCi/gram

This test case is set up by selecting the three radionuclides specified above and changing the following parameters:

Time Indoors	0.0000	days/year
Time Outdoors	5.1121	days/year
Cultivated Area	186.0000	m^2
Garden Diet	0.0162	

Results:

The peak dose of 4.01E+000 TEDE (mrem) occurred 5.00 years after license termination.

Table B.5 Pathway Component of Maximum Annual Dose for NPP2 Test Case

Pathway	TEDE (mrem)	Percentage
External	1.73E-001	4.33
Inhalation	2.95E-005	0.00
Agricult	3.64E-001	9.10
Soil	3.02E-005	0.00
Drinking	4.32E-001	10.77
Irrigated	3.88E-002	0.97
Aquatic	3.00E+000	74.83
Total	4.10E+000	100.00

Table B.6 Radionuclide Component of Maximum Annual Dose for NPP2 Test Case

Radionuclide	TEDE (mrem)	Percentage
^{60}Co	1.74E-001	4.34
^{90}Sr	5.70E-001	14.23
^{90}Y	3.09E-002	0.77
^{137}Cs	3.23E+000	80.66
Total	4.10E+000	100.00

B.3.2 Non-Power Reactors

Non-power reactors are generally divided into two classes, research reactors and test reactors. The reference research reactor is considered to be a 1 megawatt nuclear training and research facility. The facility is made up of a reactor tank, a core structure, and a TRIGA type control system.

The reference research reactor comprises these major structures:

- reactor building - housing the reactor and support area
- annex - housing a hot laboratory and hot cell
- heat exchanger building - housing a water purification system, water pumping systems,
- and air compressor system
- pump house - housing a liquid waste retention tank
- cooling tower

Test reactors are larger facilities than research reactors and the reference test reactor considered is a 60-megawatt test reactor, light water moderated and cooled, used in testing materials for certain applications. The system is

made up principally of the test reactor vessel (containing the nuclear core and experimental beam tubes) and the reactor water recirculation system.

The reference test reactor comprises these major structures:

- reactor building - housing the test reactor
- hot laboratory building
- primary pump house
- office and laboratory building - housing radiochemistry laboratories
- fan house - housing ventilation systems and waste ion exchangers and filters
- hot retention area - holding waste tanks
- waste handling building

B.3.2.1 Research Reactor

The test case file names for this group are preceded by RR and the contaminants consist of ^{60}Co, ^{90}Sr, and ^{137}Cs with initial activities of 1 pCi/gram. The baseline test case for the research reactor is the same as the one used for the power reactor in Section B.3.1.1 and will not be included here. The results shown are for the residential scenario only and have the Combine Progeny switch turned on.

B.3.2.1.1 RR1 Contamination Under a House

This test case is comparable to the test case in Section B.3.1.2 except the area of contamination is smaller (.25 * 186 m^2 = 46.5 m^2) and the time spent indoors is therefore reduced to 52.5 (210 * .25) days per year. The change in the time indoors assumes that a person spends 200 days spread over the 2000 ft^2 house but that only 25% of the time is on contaminated ground because of the smaller contaminated area.

Selected Radionuclides	Initial Activity
^{60}Co	1 pCi/gram
^{90}Sr	1 pCi/gram
^{137}Cs	1 pCi/gram

This test case is set up by selecting the three radionuclides specified above and changing the following parameters:

Time Indoors	52.5000	days/year
Time Outdoors	0.0000	days/year
Time Gardening	0.0000	days/year
Water Ingestion	0.0000	L/d
Cultivated Area	46.5000	m^2
Garden Diet	0.0000	
Diet - Fish	0.0000	kg/y

Results:

The peak dose of 1.27E+000 TEDE (mrem) occurred 1.00 year after license termination.

Table B.7 Pathway Component of Maximum Annual Dose for RR1 Test Case

Pathway	TEDE (mrem)	Percentage
External	1.27E+000	99.95
Inhalation	3.22E-006	0.00
Agricult	0.00E+000	0.00
Soil	5.93E-004	0.05
Drinking	0.00E+000	0.00
Irrigated	0.00E+000	0.00
Aquatic	0.00E+000	0.00
Total	1.27E+000	100.00

Table B.8 Radionuclide Component of Maximum Annual Dose for RR1 Test Case

Radionuclide	TEDE (mrem)	Percentage
^{60}Co	1.02E+000	80.76
^{90}Sr	4.27E-004	0.03
^{90}Y	1.78E-003	0.14
^{137}Cs	2.41E-001	19.07
Total	1.27E+000	100.00

B.3.2.1.2 RR2 Contamination In the Garden

This test case assumes that the contamination is only in the garden and is comparable to the test case in B.3.1.3 except the area of contamination is smaller (46.5 m^2).

Selected Radionuclides	Initial Activity
^{60}Co	1 pCi/gram
^{90}Sr	1 pCi/gram
^{137}Cs	1 pCi/gram

Time Indoors	0.0000	days/year
Time Outdoors	1.2780	days/year
Cultivated Area	46.5000	m^2

Garden Diet 4.0000E-003

Results:

The peak dose of 1.11E+000 TEDE (mrem) occurred 5.00 years after license termination.

Table B.9 Pathway Component of Maximum Annual Dose for RR2 Test Case

Pathway	TEDE (mrem)	Percentage
External	9.16E-002	8.25
Inhalation	2.90E-005	0.00
Agricult	8.89E-002	8.01
Soil	1.55E-005	0.00
Drinking	1.57E-001	14.12
Irrigated	3.47E-003	0.31
Aquatic	7.69E-001	69.31
Total	1.11E+000	100.00

Table B.10 Radionuclide Component of Maximum Annual Dose for RR2 Test Case

Radionuclide	TEDE (mrem)	Percentage
^{60}Co	9.05E-002	8.15
^{90}Sr	1.59E-001	14.33
^{90}Y	9.05E-003	0.82
^{137}Cs	8.51E-001	76.70
Total	1.11E+000	100.00

B.3.2.2 Test Reactor

The test case file names for this group are preceded by TR and the contaminants consist of ^{60}Co, ^{90}Sr, and ^{137}Cs with initial activities of 1 pCi/gram. The baseline test case for the test reactor is the same as the one used for the power reactor in Section B.3.1.1 and will not be included here. The results shown are for the residential scenario only and have the Combine Progeny switch turned on.

B.3.2.2.1 TR1 Contamination Under a House

This test case is comparable to the test case in Section B.3.1.2 except the area of contamination is larger (465 m^2 or 5000 ft^2). This is larger than the default house but smaller than the default area, which assumes that there is contamination under the entire house with 3000 ft^2 (279m^2) left over to contaminate the garden. The default parameters are changed as shown below:

Selected Radionuclides	Initial Activity
^{60}Co	1 pCi/gram
^{90}Sr	1 pCi/gram
^{137}Cs	1 pCi/gram

Time Outdoors	7.6681	days/year
Cultivated Area	465.0000	m^2
Garden Diet	0.0242	

Results:

The peak dose of 1.14E+001 TEDE (mrem) occurred 4.00 years after license termination.

Table B.11 Pathway Component of Maximum Annual Dose for the TR1 Test Case

Pathway	TEDE (mrem)	Percentage
External	3.55E+000	31.01
Inhalation	4.45E-005	0.00
Agricult	6.96E-001	6.08
Soil	1.21E-003	0.01
Drinking	5.89E+001	5.15
Irrigated	8.06E-002	0.70
Aquatic	6.52E+000	57.04
Total	1.14E+001	100.00

Table B.12 Radionuclide Component of Maximum Annual Dose for the TR1 Test Case

Radionuclide	TEDE (mrem)	Percentage
^{60}Co	3.37E+000	29.48
^{90}Sr	9.78E-001	8.55
^{90}Y	5.44E-002	0.48
^{137}Cs	7.03E+000	61.49
Total	1.14E+001	100.00

B.3.2.2.2 TR2 Garden Contamination

This test case assumes that the contamination is only in the garden and is comparable to the test case in Section B.3.1.3 except that the area of contamination is larger (465 m^2). The default parameters are changed as shown below:

Appendix C Data Tables

C.1 Introduction

This appendix contains data tables needed in the calculation of annual total effective dose equivalents (TEDEs) for the residual radioactive contamination scenarios defined in [Kennedy and Strenge, 1992] with differences as described in Appendix A. The tables provided in this appendix cannot be changed by the user of the DandD software. The numbers of significant figures shown in each table were obtained from primary references and are included for completeness.

C.2 Contents and Units of Tables

Table C.1 contains radioactive decay data and decay-chain specifications for the listing of radionuclides. The radioactive decay database contains information taken directly from ICRP Publication 38 [ICRP 1983] and modifications described in Appendix A. The database contains a data set for each radionuclide or chain, except for natural thorium (Th_Nat) or natural uranium (U_Nat), for which dose factors are calculated from the radionuclides in the decay chain (see Section 2.6.1). The entries are organized by increasing atomic number and by decay chain. Within each chain, members follow according to their decay sequence. Progeny radionuclides are treated as implicit or explicit, as defined in Section 6.1.1 of [Kennedy and Strenge, 1992]. The columns in Table C.1 include listings for 1) parent radionuclide, 2) progeny radionuclides in the chain, 3) radioactive half-life (in days), 4) position of each radionuclide in the chain, 5) branching information for the first parent of each radionuclide (index and fraction), 6) branching information for the second parent, if any, of each radionuclide (index and fraction), and 7) atomic number. The files provided with DandD that contain this table are named CHAIN.ASC and RADTAB.RAD.

Tables C.2 through C.5 contain the standardized databases of external and internal effective dose equivalent factors, committed effective dose conversion factors, and the organ dose equivalent factors used in this generic screening analysis. These factors are obtained from recent Federal Guidance published by the EPA, implementing the recommendations of the ICRP [Eckerman, Wolbarst, and Richardson 1988; Eckerman and Ryman 1992]. Entries are included for each parent radionuclide with implicit progeny immediately following the parent entry. Implicit progency are signified by an entry showing in the second column of each table.

Table C.2 lists the internal committed effective dose equivalent (CEDE) and the external effective dose equivalent factors. The columns in Table C.2 include listings for 1) radionuclide, 2) implicit progeny fraction, 3) internal committed effective dose equivalent for ingestion (Sv/Bq), 4) internal CEDE for inhalation (Sv/Bq), and 5) external effective dose equivalents for surface sources (Sv/d per Bq/m^2) and volume sources with 15-cm thickness (Sv/d per Bq/m^3). The implicit progeny fraction is the parameter F_j used in Equation (6.7) in [Kennedy and Strenge, 1992] to calculate combined dose factors for parent radionuclides with implicit (short-lived) progeny. The files provided with DandD that contain this table are named CEDE.ASC and CEDETAB.CED.

Tables C.3 through C.5 contain ingestion organ dose equivalent factors for use in the drinking water scenario. Each table contains columns showing the radionuclide, implicit progeny fraction, and organ dose equivalent factors in units of Sv per Bq ingested. Table C.3 includes factors for adrenals, urinary bladder wall (Bld Wall), bone surface (B Surface), brain, breast, stomach wall (S Wall), and small intestine wall (SI Wall). Table C.4 includes factors for upper large intestine wall (ULI Wall), lower large intestine wall (LLI Wall), kidneys, liver, lungs, ovaries, and pancreas. Table C.5 includes factors for red bone marrow (R Marrow), skin, spleen, testes, thymus, thyroid, and uterus. The files provided with DandD that contain these tables are named ORGAN1.ASC, ORGAN2.ASC, and ORGAN3.ASC for Tables C.3, C.4, and C.5, respectively.

C.3 References

Eckerman, K. F., and J. C. Ryman. 1992. *Dose Coefficients for External Exposure to Radionuclides Distributed in Air, Water, and Soil.* Federal Guidance Report No. 12, U.S. Environmental Protection Agency, Washington, D.C.

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

International Commission on Radiation Protection (ICRP). 1983. *Radionuclide Transformations - Energy*

and Intensity of Emissions. ICRP Publication 38, vol. 11-13, Pergamon Press, New York.

Kennedy, Jr., W.E., and D.L. Strenge, 1992. *Residual Radioactive Contamination From Decommissioning: Technical Basis for Translating Contamination Levels to Annual Effective Dose Equivalent*, Vol. 1, NUREG/CR-5512, PNL-7994, U.S. Nuclear Regulatory Commission, Washington, DC.

Table C.1 Decay Chain Data File

			Branching Information				
			First Parent		Second Parent		
Radio-nuclide	Half-life (d)	Position in Chain	Index	Fraction	Index	Fraction	Atomic Number
3H	4.51E+03	1					1
10Be	5.84E+08	1					4
14C	2.09E+06	1					6
18F	7.62E-02	1					9
22Na	9.50E+02	1					11
24Na	6.25E-01	1					11
32P	1.43E+01	1					15
33P	2.54E+01	1					15
35S	8.74E+01	1					16
36Cl	1.10E+08	1					17
40K	4.68E+11	1					19
41Ca	5.11E+07	1					20
45Ca	1.63E+02	1					20
46Sc	8.38E+01	1					21
51Cr	2.77E+01	1					24
54Mn	3.13E+02	1					25
56Mn	1.07E-01	1					25
55Fe	9.86E+02	1					26
59Fe	4.45E+01	1					26
57Co	2.71E+02	1					27
58Co	7.08E+01	1					27
60Co	1.93E+03	1					27
59Ni	2.74E+07	1					28
63Ni	3.51E+04	1					28
65Ni	1.05E-01	1					28
64Cu	5.29E-01	1					29
65Zn	2.44E+02	1					30
69mZn	5.73E-01	1					30
69Zn			1	0.9997			
69Zn	3.96E-02	1					30
76As	1.10E+00	1					33
75Se	1.20E+02	1					34
79Se	2.37E+07	1					34
82Br	1.47E+00	1					35
83Br	9.96E-02	1					35

Table C.1 Decay Chain Data File

Radio-nuclide	Half-life (d)	Position in Chain	Branching Information: First Parent Index	First Parent Fraction	Second Parent Index	Second Parent Fraction	Atomic Number
83mKr		1	0.9998				
84Br	2.21E-02	1					35

Table C.1 Decay Chain Data File (Continued)

Radio-nuclide	Half-life (d)	Position in Chain	Branching Information: First Parent Index	First Parent Fraction	Second Parent Index	Second Parent Fraction	Atomic Number
86Rb	1.87E+01	1					37
88Rb	1.24E-02	1					37
89Rb	1.06E-02	1					37
89Sr	5.05E+01	2	1	1.0000	0	0.0000	38
85Sr	6.48E+01	1					38
89Sr	5.05E+01	1					38
90Sr	1.06E+04	1					38
90Y	2.67E+00	2	1	1.0000	0	0.0000	39
91Sr	3.96E-01	1					38
91mY			1	0.5780			
91Y	5.85E+01	2	1	1.0000	0	0.0000	39
90Y	2.67E+00	1					39
91mY	3.45E-02	1					39
91Y	5.85E+01	2	1	1.0000	0	0.0000	39
91Y	5.85E+01	1					39
92Y	1.48E-01	1					39
93Y	4.21E-01	1					39
93Zr	5.59E+08	2	1	1.0000	0	0.0000	40
93mNb	4.97E+03	3	2	1.0000	0	0.0000	41
93Zr	5.59E+08	1					40
93mNb	4.97E+03	2	1	1.0000	0	0.0000	41
93Zr+C	(Same data as for named radionuclide)						
95Zr	6.40E+01	1					40
95mNb	3.61E+00	2	1	0.0070	0	0.0000	41
95Nb	3.52E+01	3	2	1.0000	1	0.9930	41
97Zr	7.04E-01	1					40
97mNb			1	0.9470			
97Nb			1	0.0530			
93mNb	4.97E+03	1					41
94Nb	7.41E+06	1					41
95mNb	3.61E+00	1					41
95Nb	3.52E+01	2	1	1.0000	0	0.0000	41
95Nb	3.52E+01	1					41
97mNb	6.94E-04	1					41

Table C.1 Decay Chain Data File (Continued)

Radio-nuclide	Half-life (d)	Position in Chain	Branching Information: First Parent Index	First Parent Fraction	Second Parent Index	Second Parent Fraction	Atomic Number
97Nb	5.01E-02	2	1	1.0000	0	0.0000	41
97Nb	5.01E-02	1					41
93Mo	1.28E+06	1					42
93mNb	4.97E+03	2	1	1.0000	0	0.0000	41
99Mo	2.75E+00	1					42
99mTc			1	0.8760			
99Tc	7.78E+07	2	1	1.0000	0	0.0000	43
99mTc	2.51E-01	1					43
99Tc	7.78E+07	2	1	1.0000	0	0.0000	43
99Tc	7.78E+07	1					43
101Tc	9.86E-03	1					43
103Ru	3.93E+01	1					44
103mRh			1	0.9970	0	0.0000	45
105Ru	1.85E-01	1					44
105Rh	1.47E+00	2	1	1.0000	0	0.0000	45
106Ru	3.68E+02	1					44
106Ru			1	1.0000			
103mRh	3.90E-02	1					45
105Rh	1.47E+00	1					45
103Pd	1.70E+01	1					46
103mRh			1	0.9997			
107Pd	2.37E+09	1					46
109Pd	5.59E-01	1					46
110mAg	2.50E+02	1					47
110Ag			1	0.0133			
111Ag	7.45E+00	1					47
109Cd	4.64E+02	1					48
113mCd	4.97E+03	1					48
115mCd	4.46E+01	1					48
115In	1.86E+18	2	1	1.0000			49
115Cd	2.23E+00	1					48
115mIn			1	1.0000			
115In	1.86E+18	2	1	0.9500			49
111In	2.83E+00	1					49
114mIn	4.95E+01	1					49
114In			1	0.9570			
115mIn	1.87E-01	1					49
115In	1.86E+18	2	1	0.9500			49
117mSn	1.36E+01	1					50
119mSn	2.93E+02						50
121mSn	2.01E+04	1					50

Table C.1 Decay Chain Data File (Continued)

Radio-nuclide	Half-life (d)	Position in Chain	Branching Information: First Parent Index	First Parent Fraction	Second Parent Index	Second Parent Fraction	Atomic Number
121Sn	1.13E+00	2	1	0.7760	0	0.0000	50
121Sn	1.13E+00	1					50
123Sn	1/29E+02	1					50
125Sn	9.64E+00	1					50
125Sb	1.01E+03	2	1	1.0000	0	0.0000	51
125mTe	5.80E+01	3	2	0.2280	0	0.0000	52
126Sn	3.65E+07	1					50
126mSb			1	1.0000			
126Sb	1.24E+01	2	1	0.1400	0	0.0000	51
126Sn+C	(Same data as for named radionuclide)						
124Sb	6.02E+01	1					51
125Sb	1.01E+03	1					51
125mTe	5.80E+01	2	1	0.2280	0	0.0000	52
126mSb	1.32E+02	1					51
126Sb	1.24E+01	2	1	0.1400	0	0.0000	51
126Sb	1.24E+01	1					51
127Sb	3.85E+00	1					51
127mTe	1.09E+02	2	1	0.1760	0	0.0000	52
127Te	3.90E-01	3	2	0.9760	1	0.8240	52
123mTe	1.20E+02	1					52
123Te	3.65E+15	2	1	1.0000			52
125mTe	5.80E+01	1					52
127mTe	1.09E+02	1					52
127Te	3.90E-01	2	1	0.9760	0	0.0000	52
127Te	3.90E-01	1					52
129mTe	3.36E+01	1					52
129Te			1	0.6500			
129I	5.73E+09	2	1	1.0000	0	0.0000	53
129Te	4.83E-02	1					52
129I	5.7eE+09	2	1	1.0000	0	0.0000	53
131mTe	1.25E+00	1					52
131Te			1	0.2220			
131I	8.04E+00	2	1	1.0000	0	0.0000	53
131mXe	1.19E+01	3	2	0.0111			54
131Te	1.74E-02	1					52
131I	8.04E+00	2	1	1.0000	0	0.0000	53
131mXe	1.19E+01	3	2	0.0111			54
132Te	3.26E+00	1					52
132I			1	1.0000			
133mTe	3.85E-02	1					52
133Te			1	0.1300			

Appendix D Input Parameter Defaults

D.1 Introduction

This appendix lists all of the default parameters used by the DandD software for each scenario. The parameters are listed in the same order that they are presented in the software. Each parameter is shown with its name, default value, range of allowable values, units, and a description of the parameter with its variable name.

Table D.1 Building Occupancy Scenario Parameters

Parameter Name	Default Value	Range (Low:High)	Units	Description and Variable Name
Time in Building	97.46	0.00:250.00	days/year	The time in the building (converted to effective 24-hour days) during the occupancy period (To).
Occupancy Period	365.25	1.00:365.25	days	The duration of the occupancy exposure period (TTo).
Breathing Rate	1.40	0.00:3.63	m^3/hour	The average volumetric breathing rate during building occupancy for an 8-hour work day (Vo).
Resuspension Factor	1.42E-5	0.00:4.00E-2	m^{-1}	The resuspension factor during the occupancy period (RFo).
Ingestion Rate	1.11E-5	0.00:3.00E-2	m^2/hour	The secondary ingestion transfer rate of loose removable surface activity from building surfaces to the mouth during building occupancy (GO).
Start Time	0.00	0.00:365.25E+06	days	The start time of the scenario.
End Time	365.25	1.00:365.25E+06	days	The ending time of the scenario.
Time Step Size	365.25	1.00:365.25	days	The time step size.
Print Step Size	1.00	Constant	None	The time steps for the history file. Doses will be written to the history file every n time steps.

Table D.2 Building Renovation Scenario Parameters

Parameter Name	Default Value	Range (Low:High)	Units	Description and Variable Name
Time on Job	62.83	0.12:130.39	days/year	The time in the building (converted to effective 24-hour days) during the renovation period (Tb).
Renovation Period	179.00	1.00:365.25	days	The duration of the renovation exposure period (Ttb).
Breathing Rate	1.27	0.00:3.00	m^3/hour	The average volumetric breathing rate for an 8-hour renovation work day (Vb).
Dust-loading	9.10E-3	0.00:1.50E-2	gram/m^3	The average dust-loading in the air during renovation work activities (CDB).
Transfer Rate	5.20E-2	0.00:0.10	gram/hour	The effective transfer rate for ingestion of loose dust to the hands and mouth during building renovation (GB).
Start Time	0.00	0.00:365.25E+06	days	The start time of the scenario.
End Time	365.25	0.00:365.25E+06	days	The ending time of the scenario.
Time Step Size	91.3125	1.00:365.25	days	The time step size.
Print Step Size	1.00	1.00:365.25E+06	None	The time steps for the history file. Doses will be written to the history file every n time steps.

Table D.3 Drinking Water Scenario Parameters

Parameter Name	Default Value	Range (Low:High)	Units	Description and Variable Name
Surface Soil Density	1.625	1.00:2.00	g\cm^3	Bulk density of the surface-layer soil (Rho1)
Unsaturated Density	1.625	1.00:2.00	g/cm^3	Bulk density of the unsaturated zone soil (Rho2).
Exposure Duration	365.25	Constant	days	Total time in the drinking water scenario exposure period (ttr).
Ingestion Period	365.25	0.00:365.25	days/year	The drinking water ingestion period during the exposure period (td).
Ingestion Rate	2.02	1.00:2.50	L/day	The daily drinking water ingestion rate (Uw).
Layer Thickness	0.15	0.01:1.00	m	The thickness of the surface-layer soil (H1).
Unsaturated Zone	1.00	1.00:200.00	m	The thickness of the unsaturated zone soil (H2).
Zone Layers	1	1:10	None	The number of layers in the unsaturated zone (See Appendix A).
Layer Porosity	0.35	0.01:0.60	None	The porosity of the surface-layer soil (n1).
Unsaturated Porosity	0.35	0.01:0.60	None	The porosity of the unsaturated zone soil (n2).
Layer Ratio	0.17	0.14:0.80	None	Degree of saturation for the surface-layer soil (f1).
Unsaturated Ratio	0.20	0.14:1.00	None	Degree of saturation for the unsaturated zone soil (f2).
Domestic Use	110000.00	63000:290000	L	The volume of water removed from the ground-water aquifer annually by an individual for all domestic purposes (Vdr).
Infiltration Rate	0.17	0.00:0.46	m/year	Infiltration rate of recharge through the contaminated area (I).
Site Area	610.00	1.00:1.0E+07	m^2	Area of contaminated site (Ar).
Start Time	0.00	0.00:365.25E+06	days	The start time of the scenario.
End Time	365250.00	1.00:365.25E+06	days	The ending time of the scenario.
Time Step Size	365.25	1.00:365.25	days	The time step size.
Print Step Size	1.00	1.00:365.25E+06	None	The time steps for the history file. Doses will be written to the history file every n time steps.

Table D.4 Residential Scenario Basic Parameters

Parameter Name	Default Value	Range (Low:High)	Units	Description and Variable Name
Time Indoors	240.00	0.00:300.00	days/year	Time an individual spends indoors (in equivalent 24-hour days) during the exposure period (ti).

Time Outdoors	40.20	0.00:365.25	days/year	Time an individual spends outdoors (in equivalent 24-hour days) during the exposure period (tx).

Table D.4 Residential Scenario Basic Parameters (Continued)

Parameter Name	Default Value	Range (Low:High)	Units	Description and Variable Name
Time Gardening	2.92	0.00:130.00	days/year	Time an individual spends gardening (in equivalent 24-hour days) during the exposure period (tg).
Exposure Period	365.25	Constant	days	Total time in the residential scenario exposure period (ttr).
Garden Period	90.00	0.01:365.25	days	Total time in gardening period (ttg).
Weathering Rate	0.0495	0.007:0.70	days	Weathering rate for activity removal or radionuclides from plants (lambda(w)).
Indoor Shielding	0.552	0.00:1.00	None	Indoor shielding factor for external dose (SFI).
Outdoor Shielding	1.00	0.00:1.00	None	Outdoor shielding factor for external dose (SFO).
Floor Dust	0.15986	0.00:10.00	g/m^2	Indoor air dust-loading factor for floors (Pd).
Resuspension Factor	2.82E-6	0.00:1.00E-4	m^{-1}	Resuspension factor for indoor dust (RFr).
Indoor Dust	1.41E-4	0.00:5.00E-03	g/m^3	Air dust-loading factor for indoor exposure period (CDI).
Outdoor Dust	3.14E-6	0.00:2.00E-3	g/m^3	Air dust-loading factor for outdoor exposure period (CDO).
Gardening Dust	4.00E-4	0.00:1.00E-2	g/m^3	Air dust-loading factor for gardening exposure period (CDG).
Indoor Breathing	0.90	0.00:3.70	m^3/h	Volumetric breathing rate for time spent indoors (Vr).
Outdoor Breathing	1.40	0.00:3.70	m^3/h	Volumetric breathing rate for time spent outdoors (Vx).
Gardening Breathing	2.70	0.00:3.70	m^3/h	Volumetric breathing rate for time spent gardening (Vg).
Soil Ingestion	5.00E-2	0.00:1.00E-1	g/day	The effective soil ingestion transfer rate for ingestion of soil and dust transferred to the mouth (GR).
Water Ingestion	1.31	0.00:5.10	L/day	The daily drinking water ingestion rate (Uw).
Water Period	365.25	None	days	The drinking water consumption period (td)
Layer Thickness	0.15	Constant	m	The thickness of the surface layer soil. This is set equal to the assumed plow layer depth (H1).
Unsaturated Zone	1.2287	0.30:320.00	m	The thickness of the unsaturated zone soil (H2).
Zone Layers	1	1:10	None	The number of layers in the unsaturated zone (See Appendix A).
Layer Porosity	0.4599	0.01:0.60	None	The porosity of the surface-layer soil (n1).

Table D.4 Residential Scenario Basic Parameters (Continued)

Parameter Name	Default Value	Range (Low:High)	Units	Description and Variable Name
Unsaturated Porosity	0.4599	0.01:0.60	None	The porosity of the unsaturated zone soil (n2).
Surface Layer Ratio	0.1625	0.05:0.80	None	Degree of saturation for the surface-layer soil (f1).
Unsaturated Ratio	0.1625	0.05:1.00	None	Degree of saturation for the unsaturated zone soil (f2).
Surface Soil Density	1.4312	1.00:2.00	g/cm^3	Bulk density of the surface-layer soil (rho1).
Unsaturated Density	1.4312	1.00:2.00	g/cm^3	Bulk density of the unsaturated zone soil (rho2).
Domestic Use	118000.00	0.00:290000.00	L	The volume of water removed annually from the ground-water aquifer for all domestic purposes by an individual (Vdr).
Pond Volume	1300000	0:1300000	L	The volume of water in the surface-water pond used to grow fish for the aquatic food ingestion pathway (Vsw).
Infiltration Rate	0.2525	0.00:0.60	m/year	Infiltration rate of recharge through the cultivated area (I).
Cultivated Area	2400.00	1.00:100000.00	m^2	Area of cultivated land (Ar).
Irrigation Rate	1.29	0.00:9.30	L/m^2*days	Irrigation water application rate to the cultivated area (IR).
Start Time	0.00	0.00:365.25E+06	days	The start time of the scenario.
End Time	365250.00	1.00:365.25E+06	days	The end time of the scenario.
Time Step Size	365.25	1.00:365.25	days	The time step size.
Print Step Size	1.00	Constant	None	The time steps for the history file. Doses will be written to the history file every n time steps.

Table D.5 Human Diet Parameters from Table 6.23 of [Kennedy and Strenge, 1992]

Parameter Name	Default Value	Range (Low:High)	Units	Description and Variable Name
*Garden Diet	1.00	0.00:1.00	None	Fraction of human diet grown onsite (DIET).
Diet - Leafy	21.40	None	kg/year	Yearly human consumption of leafy vegetables (Uv(1)).
Diet - Roots	44.60	None	kg/year	Yearly human consumption of other vegetables (Uv(2)).
Diet - Fruit	52.80	None	kg/year	Yearly human consumption of fruits (Uv(3)).
Diet - Grain	14.40	None	kg/year	Yearly human consumption of grain (Uv(4)).
Diet - Beef	39.80	None	kg/year	Yearly human consumption of beef (Ua(1)).
Diet - Poultry	25.30	None	kg/year	Yearly human consumption of poultry (Ua(2)).
Diet - Milk	233.00	None	kg/year	Yearly human consumption of milk (Ua(3)).
Diet - Egg	19.10	None	kg/year	Yearly human consumption of eggs (Ua(4)).

Table D.5 Human Diet Parameters from Table 6.23 of [Kennedy and Strenge, 1992]

Parameter Name	Default Value	Range (Low:High)	Units	Description and Variable Name
*Diet - Fish	20.60	>=0.00	kg/year	Yearly human consumption of fish produced from an onsite pond (Uf).
**Consumption Period	365.25	None	days	Consumption period for fish (tf).

* Modified by parameter analysis

** Described in Section 6.6 of [Kennedy and Strenge, 1992].

Table D.6 Food Consumption Periods from Table 6.23 of [Kennedy and Strenge, 1992]

Parameter Name	Default Value	Range (Low:High)	Units	Description and Variable Name
Food - Leafy	365.25	None	days	Food consumption period for leafy vegetables (tcv(1)).
Food - Roots	365.25	None	days	Food consumption period for other vegetables (tcv(2)).
Food - Fruit	365.25	None	days	Food consumption period for fruits (tcv(3)).
Food - Grain	365.25	None	days	Food consumption period for grains (tcv(4)).
Food - Beef	365.25	None	days	Food consumption period for beef (tca(1)).
Food - Poultry	365.25	None	days	Food consumption period for poultry (tca(2)).
Food - Milk	365.25	None	days	Food consumption period for milk (tca(3)).
Food - Egg	365.25	None	days	Food consumption period for eggs (tca(4)).

Table D.7 Holdup Periods from Table 6.23 of [Kennedy and Strenge, 1992]

Parameter Name	Default Value	Range (Low:High)	Units	Description and Variable Name
Period - Leafy	1	None	days	Holdup period for leafy vegetables (thv(1)).
Period - Root	14	None	days	Holdup period for other vegetables (thv(2)).
Period - Fruit	14	None	days	Holdup period for fruit (thv(3)).
Period - Grain	14	None	days	Holdup period for grain (thv(4)).
Period - Beef	20	None	days	Holdup period for beef (tha(1)).
Period - Poultry	1	None	days	Holdup period for poultry (tha(2)).
Period - Milk	1	None	days	Holdup period for milk (tha(3)).
Period - Eggs	1	None	days	Holdup period for eggs (tha(4)).

Table D.8 Growing Periods from Table 6.23 of [Kennedy and Strenge, 1992]

Parameter Name	Default Value	Range (Low:High)	Units	Description and Variable Name
Leafy	45	None	days	Minimum growing period for leafy vegetables (tgv(1)).
Root	90	None	days	Minimum growing period for other vegetables (tgv(2)).
Fruit	90	None	days	Minimum growing period for fruit (tgv(3)).
Grain	90	None	days	Minimum growing period for grain (tgv(4)).
Beef Forage	30	None	days	Minimum growing period for forage consumed by poultry (tgf(1)).
Poultry Forage	30	None	days	Minimum growing period for forage consumed by beef cattle (tgf(2)).

Milk Cow Forage	30	None	days	Minimum growing period for forage consumed by milk cows (tgf(3)).
Layer Hen Forage	30	None	days	Minimum growing period for forage consumed by layer hens (tgf(4)).
Beef Grain	90	None	days	Minimum growing period for stored grain consumed by beef cattle (tgg(1)).
Poultry Grain	90	None	days	Minimum growing period for stored grain consumed by poultry (tgg(2)).
Milk Cow Grain	90	None	days	Minimum growing period for stored grain consumed by milk cows (tgg(3)).

Table D.8 Growing Periods from Table 6.23 of [Kennedy and Strenge, 1992] (Continued)

Parameter Name	Default Value	Range (Low:High)	Units	Description and Variable Name
Layer Hen Grain	90	None	days	Minimum growing period for stored grain consumed by layer hens (tgg(4)).
Beef Hay	45	None	days	Minimum growing period for stored hay consumed by beef cattle (tgh(1)).
Poultry Hay	45	None	days	Minimum growing period for stored hay consumed by poultry (tgh(2)).
Milk Cow Hay	45	None	days	Minimum growing period for stored hay consumed by milk cows (tgh(3)).
Layer Hen Hay	45	None	days	Minimum growing period for stored hay consumed by layer hens (tgh(4)).

Table D.9 Animal Feeding Period from Section 6.5.4 of [Kennedy and Strenge, 1992]

Parameter Name	Default Value	Range (Low:High)	Units	Description and Variable Name
Beef	365.25	0.01:365.25	days	Time period for feeding forage to beef cattle (tff(1)).
Poultry	365.25	0.01:365.25	days	Time period for feeding forage to poultry (tff(2)).
Milk Cow	365.25	0.01:365.25	days	Time period for feeding forage to milk cows (tff(3)).
Layer Hen	365.25	0.01:365.25	days	Time period for feeding forage to layer hens (tff(4)).
Beef Grain	365.25	0.00:365.25	days	Time period for feeding grain to beef cattle (tfg(1)).
Poultry Grain	365.25	0.00:365.25	days	Time period for feeding grain to poultry (tfg(2)).
Milk Cow Grain	365.25	0.00:365.25	days	Time period for feeding grain to milk cows (tfg(3)).
Layer Hen Grain	365.25	0.00:365.25	days	Time period for feeding grain to layer hens (tfg(4)).
Beef Hay	365.25	0.00:365.25	days	Time period for feeding stored hay to beef cattle (tfh(1)).
Poultry Hay	365.25	0.00:365.25	days	Time period for feeding stored hay to poultry (tfh(2)).
Milk Cow Hay	365.25	0.00:365.25	days	Time period for feeding stored hay to milk cows (tfh(3)).
Layer Hen Hay	365.25	0.00:365.25	days	Time period for feeding stored hay to layer hens (tfh(4)).

Table D.10 Interception Fractions from Table 6.23 of [Kennedy and Strenge, 1992]

Parameter Name	Default Value	Range (Low:High)	Units	Description and Variable Name

Leafy	0.350	0.00:1.00	days	Interception fraction for radionuclides in water deposited on leafy vegetables (rv(1)).
Root	0.350	0.00:1.00	days	Interception fraction for radionuclides in water deposited on other vegetables (rv(2)).
Fruit	0.350	0.00:1.00	days	Interception fraction for radionuclides in water deposited on fruit (rv(3)).
Grain	0.350	0.00:1.00	days	Interception fraction for radionuclides in water deposited on grain (rv(4)).
Beef Forage	0.349	0.00:1.00	days	Interception fraction for radionuclides in water deposited on forage consumed by beef cattle (rf(1)).

Table D.10 Interception Fractions from Table 6.23 of [Kennedy and Strenge, 1992] (Continued)

Parameter Name	Default Value	Range (Low:High)	Units	Description and Variable Name
Poultry Forage	0.349	0.00:1.00	days	Interception fraction for radionuclides in water deposited on forage consumed by poultry (rf(2)).
Milk Cow Forage	0.349	0.00:1.00	days	Interception fraction for radionuclides in water deposited on forage consumed by milk cows (rf(3)).
Layer Hen Forage	0.349	0.00:1.00	days	Interception fraction for radionuclides in water deposited on forage consumed by layer hens (rf(4)).
Beef Grain	0.350	0.00:1.00	days	Interception fraction for radionuclides in water deposited on stored grain consumed by beef cattle (rg(1)).
Poultry Grain	0.350	0.00:1.00	days	Interception fraction for radionuclides in water deposited on stored grain consumed by poultry (rg(2)).
Milk Cow Grain	0.350	0.00:1.00	days	Interception fraction for radionuclides in water deposited on stored grain consumed by milk cows (rg(3)).
Layer Hen Grain	0.350	0.00:1.00	days	Interception fraction for radionuclides in water deposited on stored grain consumed by layer hens (rg(4)).
Beef Hay	0.349	0.00:1.00	days	Interception fraction for radionuclides in water deposited on stored hay consumed by beef cattle (rh(1)).
Poultry Hay	0.349	0.00:1.00	days	Interception fraction for radionuclides in water deposited on stored hay consumed by poultry (rh(2)).
Milk Cow Hay	0.349	0.00:1.00	days	Interception fraction for radionuclides in water deposited on stored hay consumed by milk cows (rh(3)).
Layer Hen Hay	0.349	0.00:1.00	days	Interception fraction for radionuclides in water deposited on stored hay consumed by layer hens (rh(4)).

Table D.11 Translocation Factors from Table 6.23 of [Kennedy and Strenge, 1992]

Parameter Name	Default Value	Range (Low:High)	Units	Description and Variable Name
Leafy	1.00	0.00:1.00	None	Translocation factor for leafy vegetables (Tv(1)).
Root	0.10	0.00:1.00	None	Translocation factor for other vegetables (Tv(2)).
Fruit	0.10	0.00:1.00	None	Translocation factor for fruit (Tv(3)).
Grain	0.10	0.00:1.00	None	Translocation factor for grain (Tv(4)).
Beef Forage	1.00	0.00:1.00	None	Translocation factor for forage consumed by beef cattle (Tf(1)).
Poultry Forage	1.00	0.00:1.00	None	Translocation factor for forage consumed by poultry (Tf(2)).

Milk Cow Forage	1.00	0.00:1.00	None	Translocation factor for forage consumed by milk cows (Tf(3)).
Layer Hen Forage	1.00	0.00:1.00	None	Translocation factor for forage consumed by layer hens (Tf(4)).
Beef Grain	0.10	0.00:1.00	None	Translocation factor for stored grain consumed by beef cattle (Tg(1)).
Poultry Grain	0.10	0.00:1.00	None	Translocation factor for stored grain consumed by poultry (Tg(2)).
Milk Cow Grain	0.10	0.00:1.00	None	Translocation factor for stored grain consumed by milk cows (Tg(3)).
Layer Hen Grain	0.10	0.00:1.00	None	Translocation factor for stored grain consumed by layer hens (Tg(4)).

Table D.11 Translocation Factors from Table 6.23 of [Kennedy and Strenge, 1992] (Continued)

Parameter Name	Default Value	Range (Low:High)	Units	Description and Variable Name
Beef Hay	1.00	0.00:1.00	None	Translocation factor for stored hay consumed by beef cattle (Th(1)).
Poultry Hay	1.00	0.00:1.00	None	Translocation factor for stored hay consumed by poultry (Th(2)).
Milk Cow Hay	1.00	0.00:1.00	None	Translocation factor for stored hay consumed by milk cows (Th(3)).
Layer Hen Hay	1.00	0.00:1.00	None	Translocation factor for stored hay consumed by layer hens (Th(4)).

Table D.12 Fraction of Contaminated Foods from Table 6.23 of [Kennedy and Strenge, 1992]

Parameter Name	Default Value	Range (Low:High)	Units	Description and Variable Name
Beef Forage	1.00	0.00:1.00	None	Fraction of forage consumed by beef cattle that is contaminated (xf(1)).
Poultry Forage	1.00	0.00:1.00	None	Fraction of forage consumed by poultry that is contaminated (xf(2)).
Milk Cow Forage	1.00	0.00:1.00	None	Fraction of forage consumed by milk cows that is contaminated (xf(3)).
Layer Hen Forage	1.00	0.00:1.00	None	Fraction of forage consumed by layer hens that is contaminated (xf(4)).
Beef Grain	1.00	0.00:1.00	None	Fraction of stored grain consumed by beef cattle that is contaminated (xg(1)).
Poultry Grain	1.00	0.00:1.00	None	Fraction of stored grain consumed by poultry that is contaminated (xg(2)).
Milk Cow Grain	1.00	0.00:1.00	None	Fraction of stored grain consumed by milk cows that is contaminated (xg(3)).
Layer Hen Grain	1.00	0.00:1.00	None	Fraction of stored grain consumed by layer hens that is contaminated (xg(4)).
Beef Hay	1.00	0.00:1.00	None	Fraction of stored hay consumed by beef cattle that is contaminated (xh(1)).
Poultry Hay	1.00	0.00:1.00	None	Fraction of stored hay consumed by poultry that is contaminated (xh(2)).
Milk Cow Hay	1.00	0.00:1.00	None	Fraction of stored hay consumed by milk cows that is contaminated (xh(3)).

Table D.12 Fraction of Contaminated Foods from Table 6.23 of [Kennedy and Strenge, 1992]

Parameter Name	Default Value	Range (Low:High)	Units	Description and Variable Name
Layer Hen Hay	1.00	0.00:1.00	None	Fraction of stored hay consumed by layer hens that is contaminated (xh(4)).
Beef Water	1.00	0.00:1.00	None	Fraction of water consumed by beef cattle that is contaminated (xw(1)).
Poultry Water	1.00	0.00:1.00	None	Fraction of water consumed by poultry that is contaminated (xw(2)).
Milk Cow Water	1.00	0.00:1.00	None	Fraction of water consumed by milk cows that is contaminated (xw(3)).
Layer Hen Water	1.00	0.00:1.00	None	Fraction of water consumed by layer hens that is contaminated (xw(4)).

Table D.13 Crop Yields from Table 6.23 of [Kennedy and Strenge, 1992]

Parameter Name	Default Value	Range (Low:High)	Units	Description and Variable Name
Leafy	1.88921	>=0.01	kg/m^3	Crop yields for leafy vegetables (Yv(1)).
Root	2.40002	>=0.01	kg/m^3	Crop yields for other vegetables (Yv(2)).
Fruit	2.36732	>=0.01	kg/m^3	Crop yields for fruit (Yv(3)).
Grain	0.390429	>=0.01	kg/m^3	Crop yields for grain (Yv(4)).
Beef Forage	1.8868	>=0.01	kg/m^3	Crop yields for forage consumed by beef cattle (Yf(1)).
Poultry Forage	1.8868	>=0.01	kg/m^3	Crop yields for forage consumed by poultry (Yf(2)).
Milk Cow Forage	1.8868	>=0.01	kg/m^3	Crop yields for forage consumed by milk cows (Yf(3)).
Layer Hen Forage	1.8868	>=0.01	kg/m^3	Crop yields for forage consumed by layer hens (Yf(4)).
Beef Grain	0.656769	>=0.01	kg/m^3	Crop yields for stored grain consumed by beef cattle (Yg(1)).
Poultry Grain	0.656769	>=0.01	kg/m^3	Crop yields for stored grain consumed by poultry (Yg(2)).
Milk Cow Grain	0.656769	>=0.01	kg/m^3	Crop yields for stored grain consumed by milk cows (Yg(3)).
Layer Hen Grain	0.656769	>=0.01	kg/m^3	Crop yields for stored grain consumed by layer hens (Yg(4)).
Beef Hay	1.8868	>=0.01	kg/m^3	Crop yields for stored hay consumed by beef cattle (Yh(1)).
Poultry Hay	1.8868	>=0.01	kg/m^3	Crop yields for stored hay consumed by poultry (Yh(2)).
Milk Cow Hay	1.8868	>=0.01	kg/m^3	Crop yields for stored hay consumed by milk cows (Yh(3)).
Layer Hen Hay	1.8868	>=0.01	kg/m^3	Crop yields for stored hay consumed by layer hens (Yh(4)).

Table D.14 Wet-to-Dry Weight Conversion Factors from Table 6.23 of [Kennedy and Strenge, 1992]

Parameter Name	Default Value	Range (Low:High)	Units	Description and Variable Name

Leafy	0.133577	None	None	Wet-to-dry weight conversion factors for leafy vegetables (Wv(1)).
Root	0.162031	None	None	Wet-to-dry weight conversion factors for other vegetables (Wv(2)).
Fruit	0.284903	None	None	Wet-to-dry weight conversion factors for fruit (Wv(3)).
Grain	0.88	None	None	Wet-to-dry weight conversion factors for grain (Wv(4)).
Beef Forage	0.251767	None	None	Wet-to-dry weight conversion factors for forage consumed by beef cattle (Wf(1)).
Poultry Forage	0.251767	None	None	Wet-to-dry weight conversion factors for forage consumed by poultry (Wf(2)).
Milk Cow Forage	0.251767	None	None	Wet-to-dry weight conversion factors for forage consumed by milk cows (Wf(3)).
Layer Hen Forage	0.251767	None	None	Wet-to-dry weight conversion factors for forage consumed by layer hens (Wf(4)).
Beef Grain	0.88	None	None	Wet-to-dry weight conversion factors for stored grain consumed by beef cattle (Wg(1)).

Table D.14 Wet-to-Dry Weight Conversion Factors from Table 6.23 of [Kennedy and Strenge, 1992] (Continued)

Parameter Name	Default Value	Range (Low:High)	Units	Description and Variable Name
Poultry Grain	0.88	None	None	Wet-to-dry weight conversion factors for stored grain consumed by poultry (Wg(2)).
Milk Cow Grain	0.88	None	None	Wet-to-dry weight conversion factors for stored grain consumed by milk cows (Wg(3)).
Layer Hen Grain	0.88	None	None	Wet-to-dry weight conversion factors for stored grain consumed by layer hens (Wg(4)).
Beef Hay	0.251767	None	None	Wet-to-dry weight conversion factors for stored hay consumed by beef cattle (Wh(1)).
Poultry Hay	0.251767	None	None	Wet-to-dry weight conversion factors for stored hay consumed by poultry (Wh(2)).
Milk Cow Hay	0.251767	None	None	Wet-to-dry weight conversion factors for stored hay consumed by milk cows (Wh(3)).
Layer Hen Hay	0.251767	None	None	Wet-to-dry weight conversion factors for stored hay consumed by layer hens (Wh(4)).

Table D.15 Animal Intake Rates from Table 6.23 of [Kennedy and Strenge, 1992]

Parameter Name	Default Value	Range (Low:High)	Units	Description and Variable Name
Beef Forage	8.133	None	kg/day	Animal feed intake rates for forage consumed by beef cattle (Qf(1)).
Poultry Forage	5.6E-2	None	kg/day	Animal feed intake rates for forage consumed by poultry (Qf(2)).
Milk Cow Forage	35.1654	None	kg/day	Animal feed intake rates for forage consumed by milk cows (Qf(3)).
Layer Hen Forage	7.55E-2	None	kg/day	Animal feed intake rates for forage consumed by layer hens (Qf(4)).
Beef Grain	2.41877	None	kg/day	Animal feed intake rates for stored grain consumed by beef cattle (Qg(1)).

Poultry Grain	6.30E-2	None	kg/day	Animal feed intake rates for stored grain consumed by poultry (Qg(2)).
Milk Cow Grain	1.94662	None	kg/day	Animal feed intake rates for stored grain consumed by milk cows (Qg(3)).
Layer Hen Grain	6.10E-2	None	kg/day	Animal feed intake rates for stored grain consumed by layer hens (Qg(4)).
Beef Hay	16.2535	None	kg/day	Animal feed intake rates for stored hay consumed by beef cattle (Qh(1)).
Poultry Hay	0.00	None	kg/day	Animal feed intake rates for stored hay consumed by poultry (Qh(2)).
Milk Cow Hay	26.1089	None	kg/day	Animal feed intake rates for stored hay consumed by milk cows (Qh(3)).
Layer Hen Hay	0.00	None	kg/day	Animal feed intake rates for stored hay consumed by layer hens (Qh(4)).
Beef Water	50.00	None	L/day	Water intake rates for beef cattle (Qw(1)).
Poultry Water	0.30	None	L/day	Water intake rates for poultry (Qw(2)).

Table D.15 Animal Intake Rates from Table 6.23 of [Kennedy and Strenge, 1992] (Continued)

Parameter Name	Default Value	Range (Low:High)	Units	Description and Variable Name
Milk Cow Water	60.00	None	L/day	Water intake rates for milk cows (Qw(3)).
Layer Hen Water	0.30	None	L/day	Water intake rates for layer hens (Qw(4)).
Beef Period	365.25	0.00:365.25	days	Water intake consumption period for beef cattle (tfw(1)).
Poultry Period	365.25	0.00:365.25	days	Water intake consumption period for poultry (tfw(2)).
Milk Cow Period	365.25	0.00:365.25	days	Water intake consumption period for milk cows (tfw(3)).
Layer Hen Period	365.25	0.00:365.25	days	Water intake consumption period for layer hens (tfw(4)).
Beef Soil	0.02	None	None	Mass fraction of soil in dry fresh forage consumed by beef cattle (Qd(1)).
Poultry Soil	0.10	None	None	Mass fraction of soil in dry fresh forage consumed by poultry (Qd(2)).
Milk Cow Soil	0.02	None	None	Mass fraction of soil in dry fresh forage consumed by milk cows (Qd(3)).
Layer Hen Soil	0.10	None	None	Mass fraction of soil in dry fresh forage consumed by layer hens (Qd(4)).

Table D.16 Plant Mass Loading Factors from Section 6.5.2 of [Kennedy and Strenge, 1992]

Parameter Name	Default Value	Range (Low:High)	Units	Description and Variable Name
Leafy	0.10	0.0011:0.26	g/g	Plant mass loading factor for leafy vegetables (mlv(1)).
Root	0.10	0.0011:0.26	g/g	Plant mass loading factor for other vegetables (mlv(2)).
Fruit	0.10	0.0011:0.26	g/g	Plant mass loading factor for fruit (mlv(3)).
Grain	0.10	0.0011:0.26	g/g	Plant mass loading factor for grain (mlv(4)).

Beef Forage	0.10	0.0011:0.26	g/g	Plant mass loading factor for forage consumed by beef cattle (mlf(1)).
Poultry Forage	0.10	0.0011:0.26	g/g	Plant mass loading factor for forage consumed by poultry (mlf(2)).
Milk Cow Forage	0.10	0.0011:0.26	g/g	Plant mass loading factor for forage consumed by milk cows (mlf(3)).
Layer Hen Forage	0.10	0.0011:0.26	g/g	Plant mass loading factor for forage consumed by layer hens (mlf(4)).
Beef Grain	0.10	0.0011:0.26	g/g	Plant mass loading factor for stored grain consumed by beef cattle (mlg(1)).
Poultry Grain	0.10	0.0011:0.26	g/g	Plant mass loading factor for stored grain consumed by poultry (mlg(2)).
Milk Cow Grain	0.10	0.0011:0.26	g/g	Plant mass loading factor for stored grain consumed by milk cows (mlg(3)).
Layer Hen Grain	0.10	0.0011:0.26	g/g	Plant mass loading factor for stored grain consumed by layer hens (mlg(4)).
Beef Hay	0.10	0.0011:0.26	g/g	Plant mass loading factor for stored hay consumed by beef cattle (mlh(1)).

Table D.16 Plant Mass Loading Factors from Section 6.5.2 of [Kennedy and Strenge, 1992] (Continued)

Parameter Name	Default Value	Range (Low:High)	Units	Description and Variable Name
Poultry Hay	0.10	0.0011:0.26	g/g	Plant mass loading factor for stored hay consumed by poultry (mlh(2)).
Milk Cow Hay	0.10	0.0011:0.26	g/g	Plant mass loading factor for stored hay consumed by milk cows (mlh(3)).
Layer Hen Hay	0.10	0.0011:0.26	g/g	Plant mass loading factor for stored hay consumed by layer hens (mlh(4)).

Table D.17 Carbon Model Parameters from Table 6.24 of [Kennedy and Strenge, 1992]

Parameter Name	Default Value	Range (Low:High)	Units	Description and Variable Name
Beef	0.36	None	None	Mass fraction of beef cattle that is carbon (fca(1)).
Poultry	0.18	None	None	Mass fraction of poultry that is carbon (fca(2)).
Milk	6.00E-2	None	None	Mass fraction of milk cows that is carbon (fca(3)).
Eggs	0.16	None	None	Mass fraction of layer hens that is carbon (fca(4)).
Beef Forage	0.11	None	None	Mass fraction of wet forage consumed by beef cattle that is carbon (fcf(1)).
Poultry Forage	0.11	None	None	Mass fraction of wet forage consumed by poultry that is carbon (fcf(2)).
Milk Cow Forage	0.11	None	None	Mass fraction of wet forage consumed by milk cows that is carbon (fcf(3)).
Layer Hen Forage	0.11	None	None	Mass fraction of wet forage consumed by layer hens that is carbon (fcf(4)).
Beef Grain	0.40	None	None	Mass fraction of wet stored grain consumed by beef cattle that is carbon (fcg(1)).
Poultry Grain	0.40	None	None	Mass fraction of wet stored grain consumed by poultry that is carbon (fcg(2)).

Table D.17 Carbon Model Parameters from Table 6.24 of [Kennedy and Strenge, 1992]

Parameter Name	Default Value	Range (Low:High)	Units	Description and Variable Name
Milk Cow Grain	0.40	None	None	Mass fraction of wet stored grain consumed by milk cows that is carbon (fcg(3)).
Layer Hen Grain	0.40	None	None	Mass fraction of wet stored grain consumed by layer hens that is carbon (fcg(4)).
Beef Hay	7.00E-2	None	None	Mass fraction of wet stored hay consumed by beef cattle that is carbon (fch(1)).
Poultry Hay	7.00E-2	None	None	Mass fraction of wet stored hay consumed by poultry that is carbon (fch(2)).
Milk Cow Hay	7.00E-2	None	None	Mass fraction of wet stored hay consumed by milk cows that is carbon (fch(3)).
Layer Hen Hay	7.00E-2	None	None	Mass fraction of wet stored hay consumed by layer hens that is carbon (fch(4)).
Soil	0.03	0.00:0.10	None	Mass fraction of dry soil that is carbon (fcs).
Animal Activity	1.00	None	None	Specific activity equivalence of animal product and specific activity of animal feed, forage, and soil (SATac).

Table D.18 Tritium Model Parameters from Table 6.23 of [Kennedy and Strenge, 1992]

Parameter Name	Default Value	Range (Low:High)	Units	Description and Variable Name
Leafy	0.10	None	None	Mass fraction of wet leafy vegetables that is hydrogen (fHv(1)).
Root	0.10	None	None	Mass fraction of wet other vegetables that is hydrogen (fHv(2)).
Fruit	0.10	None	None	Mass fraction of wet fruit that is hydrogen (fHv(3)).
Grain	6.80E-2	None	None	Mass fraction of wet grain that is hydrogen (fHv(4)).
Beef	0.10	None	None	Mass fraction of beef cattle that is hydrogen (fHa(1)).
Poultry	0.10	None	None	Mass fraction of poultry that is hydrogen (fHa(2)).
Milk	0.11	None	None	Mass fraction of milk that is hydrogen (fHa(3)).
Eggs	0.11	None	None	Mass fraction of an egg that is hydrogen (fHa(4)).
Beef Forage	0.10	None	None	Mass fraction of wet forage consumed by beef cattle that is hydrogen (fHf(1)).
Poultry Forage	0.10	None	None	Mass fraction of wet forage consumed by poultry that is hydrogen (fHf(2)).
Milk Cow Forage	0.10	None	None	Mass fraction of wet forage consumed by milk cows that is hydrogen (fHf(3)).
Layer Hen Forage	0.10	None	None	Mass fraction of wet forage consumed by layer hens that is hydrogen (fHf(4)).
Beef Grain	6.80E-2	None	None	Mass fraction of wet stored grain consumed by beef cattle that is hydrogen (fHg(1)).
Poultry Grain	6.80E-2	None	None	Mass fraction of wet stored grain consumed by poultry that is hydrogen (fHg(2)).
Milk Cow Grain	6.80E-2	None	None	Mass fraction of wet stored grain consumed by milk cows that is hydrogen (fHg(3)).
Layer Hen Grain	6.80E-2	None	None	Mass fraction of wet stored grain consumed by layer hens that is hydrogen (fHg(4)).

Table D.18 Tritium Model Parameters from Table 6.23 of [Kennedy and Strenge, 1992]

Parameter Name	Default Value	Range (Low:High)	Units	Description and Variable Name
Beef Hay	0.10	None	None	Mass fraction of wet stored hay consumed by beef cattle that is hydrogen (fHh(1)).
Poultry Hay	0.10	None	None	Mass fraction of wet stored hay consumed by poultry that is hydrogen (fHh(2)).
Milk Cow Hay	0.10	None	None	Mass fraction of wet stored hay consumed by milk cows that is hydrogen (fHh(3)).
Layer Hen Hay	0.10	None	None	Mass fraction of wet stored hay consumed by layer hens that is hydrogen (fHh(4)).
Soil	5.80E-3	0.00:0.06	None	Mass fraction of moist soil that is hydrogen expressed in terms of mass of hydrogen per mass of dry soil (fHs).
Animal Product	1.00	None	None	Specific activity equivalence of animal product and specific activity of the total tritium and hydrogen intake (SATaH).
Plant/Soil	1.00	None	None	Specific activity equivalence of soil and plants for tritium (SAsvH).
Plant/Water	1.00	None	None	Specific activity equivalence of plants and tritium in water (SAwvH).
Soil Moisture	5.22E-2	0.01:0.60	L/kg	Moisture content of the soil (SH).

Table D.19 Soil-to-Plant Concentration Factors from Table 6.16 of [Kennedy and Strenge, 1992]

Element	Atomic Number	Leafy Vegetables	Root Vegetables	Fruit	Grain
H	1	0.0E+0	0.0E+0	0.0E+0	0.0E+0
Be	4	1.0E-2	1.5E-3	1.5E-3	1.5E-3
C	6	3.2E-1	7.0E-1	7.0E-1	2.2E-1
N	7	3.0E+1	3.0E+1	3.0E+1	3.0E+1
F	9	6.0E-2	6.0E-3	6.0E-3	6.0E-3
Na	11	7.4E-2	2.8E-2	1.6E-2	5.2E-2
Mg	12	1.0E+0	5.5E-1	5.5E-1	5.5E-1
Si	14	3.5E-1	7.0E-2	7.0E-2	7.0E-2
P	15	3.5E+0	3.5E+0	3.5E+0	3.5E+0
S	16	2.3E+0	1.5E+0	1.5E+0	1.5E+0
Cl	17	1.6E+2	7.0E+1	7.0E+1	1.0E+3
Ar	18	0.0E+0	0.0E+0	0.0E+0	0.0E+0
K	19	8.4E+0	5.5E-1	5.5E-1	1.3E+0
Ca	20	1.4E+1	3.5E-1	3.5E-1	1.6E+0
Sc	21	6.0E-3	1.0E-3	1.0E-3	1.0E-3
Cr	24	2.2E-2	8.0E-2	4.6E-2	1.5E-2
Mn	25	3.3E-1	1.1E+1	4.2E+0	1.4E-1
Fe	26	5.6E-3	2.6E-3	1.5E-3	4.8E-4
Co	27	4.0E-2	3.9E+0	2.2E-2	1.1E-2
Ni	28	3.4E-2	2.5E+0	3.4E-1	3.8E-2
Cu	29	4.9E-1	2.6E-1	1.5E-1	4.9E-2
Zn	30	3.1E-1	2.4E-1	1.1E+0	5.5E+0
Ga	31	4.0E-3	4.0E-4	4.0E-4	4.0E-4
As	33	4.0E-2	6.0E-3	6.0E-3	6.0E-3
Se	34	4.9E-2	2.5E-2	2.5E-2	1.6E-1
Br	35	1.5E+0	1.5E+0	1.5E+0	1.5E+0
Kr	36	0.0E+0	0.0E+0	0.0E+0	0.0E+0

Rb	37	8.1E-1	7.0E-2	7.0E-2	7.0E-2
Sr	38	6.4E+1	4.6E-1	2.6E-1	8.5E-2
Y	39	1.5E-2	6.0E-3	6.0E-3	6.0E-3
Zr	40	7.2E-2	4.7E-3	2.7E-3	8.7E-4
Nb	41	4.6E-2	5.0E-3	5.0E-3	4.3E-3
Mo	42	5.2E+1	6.0E-2	6.0E-2	6.0E-2
Tc	43	3.6E+1	1.5E+0	1.5E+0	7.3E-1
Ru	44	1.8E-2	8.6E-3	3.0E-1	1.6E-3
Rh	45	1.5E-1	4.0E-2	4.0E-2	4.0E-2
Pd	46	1.2E+0	4.0E-2	4.0E-2	1.8E-1
Ag	47	5.5E+0	1.0E-1	1.0E-1	1.0E-1
Cd	48	5.0E+0	1.5E-1	6.7E-1	2.2E-1
In	49	4.0E-3	4.0E-4	4.0E-4	4.0E-4
Sn	50	4.3E-2	6.0E-3	6.0E-3	6.0E-3
Sb	51	9.0E-1	3.0E-2	3.0E-2	3.0E-2
Te	52	1.7E-2	4.0E-3	4.0E-3	2.5E-3
I	53	1.6E-1	2.8E-2	1.6E-2	5.1E-3
Xe	54	0.0E+0	0.0E+0	0.0E+0	0.0E+0
Cs	55	1.8E-2	3.1E-2	1.4E-1	6.6E-3
Ba	56	3.9E-2	8.0E-3	4.6E-3	1.5E-3
La	57	1.0E-2	4.0E-3	4.0E-3	4.0E-3
Ce	58	6.4E-1	4.0E-3	2.0E-3	8.2E-4
Pr	59	1.0E-2	4.0E-3	4.0E-3	4.0E-3
Nd	60	1.0E-2	4.0E-3	4.0E-3	4.0E-3
Pm	61	1.0E-2	4.0E-3	4.0E-3	4.0E-3
Sm	62	1.0E-2	4.0E-3	4.0E-3	4.0E-3

Table D.19 Soil-to-Plant Concentration Factors from Table 6.16 of [Kennedy and Strenge, 1992] (Continued)

Element	Atomic Number	Leafy Vegetables	Root Vegetables	Fruit	Grain
Eu	63	1.0E-2	4.0E-3	4.0E-3	4.0E-3
Gd	64	1.0E-2	4.0E-3	4.0E-3	4.0E-3
Tb	65	1.0E-2	4.0E-3	4.0E-3	4.0E-3
Dy	66	1.0E-2	4.0E-3	4.0E-3	4.0E-3
Ho	67	1.0E-2	4.0E-3	4.0E-3	4.0E-3
Er	68	1.0E-2	4.0E-3	4.0E-3	4.0E-3
Hf	72	3.5E-3	8.5E-4	8.5E-4	8.5E-4
Ta	73	1.0E-2	2.5E-3	2.5E-3	2.5E-3
W	74	3.1E-1	1.0E-2	1.0E-2	4.1E-2
Re	75	7.5E+0	3.5E-1	3.5E-1	9.5E-1
Os	76	9.4E-2	3.5E-3	3.5E-3	3.5E-3
Ir	77	1.5E-1	1.5E-2	1.5E-2	1.0E-2
Au	79	4.0E-1	1.0E-1	1.0E-1	1.0E-1
Hg	80	9.0E-1	2.0E-1	2.0E-1	2.0E-1
Tl	81	4.0E-3	4.0E-4	4.0E-4	4.0E-4
Pb	82	4.5E-2	9.0E-3	9.0E-3	9.0E-3
Bi	83	3.5E-2	5.0E-3	5.0E-3	5.0E-3
Po	84	2.5E-3	4.0E-4	4.0E-4	4.0E-4
Rn	86	0.0E+0	0.0E+0	0.0E+0	0.0E+0
Ra	88	1.5E-2	1.5E-3	1.5E-3	1.5E-3
Ac	89	3.5E-3	3.5E-4	3.5E-4	3.5E-4
Th	90	8.5E-4	8.5E-5	8.5E-5	8.5E-5
Pa	91	2.5E-3	2.5E-4	2.5E-4	2.5E-4
U	92	8.5E-3	4.0E-3	4.0E-3	4.0E-3
Np	93	1.9E+1	1.9E-1	1.3E-1	6.8E-2
Pu	94	4.5E-4	4.5E-5	4.5E-5	4.5E-5

Table D.19 Soil-to-Plant Concentration Factors from Table 6.16 of [Kennedy and Strenge, 1992] (Continued)

Element	Atomic Number	Leafy Vegetables	Root Vegetables	Fruit	Grain
Am	95	5.5E-3	2.5E-4	2.5E-4	2.5E-4
Cm	96	8.5E-4	1.5E-5	1.5E-5	1.5E-5
Cf	98	1.0E-2	1.0E-2	1.0E-2	1.1E-2

Table D.20 Animal Product Transfer Factors from Table 6.18 of [Kennedy and Strenge, 1992]

Element	Atomic Number	Beef (d/kg)	Poultry (d/kg)	Milk (d/L)	Eggs (d/kg)
H	1	0.0E+0	0.0E+0	0.0E+0	0.0E+0
Be	4	1.0E-3	4.0E-1	9.0E-7	2.0E-2
C	6	0.0E+0	0.0E+0	0.0E+0	0.0E+0
N	7	7.5E-2	1.0E-1	2.5E-2	8.0E-1
F	9	1.5E-1	1.0E-2	1.0E-3	2.0E+0
Na	11	5.5E-2	1.0E-2	3.5E-2	2.0E-1
Mg	12	5.0E-3	3.0E-2	4.0E-3	1.6E+0
Si	14	4.0E-5	2.0E-1	2.0E-5	8.0E-1
P	15	5.5E-2	1.9E-1	1.5E-2	1.0E+1
S	16	1.0E-1	9.0E-1	1.5E-2	7.0E+0
Cl	17	8.0E-2	3.0E-2	1.5E-2	2.0E+0
Ar	18	0.0E+0	0.0E+0	0.0E+0	0.0E+0
K	19	2.0E-2	4.0E-1	7.0E-3	7.0E-1
Ca	20	7.0E-4	4.4E-2	1.0E-2	4.4E-1
Sc	21	1.5E-2	4.0E-3	5.0E-6	3.0E-3
Cr	24	5.5E-3	2.0E-1	1.5E-3	8.0E-1
Mn	25	4.0E-4	5.0E-2	3.5E-4	6.5E-2
Fe	26	2.0E-2	1.5E+0	2.5E-4	1.3E+0

Table D.20 Animal Product Transfer Factors from Table 6.18 of [Kennedy and Strenge, 1992] (Continued)

Element	Atomic Number	Beef (d/kg)	Poultry (d/kg)	Milk (d/L)	Eggs (d/kg)
Co	27	2.0E-2	5.0E-1	2.0E-3	1.0E-1
Ni	28	6.0E-3	1.0E-3	1.0E-3	1.0E-1
Cu	29	1.0E-2	5.1E-1	1.5E-3	4.9E-1
Zn	30	1.0E-1	6.5E+0	1.0E-2	2.6E+0
Ga	31	5.0E-4	3.0E-1	5.0E-5	8.0E-1
As	33	2.0E-3	8.3E-1	6.0E-5	8.0E-1
Se	34	1.5E-2	8.5E+0	4.0E-3	9.3E+0
Br	35	2.5E-2	4.0E-3	2.0E-2	1.6E+0
Kr	36	0.0E+0	0.0E+0	0.0E+0	0.0E+0
Rb	37	1.5E-2	2.0E+0	1.0E-2	3.0E+0
Sr	38	3.0E-4	3.5E-2	1.5E-3	3.0E-1
Y	39	3.0E-4	1.0E-2	2.0E-5	2.0E-3
Zr	40	5.5E-3	6.4E-5	3.0E-5	1.9E-4
Nb	41	2.5E-1	3.1E-4	2.0E-2	1.3E-3
Mo	42	6.0E-3	1.9E-1	1.5E-3	7.8E-1
Tc	43	8.5E-3	3.0E-2	1.0E-2	3.0E+0
Ru	44	2.0E-3	7.0E-3	6.0E-7	6.0E-3
Rh	45	2.0E-3	5.0E-1	1.0E-2	1.0E-1
Pd	46	4.0E-3	3.0E-4	1.0E-2	4.0E-3
Ag	47	3.0E-3	5.0E-1	2.0E-2	5.0E-1
Cd	48	5.5E-4	8.4E-1	1.0E-3	1.0E-1
In	49	8.0E-3	3.0E-1	1.0E-4	8.0E-1
Sn	50	8.0E-2	2.0E-1	1.0E-3	8.0E-1
Sb	51	1.0E-3	6.0E-3	1.0E-4	7.0E-2

Table D.20 Animal Product Transfer Factors from Table 6.18 of [Kennedy and Strenge, 1992] (Continued)

Element	Atomic Number	Beef (d/kg)	Poultry (d/kg)	Milk (d/L)	Eggs (d/kg)
Te	52	1.5E-2	8.5E-2	2.0E-4	5.2E+0
I	53	7.0E-3	1.8E-2	1.0E-2	2.8E+0
Xe	54	0.0E+0	0.0E+0	0.0E+0	0.0E+0
Cs	55	2.0E-2	4.4E+0	7.0E-3	4.9E-1
Ba	56	1.5E-4	8.1E-4	3.5E-4	1.5E+0
La	57	3.0E-4	1.0E-1	2.0E-5	9.0E-3
Ce	58	7.5E-4	1.0E-2	2.0E-5	5.0E-3
Pr	59	3.0E-4	3.0E-2	2.0E-5	5.0E-3
Nd	60	3.0E-4	4.0E-3	2.0E-5	2.0E-4
Pm	61	5.0E-3	2.0E-3	2.0E-5	2.0E-2
Sm	62	5.0E-3	4.0E-3	2.0E-5	7.0E-3
Eu	63	5.0E-3	4.0E-3	2.0E-5	7.0E-3
Gd	64	3.5E-3	4.0E-3	2.0E-5	7.0E-3
Tb	65	4.5E-3	4.0E-3	2.0E-5	7.0E-3
Dy	66	5.5E-3	4.0E-3	2.0E-5	7.0E-3
Ho	67	4.5E-3	4.0E-3	2.0E-5	7.0E-3
Er	68	4.0E-3	4.0E-3	2.0E-5	7.0E-3
Hf	72	1.0E-3	6.0E-5	5.0E-6	2.0E-4
Ta	73	6.0E-4	3.0E-4	3.0E-6	1.0E-3
W	74	4.5E-2	2.0E-1	3.0E-4	8.0E-1
Re	75	8.0E-3	4.0E-2	1.5E-3	4.0E-1
Os	76	4.0E-1	1.0E-1	5.0E-3	9.0E-2
Ir	77	1.5E-3	5.0E-1	2.0E-6	1.0E-1
Au	79	8.0E-3	5.0E-1	5.5E-6	5.0E-1
Hg	80	2.5E-1	1.1E-2	4.5E-4	2.0E-1
Tl	81	4.0E-2	3.0E-1	2.0E-3	8.0E-1
Pb	82	3.0E-4	2.0E-1	2.5E-4	8.0E-1
Bi	83	4.0E-4	1.0E-1	5.0E-4	8.0E-1
Po	84	3.0E-4	9.0E-1	3.5E-4	7.0E+0
Rn	86	0.0E+0	0.0E+0	0.0E+0	0.0E+0
Ra	88	2.5E-4	3.0E-2	4.5E-4	2.0E-5
Ac	89	2.5E-5	4.0E-3	2.0E-5	2.0E-3
Th	90	6.0E-6	4.0E-3	5.0E-6	2.0E-3
Pa	91	1.0E-5	4.0E-3	5.0E-6	2.0E-3
U	92	2.0E-4	1.2E+0	6.0E-4	9.9E-1
Np	93	5.5E-5	4.0E-3	5.0E-6	2.0E-3
Pu	94	5.0E-7	1.5E-4	1.0E-7	8.0E-3
Am	95	3.5E-6	2.0E-4	4.0E-7	9.0E-3
Cm	96	3.5E-6	4.0E-3	2.0E-5	2.0E-3
Cf	98	5.0E-3	4.0E-3	7.5E-7	2.0E-3

Table D.21 Fish Bioaccumulation Factors from Table 6.19 of [Kennedy and Strenge, 1992]

Element	Factor	Element	Factor	Element	Factor
H	1.0E+0	Y	2.5E+1	Eu	2.5E+1
Be	2.0E+0	Zr	2.0E+2	Gd	2.5E+1
C	4.6E+3	Nb	2.0E+2	Tb	2.5E+1
N	1.5E+5	Mo	1.0E+1	Ho	2.5E+1
F	1.0E+1	Tc	1.5E+1	W	1.2E+3
Na	1.0E+2	Ru	1.0E+2	Re	1.2E+2
P	7.0E+4	Rh	1.0E+1	Os	1.0E+1
S	7.5E+2	Pd	1.0E+1	Ir	1.0E+1
Cl	5.0E+1	Ag	2.3E+0	Au	3.3E+1

Table D.21 Fish Bioaccumulation Factors from Table 6.19 of [Kennedy and Strenge, 1992]

Element	Factor	Element	Factor	Element	Factor
K	1.0E+3	Cd	2.0E+2	Hg	1.0E+3
Ca	4.0E+1	In	1.0E+5	Pb	1.0E+2
Sc	1.0E+2	Sn	3.0E+3	Bi	1.5E+1
Cr	2.0E+2	Sb	2.0E+2	Po	5.0E+2
Mn	4.0E+2	Te	4.0E+2	Rn	0.0E+0
Fe	2.0E+3	I	5.0E+2	Ra	7.0E+1
Co	3.3E+2	Xe	0.0E+0	Ac	2.5E+1
Ni	1.0E+2	Cs	2.0E+3	Th	1.0E+2
Cu	5.0E+1	Ba	2.0E+2	Pa	1.1E+1
Zn	2.5E+3	La	2.5E+1	U	5.0E+1
As	1.0E+2	Ce	5.0E+2	Np	2.5E+2
Se	1.7E+2	Pr	2.5E+1	Pu	2.5E+2
Br	4.2E+2	Nd	2.5E+1	Am	2.5E+2
Rb	2.0E+3	Pm	2.5E+1	Cm	2.5E+2
Sr	5.0E+1	Sm	2.5E+1	Cf	2.5E+1

Table D.22 Residential and Drinking Water Scenario Partition Coefficients

Element	Partition Coefficient	Element	Partition Coefficient
H	0.00E+0	Sb	6.83E+4
Be	9.29E+2	Te	5.48E+2
C	4.34E+0	I	2.83E-1
F	4.97E+0	Xe	0.00E+0
Na	3.56E-2	Cs	1.05E+1
P	2.57E+1	Ba	4.40E+1
S	9.92E+1	La	4.98E+0
Cl	4.95E+0	Ce	8.48E+1

Table D.22 Residential and Drinking Water Scenario Partition Coefficients (Continued)

Element	Partition Coefficient	Element	Partition Coefficient
K	5.01E+0	Pr	1.57E+2
Ca	1.47E+3	Nd	1.58E+2
Sc	5.06E-1	Pm	5.00E+3
Cr	1.01E+2	Sm	9.30E+2
Mn	8.41E+1	Eu	9.40E+2
Fe	5.35E+2	Gd	1.32E-2
Co	1.51E+3	Tb	5.32E+1
Ni	3.70E+1	Ho	6.69E+0
Cu	1.76E+2	W	1.56E+2
Zn	1.06E+3	Re	4.35E+1
As	1.14E+2	Os	1.57E+2
Se	1.15E+2	Ir	1.58E+2
Br	5.62E+1	Au	1.57E+2
Kr	0.00E+0	Hg	1.57E+2
Rb	2.02E+2	Tl	1.58E+2
Sr	3.14E+1	Pb	2.38E+3
Y	7.89E+2	Bi	4.43E+2
Zr	4.66E+4	Po	2.64E+1
Nb	8.26E-1	Rn	0.00E+0
Mo	2.61E+1	Ra	3.53E+3
Tc	7.37E+0	Ac	1.73E+3
Ru	1.58E+3	Th	1.19E+2
Rh	1.27E+2	Pa	4.80E+0
Pd	1.85E+2	U	2.18E+0

Table D.22 Residential and Drinking Water Scenario Partition Coefficients (Continued)

Element	Partition Coefficient	Element	Partition Coefficient
Ag	1.91E+2	Np	1.36E+1
Cd	3.36E+1	Pu	1.36E+1
In	1.58E+2	Am	1.43E+3
Sn	2.52E+1	Cm	1.09E+5
		Cf	1.58E+2

D.2 References

Kennedy, Jr., W.E., and D.L. Strenge, 1992. *Residual Radioactive Contamination From Decommissioning: Technical Basis for Translating Contamination Levels to Annual Effective Dose Equivalent*, Vol. 1, NUREG/CR-5512, PNL-7994, U.S. Nuclear Regulatory Commission, Washington, DC.

Appendix E Text File Formats

E.1 Introduction

This appendix describes the formats of the text files created or used by the DandD software. Section E.2 provides the file format required for importing soil and/or groundwater radioactivity concentrations that are computed using other modeling codes. Section E.3 shows the ***session*** file format that is created when the **Save as Text** or **Write as Text** function is invoked. Section E.4 shows the text file produced by the modeling codes that is called **FORTRAN output** in the Text Reports Window (see Figure 2.33). This report is manipulated by the graphical user interface to produce user-defined reports. Sections E.3 and E.4 used the same session input file to produce the examples.

Section E.5 shows the DandD initialization file and Section E.6 describes the format of the dose history file used to produce graphics.

E.2 Import Concentration Input File

Both the drinking water and residential scenarios include the ability to import concentrations calculated by other modeling codes to compute doses. To use this capability the two scenarios require an import concentration data file for inputting soil and/or groundwater radioactivity concentrations. This section describes the format for the concentration input files, which are similar for both scenarios. The error messages described for the drinking water scenario are identical for the residential scenario. For more information about using this feature, see Section 2.6.3.

The first record in the file tells how many chains will be in the input file and is followed by two more groups of records. The second group gives the number and description of the radionuclides for each chain. The third group contains the time step values and the concentration data associated with each time step.

Table E.1 Drinking Water Scenario File Format

Record	Description
Group 1:	Number of Chains
Record:	Number of radionuclide chains that will be imported.
Format:	Free-format integer: Valid values are 1 - 22.
Description:	The number of chains in the scenario. A description of each individual chain must be provided in the Radionuclide Description group of records following this record (one group per chain).
Comment:	The number of chains must be less than or equal to 22. If not, the results may be undefined. For the purpose of importing concentrations, the natural uranium (U-Nat) chain must be broken into its components and, therefore, counts as three chains.
Error Messages:	• Premature end of file reached reading number of chain records in import concentration file: The file has no data in it. • Error reading number of chain records in import concentration file: Input/Output system error. An error number will be shown that allows support personnel to determine the cause of the problem. If an error code of 6101 is encountered, an invalid integer (such as text) was entered. • Number of chains in import concentration file exceeds maximum allowable (22): This number must be less than or equal to 22.

Table E.1 Drinking Water Scenario File Format (Continued)

Record	Description
Group 2:	Radionuclide Descriptions
Record 1:	Number of explicit radionuclides in this chain.
Format:	Free-form integer; valid values are 1 - 20.
Description:	The number of explicit radionuclides in this chain including the parent radionuclide. This number determines the number of explicit Radionuclide Name records immediately following this record.
Comment:	The number of explicit radionuclides must be less than or equal to 20. Do not include any implicit radionuclides associated with this chain.
Error Messages:	• Premature end of file reached reading number of explicit radionuclides in chain: The file is truncated too soon. • Error reading number of explicit radionuclides in chain: Input/Output system error. An error number will be shown that allows support personnel to determine the cause of the problem. If an error code of 6101 is encountered, an invalid integer (such as letters) was entered. • Number of explicit radionuclides in chain exceeds maximum allowable (20): This number must be less than or equal to 20.
Record 2:	Name of the explicit radionuclide.
Format:	Six characters, alphanumeric, in columns 1 through 6.
Description:	Explicit radionuclide name. There must be one record for each explicit radionuclide in the chain. For example, if there are 7 explicit radionuclides in the chain, there must be 7 records of this type.
Comment:	The order and number of explicit radionuclides in this description must comply exactly with the chain descriptions presented in Appendix C, Table C.1. The code compares this description with the one presented in the table and if there are any discrepancies, the code will abort the run and issue an error message.
Error Messages:	• Premature end of file reached reading names of explicit radionuclides in chain: The file is truncated too soon. • Error reading names of explicit radionuclides in chain: Input/Output system error. An error number will be shown that allows support personnel to determine the cause of the problem. • Radionuclide name is not a valid radionuclide name: The code was unsuccessful in comparing the radionuclide name entered with Table C.1.

Group 2 is repeated until all chain descriptions have been entered.

Group 3:	Concentration Descriptions
Record 1:	Time of Interest.
Format:	Free-format; floating point.

Table E.1 Drinking Water Scenario File Format (Continued)

Record	Description
Description:	The Time of Interest is the time at which the imported concentrations were calculated and at which time the subsequent dose results will be calculated. The input must be in days.
Comment:	The user must make sure that this time corresponds with the times the user is interested in. The code does not make any interpolations or extrapolations of the concentration data to comply with any other time values. For example, if the user is interested in year-by-year doses, the Time of Interest values should be 0.0, 365.25, 730.50, 1095.75, etc.
Error Message:	No concentration data in import file: The file is truncated too soon.
Record 2:	Drinking water concentrations.
Format:	Free-format, floating point, one pair of variables per record, separated by a space.
Description:	The first variable is the groundwater concentration in pCi/L for the radionuclides described in Group 2, above, at the beginning of the Time of Interest. The second variable is the average groundwater concentration in pCi/L of the same radionuclide over a one year time frame beginning with the Time of Interest.
Comment:	This record is repeated for each radionuclide described in Group 2 above. In the example file in Table E.3, there are 3 chains with 7, 6, and 9 radionuclides in each chain, respectively. Therefore, there must be 22 (7 + 6 + 9) records of this type for each time step. The order of presentation of the concentrations must comply with the order presented in Group 2. The code has no way to check if the concentrations are in the correct order. Therefore, the user must use caution in the order and number of these concentration records.
Error Message:	Premature end of file in imported concentrations . . . Too few concentration description records: There must be a concentration record for each radionuclide in each chain for each time step.
Group 3 is repeated as many times as necessary. The code will read Group3 records until the scenario end time given in the Drinking Water Basic Parameters is reached or an end-of-file condition is reached.	

Table E.2 Residential Scenario File Format

Record	Description
Group 1:	Number of Chains - Same as for the drinking water scenario above.
Group 2:	Radionuclide Descriptions - Same as for the drinking water scenario.
Group 3:	Concentration Description.
Record 1:	Time of Interest
Description:	Same as for the drinking water scenario.
Record 2:	Residential soil and groundwater concentrations.
Format:	Free-format, floating point, one pair of variables per record, separated by a space.

Description:	The first variable is the soil concentration in pCi/g for the radionuclides described in Group 2, above, at the beginning of the Time of Interest. The second variable is the average groundwater concentration in pCi/L of the same radionuclide over a one-year time frame beginning with the Time of Interest.

Table E.2 Residential Scenario File Format (Continued)

Record	Description
Comment:	This record is repeated for each radionuclide described in Group 2, above. In the example file in Table E.3, there are three chains with 7, 6, and 9 radionuclides in each chain respectively. Therefore, there must be 22 (7 + 6 + 9) records of this type for each time step. The order of presentation of the concentrations must comply with the order presented in Group 2. The code has no way to check if the concentrations are in the correct order. Therefore, caution must be used in the order and number of these concentration records.

Group 3 is repeated as many times as necessary. The code will read Group 3 records until the scenario end time given in the Residential Basic Parameters is reached or an end-of-file condition is reached. A sample file is shown in Table E.3 which includes four time steps for the natural uranium chain. Do not include the description shown in parentheses in an actual input file.

Table E.3 Example File for Importing Concentrations for the U_Nat Chain (U234, U235, U238)

Value	Value	Comments
3		(Number of Chains)
7		(Number of Radionuclides in chain 1)
234U		(Radionuclide Names for chain 1)
230Th		
226Ra		
222Rn		
210Pb		
210Bi		
210Po		
6		(Number of Radionuclides in chain 2)
235U		(Radionuclide Names for chain 2)
231Th		
231Pa		
227Ac		
227Th		
223Ra		
9		(Number of Radionuclides in chain 3)
238U		(Radionuclide Names for chain 3)
234Th		
234U		
230Th		
226Ra		

222Rn

210Pb

210Bi

210Po

Table E.3 Example File for Importing Concentrations for the U_Nat Chain (U234, U235, U238) (Continued)

Value	Value	Comments
0.000000000000000		(Time of Interest)
0.000000000000000	0.247522757808938D-009	(Concentrations)
0.000000000000000	0.484858183517552D-015	
0.000000000000000	0.396815555468826D-019	
0.000000000000000	0.189522392289388D-017	
0.000000000000000	0.114417255157419D-019	
0.000000000000000	0.103885438401388D-019	
0.000000000000000	0.247829211431455D-020	
0.000000000000000	0.113891077868471D-010	
0.000000000000000	0.112124311179322D-010	
0.000000000000000	0.528562229862447D-016	
0.000000000000000	0.317236495338320D-018	
0.000000000000000	0.218981181517277D-018	
0.000000000000000	0.177312704861995D-018	
0.000000000000000	0.247523276052333D-009	
0.000000000000000	0.179475815505557D-009	
0.000000000000000	0.366644216333686D-015	
0.000000000000000	0.554102623933457D-021	
0.000000000000000	0.330316344522203D-025	
0.000000000000000	0.118831796675130D-023	
0.000000000000000	0.000000000000000	
0.000000000000000	0.000000000000000	
0.000000000000000	0.387340435476567D-024	
365.250000000000		(Time of Interest)
0.685469165302906D-009	0.134741024908478D-008	(Concentrations)
0.175871147027935D-014	0.495362899770729D-014	
0.180111318876507D-018	0.776131599674565D-018	
0.630573744828938D-017	0.174234077170609D-016	
0.477334332848322D-019	0.188000175498888D-018	
0.441835648737467D-019	0.178169650961613D-018	
0.123247502921888D-019	0.664865553040717D-019	
0.315400817560165D-010	0.619976830856126D-010	
0.311885356501322D-010	0.614777022667660D-010	
0.192885440984453D-015	0.548391965969810D-015	

Table E.3 Example File for Importing Concentrations for the U_Nat Chain (U234, U235, U238) (Continued)

Value	Value	Comments
0.144706730419307D-017	0.629031681703798D-017	
0.106100397269115D-017	0.507338766497242D-017	
0.894389208483707D-018	0.456388539794910D-017	
0.685471110732884D-009	0.134741631416726D-008	
0.539059448934161D-009	0.112433606659562D-008	
0.148840725562149D-014	0.511011675415349D-014	
0.276333981176464D-020	0.146035126829713D-019	
0.000000000000000	0.177935345835221D-023	
0.464827012918598D-023	0.253708387955905D-022	
0.000000000000000	0.274573361685459D-024	
0.000000000000000	0.421344011293412D-024	
0.000000000000000	0.172359583696953D-024	
730.500000000000		(Time of Interest)
0.206879336191261D-008	0.285187089665718D-008	(Concentrations)
0.903777804491166D-014	0.145081994022596D-013	
0.169366852946708D-017	0.331261541522632D-017	
0.326728487233615D-016	0.575680706015900D-016	
0.407145837169490D-018	0.835776321078462D-018	
0.388745850894128D-018	0.803567262443758D-018	
0.159893738089962D-018	0.367820596468893D-018	
0.951904187751751D-010	0.131222001499207D-009	
0.945051445996075D-010	0.130377562981958D-009	
0.100528461620271D-014	0.162396085407654D-014	
0.137743262380402D-016	0.270814455674890D-016	
0.114419666828799D-016	0.231249634432062D-016	
0.104923428582542D-016	0.215971076532749D-016	
0.206880510481380D-008	0.285189150527488D-008	
0.177048526051827D-008	0.248112032955631D-008	
0.102389493906991D-013	0.185035180403758D-013	
0.344371423650410D-019	0.749019946800425D-019	
0.496419537842190D-023	0.135319565732186D-022	
0.619363558574050D-022	0.150151780226801D-021	
0.625908757379270D-024	0.175685216656883D-023	
0.573455698477810D-024	0.164872343100813D-023	
0.245519978623145D-024	0.682968925414381D-024	
1095.75000000000		(Time of Interest)
0.364671423177404D-008	0.444454267854206D-008	(Concentrations)
0.204720606482233D-013	0.270903875440447D-01	

Table E.3 Example File for Importing Concentrations for the U_Nat Chain (U234, U235, U238) (Continued)

Value	Value	Comments
0.527880440156510D-017	0.787895755091530D-017	
0.879295903899320D-016	0.130412336018944D-015	
0.139407830043134D-017	0.225891066434437D-017	
0.134514657550422D-017	0.218692384046246D-017	
0.650941479418464D-018	0.111951764593952D-017	
0.167795027852860D-009	0.204505545609364D-009	
0.166794414137780D-009	0.203354588605064D-009	
0.230237188616761D-014	0.306465073296225D-014	
0.433096232478005D-016	0.649656442895449D-016	
0.375113673072341D-016	0.569570227737363D-016	
0.353629674552019D-016	0.541436534350816D-016	
0.364674528108021D-008	0.444458720426173D-008	
0.320488262938123D-008	0.393428619996491D-008	
0.283336250978395D-013	0.411984665622026D-013	
0.127318200099534D-018	0.205208884640300D-018	
0.258386353915024D-022	0.474492466919941D-022	
0.275567255043353D-021	0.501436105729973D-021	
0.356620193728023D-023	0.721972314186154D-023	
0.340581445167568D-023	0.695579766118310D-023	
0.145960036701288D-023	0.319243681886171D-023	

E.3 Session Files

This section describes the text file that is created by the **Save as Text** (see Section 2.6.3) or **Write as Text** (see section 2.4.3.1.4) functions. This type of file can be used for transferring sessions between different machines and to maintain session file compatibility for future releases. There is a sub-directory contained in each session group directory called **ascii** where these files are stored. When a session file is opened and the **Save as Text** function from the Main Session window is run, a file will be created in this directory for that individual session. The file name for the text file consists of the session file name (e.g. sess0004) and the extension **.asc**.

When the **Groups, Session Group, Write as Text** function is run from the main program window, it will create a text file for all sessions in the currently selected session group and put them in the **ascii** directory. This function used in conjunction with the **Read from Text** function, allows the user to move entire session groups between different machines, drives, and directories without maintaining the original directory structure that the group was created under.

These text files can also be used to create session groups for large batch runs. This is useful, for instance, if a user wants to run a lot of sessions using the same parameters but different radionuclides. The procedure to do this is as follows:

2. Create a session file.
3. Save it as a text file.
4. Using this text file as a template, edit it using a text editor such as Windows Notepad.
5. Save the edited files using valid session text file names (e.g. sess0005.asc).
6. Create the new session group.
7. Move all of the edited text into the new session group's **ascii** subdirectory.
8. Run the **Read from Text** function. See the precautionary note in Section 2.4.3.1.5.
9. Execute the **Run Batch** command.

Table E.4 shows the output file created by the DandD code for a session that has had at least one parameter modified for each parameter and table type for the residential scenario. The corresponding text session file that was generated from this session is shown in Table E.5 and is followed by a description of each section in the file.

Table E.4 Session Report for the *Modified Defaults Test Case*

```
Program     : DandD Version 1.0
Session     : Modified Defaults Test Case
Description :
     Modified Defaults Test Case for all parameters
Executed    : 06/29/98 at 08:36:52

                                NRC Report


                          Residential Input Section
                          _________________________

Execution Options
====================
History file will be generated
Implicit progeny doses will not be included with explicit parent.
Concentration data will be calculated.
Initial Radionuclide Activities
========================================
Chain        pCi/gram
========================================

60Co             1.00
90Sr             1.00
137Cs            1.00

Code-Generated Radionuclide Activities
========================================
Chain        pCi/gram
========================================

60Co      1.0000E+000
90Sr      1.0000E+000
90Y       0.0000E+000
137Cs     1.0000E+000
137mBa    0.0000E+000

Basic Parameters
=====================================================================
Name                                  Value      Units      Default
=====================================================================
```

'Time Indoors'	211.0000	days/year	240.0000
'Time Outdoors'	105.0000	days/year	40.2000

Human Diet Parameters

Name	Value	Units	Default
'Garden Diet'	0.3200	None	1.0000

Food Consumption Period Parameters

Name	Value	Units	Default
'Food - Leafy'	0.0000	days	365.2500

Holdup Period Parameters

Name	Value	Units	Default
'Period - Leafy'	3.0000	days	1.0000

Growing Period Parameters

Name	Value	Units	Default
'Leafy'	90.0000	days	45.0000

Animal Feeding Periods Parameters

Name	Value	Units	Default
'Beef'	1.0000E-002	days	365.2500

Interception Fraction Parameters

Name	Value	Units	Default
'Leafy'	0.0000	None	0.3495
'Root'	0.3498	None	0.3498
'Fruit'	0.3497	None	0.3497
'Grain'	0.3499	None	0.3499

Translocation Factor Parameters

Name	Value	Units	Default
'Leafy'	0.5000	None	1.0000

Contaminated Fraction Parameters

Name	Value	Units	Default
'Beef Forage'	0.0000	None	1.0000

Crop Yields Parameters

Name	Value	Units	Default
'Leafy'	1.0000	kg/m^2	2.8892
'Root'	2.4000	kg/m^2	2.4000
'Fruit'	2.3673	kg/m^2	2.3673
'Grain'	0.3904	kg/m^2	0.3904

Wet-To-Dry Conversion Parameters

Name	Value	Units	Default
'Leafy'	0.0000	None	0.1336
'Root'	0.1620	None	0.1620
'Fruit'	0.2849	None	0.2849

Animal Intake Rates Parameters

Name	Value	Units	Default
'Beef Forage'	15.0000	kg/day	8.1330

Plant Mass-Loading Parameters

Name	Value	Units	Default
'Leafy'	0.2000	g/g	0.1000

Residential Output Section

Table E.4 Session Report for the *Modified Defaults Test Case* (Continued)

```
                    Maximum Annual TEDE
                ============================

This scenario started 0.00 year(s) from now
and ran for 1000.00 year(s).

The peak dose of    3.46E+001 TEDE (mrem) occurred 5.00 year(s) after
license termination.

                    Pathway Component of
                    Maximum Annual Dose
                ============================
```

Pathway	TEDE (mrem)	Percentage
External	4.82E+000	13.93
Inhalation	4.17E-005	0.00
Agricult.	3.96E+000	11.45
Soil	1.24E-003	0.00
Drinking	9.37E-001	2.71
Irrigated	1.25E+000	3.62
Aquatic	2.36E+001	68.28
Total	3.46E+001	100.00

```
                Radionuclide Component of
                    Maximum Annual Dose
                ============================
```

Radionuclide	TEDE (mrem)	Percentage
60Co	4.72E+000	13.64
90Sr	4.71E+000	13.62
90Y	1.36E-001	0.39
137Cs	2.48E+001	71.72
137mBa	2.16E-001	0.63
Total	3.46E+001	100.00

Table E.5 Session Text File for the *Modified Defaults Test Case* Session

```
Modified Defaults Test Case                                    Section 1
Modified Defaults Test Case for all parameters
Combine Progeny =  false
Distribute Activity =  false
```

```
'Occupancy'                                                    Section 2
[Misc]
[Rads]
[Params]

'Renovation'                                                   Section 3
[Misc]
[Rads]
[Params]

                                                               Section 4
'Drinking Water'
[Misc]
[Rads]
60Co 1.000000E+000
90Sr 1.000000E+000
137Cs 1.000000E+000
[Params]

[Partition Coefficients]
Co  1.5100E+003
Sr  3.1400E+001
Cs  1.0500E+001
Y   7.8900E+002
Ba  4.4000E+001
```

```
                                                                    Section 5
'Residential'
[Misc]
[Rads]
60Co 1.000000E+000
90Sr 1.000000E+000
137Cs 1.000000E+000
[Params]
'Time Indoors'                    211.0000
'Time Outdoors'                   105.0000
'Garden Diet'                       0.3200
'Food - Leafy'                      0.0000
'Period - Leafy'                    3.0000
'Leafy'                            90.0000
'Beef'                          1.0000E-002
'Leafy'                             0.0000
'Root'                              0.3498
'Fruit'                             0.3497
'Grain'                             0.3499
'Leafy'                             0.5000
'Beef Forage'                       0.0000
'Leafy'                             1.0000
'Root'                              2.4000
'Fruit'                             2.3673
'Grain'                             0.3904
'Leafy'                             0.0000
'Root'                              0.1620
'Fruit'                             0.2849
'Beef Forage'                      15.0000
'Leafy'                             0.2000

[Partition Coefficients]
Co  1.5100E+003
Sr  3.1400E+001
Cs  1.0500E+001
Y   7.8900E+002
Ba  4.4000E+001

[Soil-to-Plant Concentration Factors]
Co  27  4.0000E-002 2.9000E+000 2.2000E-002 1.1000E-002
Sr  38  6.4000E+001 4.6000E-001 2.6000E-001 8.5000E-002
Cs  55  1.8000E-002 3.1000E-002 1.4000E-001 6.6000E-003
Y   39  1.5000E-002 6.0000E-003 6.0000E-003 6.0000E-003
Ba  56  3.9000E-002 8.0000E-003 4.6000E-003 1.5000E-003
[Animal Product Transfer Factors]
Co  27  2.0000E-002 5.0000E-001 2.0000E-003 1.0000E-001
Sr  38  3.0000E-004 3.5000E-002 1.5000E-003 3.0000E-001
Cs  55  2.0000E-002 4.4000E+000 7.0000E-003 4.9000E-001
Y   39  3.0000E-004 1.0000E-002 2.0000E-005 2.0000E-003
Ba  56  1.5000E-004 8.1000E-004 3.5000E-004 1.5000E+000

[Fish Bioaccumulation Factors]
Co  3.3000E+002
Sr  5.0000E+001
Cs  2.0000E+003
Y   2.5000E+001
Ba  2.0000E+002
```

E.3.1 Section 1

This section (Table E.6) contains global information that applies to all sessions. In contains the session title, description, the combine progeny and distribute activity

execution option settings.

Table E.6 Global Session Information

Modified Defaults Test Case
Modified Defaults Test Case for all parameters
Combine Progeny = false
Distribute Activity = false

E.3.2 Section 2

This section (Table E.5) contains the information for the occupancy scenario. Each scenario has a similar section that contains the scenario title, miscellaneous flags, radionuclide information, and the input parameters. The [Misc] section contains miscellaneous flags such as activity units (pCi, mCi, etc.) and scenario selection. In this case, the occupancy scenario was not selected for execution. The [Rads] section contains the radionuclides selected and their activity. Input parameters that have been modified from the default value are contained in the [Params] section.

Table E.7 Occupancy Scenario Section

'Occupancy'
[Misc]
[Rads]
[Params]

E.3.3 Section 3

This section (Table E.8) contains the information for the renovation scenario. The [Misc] section contains miscellaneous flags such as activity units (pCi, mCi, etc.) and scenario selection. In this case, the renovation scenario was not selected for execution. The [Rads] section contains the radionuclides selected and their activity. Input parameters that have been modified from the default value are contained in the [Params] section.

Table E.8 Renovation Scenario Section

'Renovation'
[Misc]
[Rads]
[Params]

E.3.4 Section 4

This section (Table E.9) contains the information for the drinking water scenario. In this case, the drinking water scenario is selected for execution and no basic parameters have been changed. The [Misc] section contains miscellaneous flags such as activity units (pCi, mCi, etc,) and scenario selection. The [Rads] section contains the radionuclides selected and their activity. Input parameters that have been modified from the default values are contained in the [Params] section. The partition coefficients table for the selected radionuclides and their progeny are contained in the [Partition Coefficients] section.

Table E.9 Drinking Water Scenario Section

'Drinking Water'
[Misc]
[Rads]
60Co 1.000000E+000
90Sr 1.000000E+000
137Cs 1.00000E+000
[Params]
[Partition Coefficients]
Co 1.5100E+003
Sr 3.1400E+001
Cs 1.0500E+001
Y 7.8900E+002
Ba 4.4000E+001

E.3.5 Section 5

This section contains the information for the residential scenario. In this case, the residential scenario is selected for execution , two basic parameters have been changed (time indoors and time outdoors), and a parameter from all other parameter types (see Section 2.7) has been changed. The parameters that were changed along with their value and parameter type are shown in Table E.11.

The [Misc] section contains miscellaneous flags such as activity units (pCi, mCi, etc.) and scenario selection. The [Rads] section contains the radionuclides selected and their activity. Input parameters that have ben modified from the default value are contained in the [Params] section. As in the drinking water scenario, all of the tables that are needed follow the basic parameter section with their name in brackets and the data for the selected radionuclides and their progeny.

Table E.10 Residential Scenario Section

```
'Residential'
[Misc]
[Rads]
60Co 1.000000E+000
90Sr 1.000000E+000
137Cs 1.000000E+000
[Params]
'Time Indoors'

   211.0000
'Time Outdoors'

   105.0000
'Garden Diet'

   0.3200
'Food - Leafy'

   0.0000
'Period - Leafy'

   3.0000
'Leafy'


   90.0000
'Beef'


   1.0000E-002
'Leafy'


   0.0000
'Root'


   0.3498
'Fruit'


   0.3497
'Grain'


   0.3499
```

```
'Leafy'

    0.5000
'Beef Forage'

    0.0000
'Leafy'

    1.0000
'Root'

    2.4000
'Fruit'

    2.3673
'Grain'

    0.3904
'Leafy'

    0.0000
'Root'

    0.1620
'Fruit'

    0.2849
'Beef Forage'

    15.0000
'Leafy'

    0.2000

[Partition Coefficients]
Co    1.5100E+003
Sr    3.1400E+001
Cs    1.0500E+001
Y    7.8900E+002
Ba    4.4000E+001
```

Table E.10 Residential Scenario Section (Continued)

```
[Soil-to-Plant Concentration Factors]
Co  27  4.0000E-002 2.9000E+000 2.2000E-002 1.1000E-002
Sr  38  6.4000E+001 4.6000E-001 2.6000E-001 8.5000E-002
Cs  55  1.8000E-002 3.1000E-002 1.4000E-001 6.6000E-003
Y   39  1.5000E-002 6.0000E-003 6.0000E-003 6.0000E-003
Ba  56  3.9000E-002 8.0000E-003 4.6000E-003 1.5000E-003

[Animal Product Transfer Factors]
Co  27  2.0000E-002 5.0000E-001 2.0000E-003 1.0000E-001
Sr  38  3.0000E-004 3.5000E-002 1.5000E-003 3.0000E-001
Cs  55  2.0000E-002 4.4000E+000 7.0000E-003 4.9000E-001
Y   39  3.0000E-004 1.0000E-002 2.0000E-005 2.0000E-003
Ba  56  1.5000E-004 8.1000E-004 3.5000E-004 1.5000E+000

[Fish Bioaccumulation Factors]
Co  3.3000E+002
Sr  5.0000E+001
Cs  2.0000E+003
Y   2.5000E+001
Ba  2.0000E+002
```

Table E.11 Parameter Changes

Parameter	Default Value	New Value	Parameter Type
Time Indoors	240.00	211.00	Basic
Time Outdoors	40.20	105.00	Basic
Garden Diet	1.00	0.3200	Human Diet
Food - Leafy	365.25	0.0000	Food Consumption Period
Period - Leafy	1.0000	3.0000	Holdup Period
Leafy	45	90	Growing Period
Beef	365.25	1.0000E-002	Animal Feeding Period
Leafy	0.349508	0.0000	Interception Fraction
Leafy	1.0000	0.5000	Translocation Factor
Beef Forage	1.0000	0.0000	Contaminated Fraction
Leafy	2.88921	1.0000	Crop Yield
Leafy	0.133577	0.0000	Wet-to-Dry Conversion
Beef Forage	8.133	15.0000	Animal Intake
Leafy	0.1000	0.2000	Plant Mass-Loading

E.4 FORTRAN Output

Table E.12 shows the modeling code output file that was

generated using the session file from Section E.3 for the residential scenario. This is the text file produced by the modeling code that is called **FORTRAN Output** in the Text Reports Window (See Figure 2.30). All input parameters, tables, and initial activities, along with computed results are written to this file during the modeling code execution.

Table E.12 FORTRAN Output for *Modified Defaults Test Case* Session

```
Output From Program 'RESIDE
Residential Scenario
Run Date: 06/29/1998
Run Time: 08:35:06
INPUT DATA:
Title: Modified Defaults Test Case
Notes:
History file will be generated.
Implicit progeny doses will not be included with explicit parent.
Concentration data will be calculated.

PARAMETER DATA:
Time Spent Indoors                                    : 2.1100E+02 d/y
Time Spent Outdoors                                   : 1.0500E+02 d/y
Time Spent Gardening                                  : 2.9200E+00 d/y
Total Time in One Year Exposure Period                : 3.6525E+02 d
Total Time in Gardening Period                        : 9.0000E+01 d
Weathering Rate for Activity Removal from Plants      : 4.9500E-02 /d
Indoor Shielding Factor                               : 5.5200E-01
Outdoor Shielding Factor                              : 1.0000E+00
Floor Dust Loading                                    : 1.5990E-01 g/m^2
Resuspension Factor for Indoor Dust                   : 2.8200E-06 /m
Indoor Air-Dust Loading Factor                        : 1.4100E-06 g/m^3
Outdoor Air-Dust Loading Factor                       : 3.1400E-06 g/m^3
Gardening Air-Dust Loading Factor                     : 4.0000E-04 g/m^3
Indoor Volumetric Breathing Rate                      : 9.0000E-01 m^3/h
Outdoor Volumetric Breathing Rate                     : 1.4000E+00 m^3/h
Gardening Volumetric Breathing Rate                   : 1.7000E+00 m^3/h
Soil Ingestion Transfer Rate                          : 5.0000E-02 g/d
Drinking Water Ingestion Rate                         : 1.3100E+00 L/d
Drinking Water Consumption Period                     : 3.6525E+02 d
Surface Soil Layer Thickness                          : 1.5000E-01 m
Unsaturated Zone Layer Thickness                      : 1.2288E+00 m
Number of Unsaturated Zone Layers                     :  1
Surface Soil Porosity                                 : 4.5990E-01
```

Table E.12 FORTRAN Output for *Modified Defaults Test Case* Session (Continued)

```
Unsaturated Zone Porosity                             : 4.5990E-01
Surface Soil Saturation                               : 1.6260E-01
Unsaturated Zone Saturation                           : 1.6260E-01
```

```
Surface Soil Bulk Soil Density                          : 1.4312E+00 g/cm^3
Unsaturated Zone Bulk Soil Density                      : 1.4312E+00 g/cm^3
Water Volume Removed From Aquifer Per Year              : 1.1800E+05 L
Volume of Surface Water Pond                            : 1.3000E+06 L
Infiltration Rate                                       : 2.5260E-01 m/y
Area of Cultivated Land                                 : 2.4000E+03 m^2
Irrigation Water Application Rate                       : 1.2900E+00 L/m^2-d
Soil Areal Density of Surface Plow Layer                : 2.1468E+02 kg/m^2
Fraction of Diet From Garden                            : 3.2000E-01
Human Diet of Leafy Vegetables                          : 2.1400E+01 kg/y
Human Diet of Other Vegetables                          : 4.4600E+01 kg/y
Human Diet of Fruits                                    : 5.2800E+01 kg/y
Human Diet of Grains                                    : 1.4400E+01 kg/y
Human Diet of Beef                                      : 3.9800E+01 kg/y
Human Diet of Poultry                                   : 2.5300E+01 kg/y
Human Diet of Milk                                      : 2.3300E+02 L/y
Human Diet of Eggs                                      : 1.9100E+01 kg/y
Human Diet of Fish                                      : 2.0600E+01 kg/y
Consumption Period for Fish                             : 3.6525E+02 d
Food Consumption Period for Leafy Vegetables            : 0.0000E+00 d
Food Consumption Period for Other Vegetables            : 3.6525E+02 d
Food Consumption Period for Fruits                      : 3.6525E+02 d
Food Consumption Period for Grains                      : 3.6525E+02 d
Food Consumption Period for Beef                        : 3.6525E+02 d
Food Consumption Period for Poultry                     : 3.6525E+02 d
Food Consumption Period for Milk                        : 3.6525E+02 d
Food Consumption Period for Eggs                        : 3.6525E+02 d
Holdup Period For Leafy Vegetables                      : 3.0000E+00 d
Holdup Period For Other Vegetables                      : 1.4000E+01 d
Holdup Period For Fruits                                : 1.4000E+01 d
Holdup Period For Grains                                : 1.4000E+01 d
Holdup Period For Beef                                  : 2.0000E+01 d
Holdup Period For Poultry                               : 1.0000E+00 d
Holdup Period For Milk                                  : 1.0000E+00 d
Holdup Period For Eggs                                  : 1.0000E+00 d
Minimum Growing Period For Leafy Vegetables             : 9.0000E+01 d
Minimum Growing Period For Other Vegetables             : 9.0000E+01 d
Minimum Growing Period For Fruits                       : 9.0000E+01 d
Minimum Growing Period For Grains                       : 9.0000E+01 d
Minimum Growing Period For Beef Forage                  : 3.0000E+01 d
Minimum Growing Period For Poultry Forage               : 3.0000E+01 d
Minimum Growing Period For Milk Cow Forage              : 3.0000E+01 d
Minimum Growing Period For Layer Hen Forage             : 3.0000E+01 d
Minimum Growing Period For Beef Grain                   : 9.0000E+01 d
Minimum Growing Period For Poultry Grain                : 9.0000E+01 d
Minimum Growing Period For Milk Cow Grain               : 9.0000E+01 d
```

```
Minimum Growing Period For Layer Hen Grain          : 9.0000E+01 d
Minimum Growing Period For Beef Hay                 : 4.5000E+01 d
Minimum Growing Period For Poultry Hay              : 4.5000E+01 d
Minimum Growing Period For Milk Cow Hay             : 4.5000E+01 d
Minimum Growing Period For Layer Hen Hay            : 4.5000E+01 d
Time Period For Feeding Forage to Beef Cattle       : 1.0000E-02 d
Time Period For Feeding Forage to Poultry           : 3.6525E+02 d
Time Period For Feeding Forage to Milk Cattle       : 3.6525E+02 d
Time Period For Feeding Forage to Layer Hens        : 3.6525E+02 d
Time Period For Feeding Grain to Beef Cattle        : 3.6525E+02 d
Time Period For Feeding Grain to Poultry            : 3.6525E+02 d
Time Period For Feeding Grain to Milk Cattle        : 3.6525E+02 d
Time Period For Feeding Grain to Layer Hens         : 3.6525E+02 d
Time Period For Feeding Hay to Beef Cattle          : 3.6525E+02 d
Time Period For Feeding Hay to Poultry              : 3.6525E+02 d
Time Period For Feeding Hay to Milk Cattle          : 3.6525E+02 d
Time Period For Feeding Hay to Layer Hens           : 3.6525E+02 d
Interception Fraction For Leafy Vegetables          : 0.0000E+00
Interception Fraction For Other Vegetables          : 3.4980E-01
Interception Fraction For Fruits                    : 3.4970E-01
Interception Fraction For Grains                    : 3.4990E-01
Interception Fraction For Beef Forage               : 3.4950E-01
Interception Fraction For Poultry Forage            : 3.4950E-01
Interception Fraction For Milk Cow Forage           : 3.4950E-01
Interception Fraction For Layer Hen Forage          : 3.4950E-01
Interception Fraction For Beef Grain                : 3.4970E-01
Interception Fraction For Poultry Grain             : 3.4970E-01
Interception Fraction For Milk Cow Grain            : 3.4970E-01
Interception Fraction For Layer Hen Grain           : 3.4970E-01
Interception Fraction For Beef Hay                  : 3.4950E-01
Interception Fraction For Poultry Hay               : 3.4950E-01
Interception Fraction For Milk Cow Hay              : 3.4950E-01
Interception Fraction For Layer Hen Hay             : 3.4950E-01
Translocation Factor For Leafy Vegetables           : 5.0000E-01
Translocation Factor For Other Vegetables           : 1.0000E-01
Translocation Factor For Fruits                     : 1.0000E-01
Translocation Factor For Grains                     : 1.0000E-01
Translocation Factor For Beef Forage                : 1.0000E+00
Translocation Factor For Poultry Forage             : 1.0000E+00
Translocation Factor For Milk Cow Forage            : 1.0000E+00
Translocation Factor For Layer Hen Forage           : 1.0000E+00
Translocation Factor For Beef Grain                 : 1.0000E-01
Translocation Factor For Poultry Grain              : 1.0000E-01
Translocation Factor For Milk Cow Grain             : 1.0000E-01
Translocation Factor For Layer Hen Grain            : 1.0000E-01
Translocation Factor For Beef Hay                   : 1.0000E+00
```

```
Translocation Factor For Poultry Hay                          : 1.0000E+00
Translocation Factor For Milk Cow Hay                         : 1.0000E+00
Translocation Factor For Layer Hen Hay                        : 1.0000E+00
Forage Fraction Consumed By Beef Cattle                       : 0.0000E+00
Forage Fraction Consumed By Poultry                           : 1.0000E+00
Forage Fraction Consumed By Milk Cows                         : 1.0000E+00
Forage Fraction Consumed By Layer Hens                        : 1.0000E+00
Grain  Fraction Consumed By Beef Cattle                       : 1.0000E+00
Grain Fraction Consumed By Poultry                            : 1.0000E+00
Grain Fraction Consumed By Milk Cows                          : 1.0000E+00
Grain Fraction Consumed By Layer Hens                         : 1.0000E+00
Hay Fraction Consumed By Beef Cattle                          : 1.0000E+00
Hay Fraction Consumed By Poultry                              : 1.0000E+00
Hay Fraction Consumed By Milk Cows                            : 1.0000E+00
Hay Fraction Consumed By Layer Hens                           : 1.0000E+00
Water Fraction Consumed By Beef Cattle                        : 1.0000E+00
Water Fraction Consumed By Poultry                            : 1.0000E+00
Water Fraction Consumed By Milk Cows                          : 1.0000E+00
Water Fraction Consumed By Layer Hens                         : 1.0000E+00
Crop Yield For Leafy Vegetables                               : 1.0000E+00 kg/m^2
Crop Yield For Other Vegetables                               : 2.4000E+00 kg/m^2
Crop Yield For Fruits                                         : 2.3673E+00 kg/m^2
Crop Yield For Grains                                         : 3.9040E-01 kg/m^2
Crop Yield For Beef Cattle Forage                             : 1.8868E+00 kg/m^2
Crop Yield For Poultry Forage                                 : 1.8868E+00 kg/m^2
Crop Yield For Milk Cow Forage                                : 1.8868E+00 kg/m^2
Crop Yield For Layer Hen Forage                               : 1.8868E+00 kg/m^2
Crop Yield For Beef Cattle Grain                              : 6.5680E-01 kg/m^2
Crop Yield For Poultry Grain                                  : 6.5680E-01 kg/m^2
Crop Yield For Milk Cow Grain                                 : 6.5680E-01 kg/m^2
Crop Yield For Layer Hen Grain                                : 6.5680E-01 kg/m^2
Crop Yield For Beef Cattle Hay                                : 1.8868E+00 kg/m^2
Crop Yield For Poultry Hay                                    : 1.8868E+00 kg/m^2
Crop Yield For Milk Cow Hay                                   : 1.8868E+00 kg/m^2
Crop Yield For Layer Hen Hay                                  : 1.8868E+00 kg/m^2
Wet-to-Dry Weight Conversion Factor For Leafy Vegetables      : 0.0000E+00
Wet-to-Dry Weight Conversion Factor For Other Vegetables      : 1.6200E-01
Wet-to-Dry Weight Conversion Factor For Fruits                : 2.8490E-01
Wet-to-Dry Weight Conversion Factor For Grains                : 8.8000E-01
Wet-to-Dry Weight Conversion Factor For Beef Cattle Forage    : 2.5180E-01
Wet-to-Dry Weight Conversion Factor For Poultry Forage        : 2.5180E-01
Wet-to-Dry Weight Conversion Factor For Milk Cow Forage       : 2.5180E-01
Wet-to-Dry Weight Conversion Factor For Layer Hen Forage      : 2.5180E-01
Wet-to-Dry Weight Conversion Factor For Beef Cattle Grain     : 8.8000E-01
Wet-to-Dry Weight Conversion Factor For Poultry Grain         : 8.8000E-01
Wet-to-Dry Weight Conversion Factor For Milk Cow Grain        : 8.8000E-01
```

```
Wet-to-Dry Weight Conversion Factor For Layer Hen Grain     : 8.8000E-01
Wet-to-Dry Weight Conversion Factor For Beef Cattle Hay     : 2.5180E-01
Wet-to-Dry Weight Conversion Factor For Poultry Hay         : 2.5180E-01
Wet-to-Dry Weight Conversion Factor For Milk Cow Hay        : 2.5180E-01
Wet-to-Dry Weight Conversion Factor For Layer Hen Hay       : 2.5180E-01
Animal Feed Intake Rate For Forage Consumed By Beef Cattle : 1.5000E+01 kg/d
Animal Feed Intake Rate For Forage Consumed By Poultry      : 5.6200E-02 kg/d
Animal Feed Intake Rate For Forage Consumed By Milk Cows    : 3.5165E+01 kg/d
Animal Feed Intake Rate For Forage Consumed By Layer Hens   : 7.5500E-02 kg/d
Animal Feed Intake Rate For Grain Consumed By Beef Cattle   : 2.4188E+00 kg/d
Animal Feed Intake Rate For Grain Consumed By Poultry       : 6.3000E-02 kg/d
Animal Feed Intake Rate For Grain Consumed By Milk Cows     : 1.9466E+00 kg/d
Animal Feed Intake Rate For Grain Consumed By Layer Hens    : 6.1000E-02 kg/d
Animal Feed Intake Rate For Hay Consumed By Beef Cattle     : 1.6253E+01 kg/d
Animal Feed Intake Rate For Hay Consumed By Poultry         : 0.0000E+00 kg/d
Animal Feed Intake Rate For Hay Consumed By Milk Cows       : 2.6109E+01 kg/d
Animal Feed Intake Rate For Hay Consumed By Layer Hens      : 0.0000E+00 kg/d
Water Intake Rate For Beef Cattle                           : 5.0000E+01 L/d
Water Intake Rate For Poultry                               : 3.0000E-01 L/d
Water Intake Rate For Milk Cows                             : 6.0000E+01 L/d
Water Intake Rate For Layer Hens                            : 3.0000E-01 L/d
Water Intake Consumption Period for Beef Cattle             : 3.6525E+02 d
Water Intake Consumption Period for Poultry                 : 3.6525E+02 d
Water Intake Consumption Period for Milk Cow                : 3.6525E+02 d
Water Intake Consumption Period for Layer Hens              : 3.6525E+02 d
Soil Intake Fraction For Beef Cattle                        : 2.0000E-02
Soil Intake Fraction For Poultry                            : 1.0000E-01
Soil Intake Fraction For Milk Cows                          : 2.0000E-02
Soil Intake Fraction For Layer Hens                         : 1.0000E-01
Plant Mass Loading Factor For Leafy Vegetables              : 2.0000E-01 g/g
Plant Mass Loading Factor For Other Vegetables              : 1.0000E-01 g/g
Plant Mass Loading Factor For Fruits                        : 1.0000E-01 g/g
Plant Mass Loading Factor For Grains                        : 1.0000E-01 g/g
Plant Mass Loading Factor For Beef Cattle Forage            : 1.0000E-01 g/g
Plant Mass Loading Factor For Poultry Forage                : 1.0000E-01 g/g
Plant Mass Loading Factor For Milk Cow Forage               : 1.0000E-01 g/g
Plant Mass Loading Factor For Layer Hen Forage              : 1.0000E-01 g/g
Plant Mass Loading Factor For Beef Cattle Grain             : 1.0000E-01 g/g
Plant Mass Loading Factor For Poultry Grain                 : 1.0000E-01 g/g
Plant Mass Loading Factor For Milk Cow Grain                : 1.0000E-01 g/g
Plant Mass Loading Factor For Layer Hen Grain               : 1.0000E-01 g/g
Plant Mass Loading Factor For Beef Cattle Hay               : 1.0000E-01 g/g
Plant Mass Loading Factor For Poultry Hay                   : 1.0000E-01 g/g
Plant Mass Loading Factor For Milk Cow Hay                  : 1.0000E-01 g/g
Plant Mass Loading Factor For Layer Hen Hay                 : 1.0000E-01 g/g
Fraction of Carbon in Beef                                  : 3.6000E-01
```

```
Fraction of Carbon in Poultry                    : 1.8000E-01
Fraction of Carbon in Milk                       : 6.0000E-02
Fraction of Carbon in Eggs                       : 1.6000E-01
Fraction of Carbon in Beef Cattle Forage         : 1.1000E-01
Fraction of Carbon in Poultry Forage             : 1.1000E-01
Fraction of Carbon in Milk Cow Forage            : 1.1000E-01
Fraction of Carbon in Layer Hen Forage           : 1.1000E-01
Fraction of Carbon in Beef Cattle Grain          : 4.0000E-01
Fraction of Carbon in Poultry Grain              : 4.0000E-01
Fraction of Carbon in Milk Cow Grain             : 4.0000E-01
Fraction of Carbon in Layer Hen Grain            : 4.0000E-01
Fraction of Carbon in Beef Cattle Hay            : 7.0000E-02
Fraction of Carbon in Poultry Hay                : 7.0000E-02
Fraction of Carbon in Milk Cow Hay               : 7.0000E-02
Fraction of Carbon in Layer Hen Hay              : 7.0000E-02
Fraction of Carbon in Soil                       : 3.0000E-02
Specific Activity Equivalence of Animal          : 1.0000E+00
Fraction of Hydrogen in Leafy Vegetables         : 1.0000E-01
Fraction of Hydrogen in Other Vegetables         : 1.0000E-01
Fraction of Hydrogen in Fruits                   : 1.0000E-01
Fraction of Hydrogen in Grains                   : 6.8000E-02
Fraction of Hydrogen in Beef                     : 1.0000E-01
Fraction of Hydrogen in Poultry                  : 1.0000E-01
Fraction of Hydrogen in Milk                     : 1.1000E-01
Fraction of Hydrogen in Eggs                     : 1.1000E-01
Fraction of Hydrogen in Beef Cattle Forage       : 1.0000E-01
Fraction of Hydrogen in Poultry Forage           : 1.0000E-01
Fraction of Hydrogen in Milk Cow Forage          : 1.0000E-01
Fraction of Hydrogen in Layer Hen Forage         : 1.0000E-01
Fraction of Hydrogen in Beef Cattle Grain        : 6.8000E-02
Fraction of Hydrogen in Poultry Grain            : 6.8000E-02
Fraction of Hydrogen in Milk Cow Grain           : 6.8000E-02
Fraction of Hydrogen in Layer Hen Grain          : 6.8000E-02
Fraction of Hydrogen in Beef Cattle Hay          : 1.0000E-01
Fraction of Hydrogen in Poultry Hay              : 1.0000E-01
Fraction of Hydrogen in Milk Cow Hay             : 1.0000E-01
Fraction of Hydrogen in Layer Hen Hay            : 1.0000E-01
Fraction of Hydrogen in Soil                     : 5.8000E-03
Tritium Equivalence Between Animal and Food      : 1.0000E+00
Tritium Equivalence Between Plant and Soil       : 1.0000E+00
Tritium Equivalence Between Plant and Water      : 1.0000E+00
Moisture Content of Soil                         : 5.2200E-02 L/kg
TIME DATA:
Start Time                                       : 0.0000E+00 d
End Time                                         : 3.6525E+05 d
DT Size                                          : 3.6525E+02 d
```

```
Time Step Size                                          : 3.6525E+02 d
Write results every                                     :      1 calculation times

INITIAL ACTIVITIES

Number of chains:   3
Chain      Chain      Initial
Number       Name     Activity
                       (pCi/g)
     1  60Co       1.0000E+00
     2  90Sr       1.0000E+00
     3  137Cs      1.0000E+00

Chain No.  1: 60Co

Nuclide   Chain   Half Life    Initial    First  Fractional Second  Fractional   Ingestion
Inhalation    Surface      15 cm
        Position              Inventory  Parent    Yield    Parent    Yield     CEDE Factor
CEDE Factor   Dose Rate   Dose Rate

          Factor      Factor
                     (d)        (pCi/g)                                          (Sv/Bq)
(Sv/Bq)      ((Sv/d)/      ((Sv/d)/

         (Bq/m**2))  (Bq/m**3))

Nuclides in chain :  1

60Co        1    1.9300E+03  1.0000E+00     0       .00000     0       .00000      7.28E-09
5.91E-08    2.03E-10    6.26E-12

Chain No.  2: 90Sr

Nuclide   Chain   Half Life    Initial    First  Fractional Second  Fractional   Ingestion
Inhalation    Surface      15 cm
        Position              Inventory  Parent    Yield    Parent    Yield     CEDE Factor
CEDE Factor   Dose Rate   Dose Rate

          Factor      Factor
                     (d)        (pCi/g)                                          (Sv/Bq)
(Sv/Bq)      ((Sv/d)/      ((Sv/d)/

         (Bq/m**2))  (Bq/m**3))

Nuclides in chain :  2

90Sr        1    1.0600E+04  1.0000E+00     0       .00000     0       .00000      3.85E-08
3.51E-07    2.46E-14    3.21E-16
90Y         2    2.6700E+00  0.0000E+00     1      1.00000     0       .00000      2.91E-09
2.28E-09    4.60E-13    1.03E-14

Chain No.  3: 137Cs

Nuclide   Chain   Half Life    Initial    First  Fractional Second  Fractional   Ingestion
Inhalation    Surface      15 cm
        Position              Inventory  Parent    Yield    Parent    Yield     CEDE Factor
CEDE Factor   Dose Rate   Dose Rate

          Factor      Factor
                     (d)        (pCi/g)                                          (Sv/Bq)
(Sv/Bq)      ((Sv/d)/      ((Sv/d)/

         (Bq/m**2))  (Bq/m**3))
```

```
Nuclides in chain :  2

137Cs        1    1.1000E+04  1.0000E+00     0        .00000     0        .00000      1.35E-08
8.63E-09    2.46E-14    3.40E-16
137mBa  Implicit              0.0000E+00     1        .94600                          0.00E+00
0.00E+00    5.06E-11    1.48E-12

TIMESTEP DATA:

Number of nuclides    :          5
Number of time steps  :        353
Number of print steps:         353
Timestep of maximum   :          5
Day of maximum        :  .1826250000E+04
Year of maximum       :  .5000000000E+01

For Period          5: 1.4610000000E+03 days  to 1.8262500000E+03 days
                       4.0000000000E+00 years to 5.0000000000E+00 years

Dose Components of Maximum TEDE

Nuclide     External    Inhalation  Agricult.   Soil        Drinking    Irrigated   Aquatic
Total       Soil        Water Ave.
            Dose        Dose        Food        Ingestion   Water       Food        Food
Nuclide     Activity    Activity
                                    Ingestion   Dose        Ingestion   Ingestion   Ingestion
Dose
                                    Dose                    Dose        Dose        Dose

            (mrem/y)    (mrem/y)    (mrem/y)    (mrem/y)    (mrem/y)    (mrem/y)    (mrem/y)
(mrem/y)    (pCi/g)     (pCi/l)

Dose Components

    60Co    4.59741E+00 8.27933E-06 1.20780E-01 2.33677E-04 2.86365E-05 2.49104E-05 1.99342E-04
4.71869E+00 5.80502E-01 2.22190E-03
    90Sr    1.62724E-04 3.30068E-05 3.74321E+00 8.53461E-04 3.59509E-01 2.26294E-01 3.79180E-01
4.70924E+00 3.80229E-01 5.27455E+00
    90Y     5.22281E-03 2.14474E-07 7.70141E-02 6.45262E-05 2.68594E-02 1.27920E-02 1.41645E-02
1.36117E-01 3.81204E-01 5.21362E+00
    137Cs   5.25435E-05 2.47344E-07 1.99790E-02 9.12329E-05 5.50361E-01 1.01385E+00 2.32190E+01
2.48033E+01 1.15865E-01 2.30277E+01
    137mBa  2.16368E-01 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00 0.00000E+00
2.16368E-01 1.09608E-01 2.17842E+01
-------------------------------------------------------------------------------------------
-------------------------------
  Totals    4.81922E+00 4.17480E-05 3.96098E+00 1.24290E-03 9.36758E-01 1.25296E+00 2.36125E+01
3.45837E+01

Component Maximums and Time of Occurrence

   Type     Time    Maximum     Time of Occurrence
            Step    Value       (days)      (years)
External      1 9.80159E+00 0.00000E+00 0.00000E+00
Inhalation    1 1.03750E-04 0.00000E+00 0.00000E+00
Agricultur    1 1.04276E+01 0.00000E+00 0.00000E+00
   Soil       1 3.60243E-03 0.00000E+00 0.00000E+00
 Drinking     9 1.04290E+00 2.92200E+03 8.00000E+00
Irrigated     7 1.32010E+00 2.19150E+03 6.00000E+00
  Aquatic     6 2.41395E+01 1.82625E+03 5.00000E+00
   TEDE       5 3.45837E+01 1.46100E+03 4.00000E+00
  60Co        1 7.91971E+00 0.00000E+00 0.00000E+00  Maximum External
  60Co        1 1.42623E-05 0.00000E+00 0.00000E+00  Maximum Inhalation
```

Table E.12 FORTRAN Output for *Modified Defaults Test Case* Session (Continued)

60Co	1	2.08061E-01	0.00000E+00	0.00000E+00	Maximum Agricultural Food Ingestion
60Co	1	4.02542E-04	0.00000E+00	0.00000E+00	Maximum Soil Ingestion
60Co	9	3.58790E-05	2.92200E+03	8.00000E+00	Maximum Drinking Water Ingestion
60Co	9	3.12105E-05	2.92200E+03	8.00000E+00	Maximum Irrigated Food Ingestion
60Co	9	2.49758E-04	2.92200E+03	8.00000E+00	Maximum Aquatic Food Ingestion
60Co	1	8.12821E+00	0.00000E+00	0.00000E+00	Maximum Nuclide
90Sr	1	4.27963E-04	0.00000E+00	0.00000E+00	Maximum External
90Sr	1	8.68076E-05	0.00000E+00	0.00000E+00	Maximum Inhalation
90Sr	1	9.84460E+00	0.00000E+00	0.00000E+00	Maximum Agricultural Food Ingestion
90Sr	1	2.24460E-03	0.00000E+00	0.00000E+00	Maximum Soil Ingestion
90Sr	11	4.81622E-01	3.65250E+03	1.00000E+01	Maximum Drinking Water Ingestion
90Sr	11	3.03159E-01	3.65250E+03	1.00000E+01	Maximum Irrigated Food Ingestion
90Sr	11	5.07974E-01	3.65250E+03	1.00000E+01	Maximum Aquatic Food Ingestion
90Sr	1	9.90564E+00	0.00000E+00	0.00000E+00	Maximum Nuclide
90Y	1	1.35832E-02	0.00000E+00	0.00000E+00	Maximum External
90Y	1	5.45070E-07	0.00000E+00	0.00000E+00	Maximum Inhalation
90Y	1	2.02546E-01	0.00000E+00	0.00000E+00	Maximum Agricultural Food Ingestion
90Y	1	1.67888E-04	0.00000E+00	0.00000E+00	Maximum Soil Ingestion
90Y	11	3.60391E-02	3.65250E+03	1.00000E+01	Maximum Drinking Water Ingestion
90Y	11	1.71370E-02	3.65250E+03	1.00000E+01	Maximum Irrigated Food Ingestion
90Y	11	1.90055E-02	3.65250E+03	1.00000E+01	Maximum Aquatic Food Ingestion
90Y	1	2.19486E-01	0.00000E+00	0.00000E+00	Maximum Nuclide
137Cs	1	4.53488E-04	0.00000E+00	0.00000E+00	Maximum External
137Cs	1	2.13476E-06	0.00000E+00	0.00000E+00	Maximum Inhalation
137Cs	1	1.72433E-01	0.00000E+00	0.00000E+00	Maximum Agricultural Food Ingestion
137Cs	1	7.87407E-04	0.00000E+00	0.00000E+00	Maximum Soil Ingestion
137Cs	6	5.61624E-01	1.82625E+03	5.00000E+00	Maximum Drinking Water Ingestion
137Cs	6	1.03460E+00	1.82625E+03	5.00000E+00	Maximum Irrigated Food Ingestion
137Cs	6	2.36941E+01	1.82625E+03	5.00000E+00	Maximum Aquatic Food Ingestion
137Cs	6	2.53068E+01	1.82625E+03	5.00000E+00	Maximum Nuclide
137mBa	1	1.86741E+00	0.00000E+00	0.00000E+00	Maximum External
137mBa	0	0.00000E+00	0.00000E+00	0.00000E+00	Maximum Inhalation
137mBa	0	0.00000E+00	0.00000E+00	0.00000E+00	Maximum Agricultural Food Ingestion
137mBa	0	0.00000E+00	0.00000E+00	0.00000E+00	Maximum Soil Ingestion
137mBa	0	0.00000E+00	0.00000E+00	0.00000E+00	Maximum Drinking Water Ingestion
137mBa	0	0.00000E+00	0.00000E+00	0.00000E+00	Maximum Irrigated Food Ingestion
137mBa	0	0.00000E+00	0.00000E+00	0.00000E+00	Maximum Aquatic Food Ingestion
137mBa	1	1.86741E+00	0.00000E+00	0.00000E+00	Maximum Nuclide

E.5 DandD Initialization File

DandD uses an initialization file called nrc.ini which is stored in the default Windows directory. The state of the program is saved in the initialization file each time the program exits. Each time the program starts up, the state from the last execution is restored, if possible. If DandD is reinstalled, the initialization file may be lost. The contents of a typical initialization file is shown in table E.13 and includes the following:

[AppFrame] Section - The size and position of the main program window is restored to each time the program is started.

[Application] Section

- Program Working Directory: The last working directory is restored each time the program starts. If the working directory does not exist, the current directory of the executable is used in its place.
- DandD Error Logging: The program message logging flag state is restored. It contains a one if logging is turned on or zero if it is turned off.

[UserInfo] Section - The user name and the organization provided when DandD is run for the first time.

[SessionGroup] - The last loaded session group's name and directory path is stored in the initialization file. DandD will attempt to open this group each time the program starts. If a group no longer exists, the default group will be open instead.

[ReportGroup] - The last loaded report group's name and directory path are stored in the initialization file. DandD will attempt to open this group each time the program starts. If a group no longer exists, the default group will be opened instead.

[RecentFileList] - The last four files that were used by DandD.

Table E.13 DandD Initialization File - NRC.INI

```
[AppFrame]
topleft_x=79
topleft_y=37
bottomright_x=703
bottomright_y=502
show=1

[Application]
WorkDirectory=C:\Program Files\DANDD
ErrorLogging=0

[UserInfo]
Name=I.M. User
Organization=NRC Licensee

[SessionGroup]
Name=sessions.sgp
Path=C:\Program Files\DANDD\sessions.sgp
```

Table E.13 DandD Initialization File - NRC.INI (Continued)

```
[ReportGroup]
Name=nrc.rgp
Path=C:\Program Files\DANDD\nrc.rgp

[Recent File List]
File1=C:\Program Files\DANDD\tables\plant.par
File2=C:\Program Files\DANDD\tables\intake.par
File3=C:\Program Files\DANDD\tables\wetdry.par
File4=C:\Program Files\DANDD\tables\yield.par
```

E.6 History File

This section describes the contents of the file that is used by the graphics processor to show the time history of doses for radionuclides, exposure pathways, and total annual maximum TEDE or percentages of those doses at a particular time step. The file contains doses at intervals specified by the variable parameters described in Section 2.11.3. It is a plain text file that can be imported into a spreadsheet or database. The files are located in the session subdirectory and have a file extension of .his. In order to save the history files for a session, the **Save History** checkbox needs to be selected (see Section 2.5.1.2). The file has a group of records for each time step out that contain the following information (refer to Table E.14).

The first record contains the number of radionuclides and the number of pathways for a particular scenario. The number of radionuclides determines how many records will be in the third group. The next record contains the beginning and end of the timestep in days. The third group of records contains the radionuclide name, the doses for each pathway, the radionuclide dose, the radionuclide activity, and the integrated activity for this radionuclide. The last record in the group contains the totals for each pathway and the total dose (TEDE).

Table E.14 History File						
	3	3				
	0.0000000000E+000	179.000000000000000				
3H	0.0000000000E+000	1.1002998669E-006	4.9507305598E-006	6.0510304267E-006	1.0000000000E+000	9.8636991736E-001
10Be	1.8508962074E-004	6.1771849080E-003	3.6555595925E-004	6.7278304880E-003	1.0000000000E+000	9.9999989377E-001
14C	2.3494354931E-006	3.6365649771E-005	1.6362497085E-004	2.0234005611E-004	1.0000000000E+000	9.9997031797E-001
Total			1.8743905623E-004	6.2146508577E-003	5.3413166065E-004	6.9362215746E-003
	3	3				
	91.312500000000000	270.312500000000000				
3H	0.0000000000E+000	1.0849661886E-006	4.8817376314E-006	5.9667038200E-006	9.8606409143E-001	9.7262395638E-001
10Be	1.8508960068E-004	6.1771842386E-003	3.6555591963E-004	6.7278297589E-003	9.9999989162E-001	9.9999978539E-001
14C	2.3493643445E-006	3.6364548500E-005	1.6362001575E-004	2.0233392859E-004	9.9996971673E-001	9.9994003559E-001
Total			1.8743896502E-004	6.2146337533E-003	5.3405767301E-004	6.9361303913E-003
	3	3				
	182.625000000000000	361.625000000000000				
3H	0.0000000000E+000	1.0698461990E-006	4.8137061821E-006	5.8835523811E-006	9.7232239241E-001	9.5906955785E-001
10Be	1.8508958062E-004	6.1771835691E-003	3.6555588001E-004	6.7278290297E-003	9.9999978324E-001	9.9999967702E-001
14C	2.3492931981E-006	3.6363447262E-005	1.6361506080E-004	2.0232780126E-004	9.9993943437E-001	9.9990975413E-001
Total			1.8743887382E-004	6.2146168626E-003	5.3398464699E-004	6.9360403834E-003
	3	3				
	186.250000000000000	365.250000000000000				
3H	0.0000000000E+000	1.0692503210E-006	4.8110250665E-006	5.8802753876E-006	9.7178083283E-001	9.5853537977E-001
10Be	1.8508957982E-004	6.1771835425E-003	3.6555587844E-004	6.7278290008E-003	9.9999977894E-001	9.9999967271E-001
14C	2.3492903737E-006	3.6363403545E-005	1.6361486409E-004	2.0232755801E-004	9.9993823221E-001	9.9990855202E-001
Total			1.8743887020E-004	6.2146161964E-003	5.3398176760E-004	6.9360368342E-003

Appendix F FORTRAN Source Code

F.1 ACTTST.FOR

```
c
c...this subroutine determines if the integrated activities are
c   decreasing. if they are decreasing the subroutine sets a test
c   value to true to indicate that a maximum tede has been reached.
c   otherwise, the current values of the integrated activities are
c   saved for the next comparison
c
      subroutine acttst(mxlist,nlist,sbar,sbars,dcreas,zero)

      double precision sbar,sbars
      double precision minimum
      integer ilist
      integer mxlist
      integer nlist
      logical dcreas,zero
      dimension sbar(mxlist),sbars(mxlist)
      parameter (minimum=1.0d-03)

c
c...compare current time integrated activities with integrated
c   actvities from previous time. if any of the integrated activities
c   are increasing, set flag to false
      dcreas=.true.
      do 10, ilist=1,nlist
         if (sbar(ilist) .gt. sbars(ilist)) dcreas=.false.
         if (sbar(ilist) .gt. minimum) zero = .false.
         sbars(ilist)=sbar(ilist)
   10 continue
c
c...save the integrated concentrations from this time step
c       do 20, ilist=1,nlist
c          sbars(ilist)=sbar(ilist)
c   20 continue
      end
```

F.2 ADJ.FOR

```
c
c   update coef array with adjoint matrix
c   based on route from parent to progeny
c
      subroutine adj(p,l12,l23,lambda,fr,wr,a)

      double complex a,alpha
      double complex beta
      double complex gamma
      double complex p
      double precision delta
      double precision fr
      double precision l12,l23,lambda
      double precision wr
      dimension a(3,3)
      alpha=p+l12+lambda
      beta=p+l23+lambda
      gamma=p+wr+lambda
      delta=fr*wr
      a(1,1)=beta*gamma
      a(2,1)=l12*gamma
      a(3,1)=l12*l23
      a(1,2)=l23*delta
      a(2,2)=alpha*gamma
      a(3,2)=l23*alpha
      a(1,3)=delta*beta
      a(2,3)=l12*delta
      a(3,3)=alpha*beta
      end
```

F.3 ADOPER.FOR

```
c*********************************************************************
c
c  subroutine adoper; 11/94
c  C. David Updegraff, GRAM, Inc., Albuquerque, NM 87112.
c  Solves the 3-box drinking water decay operator for decay for a
c  time period. Equations B.15-B.18.
c
c*********************************************************************
c
      subroutine adoper(c,a,lambdr,d,t,nr)

      double precision a
      double precision c
      double precision d
      double precision k
      double precision lambdr
      double precision one
      double precision t
      double precision zero
      integer j
      integer maxr
      integer n,nr
      parameter (maxr=60)
      parameter (one=1.0d+0,zero=0.0d+0)
      dimension a(maxr)
      dimension c(maxr)
      dimension d(maxr,maxr)
      dimension k(maxr,maxr)
      dimension lambdr(maxr)
c
      call kdcoef(c,lambdr,d,k,nr,maxr)
c
      do 20, j=1,nr
         a(j)=zero
         do 10, n=1,j
            a(j)=a(j)+k(j,n)*exp(-lambdr(n)*t)
   10    continue
         a(j)=max(zero,a(j))
   20 continue
      end
```

F.4 AGRICL.FOR

```
c
c...this subroutine calculates ingestion dose due to consumption
c   of agricultural products grown on contaminated soil according
c   to Eqns. 5.71 and 5.72
c
      subroutine agricl(nrmax,npmax,ast,dfg,diet,nip,nr,pfs,
     $   prfrac,dgr)

      integer i
      integer j
      integer k
      integer nip,npmax,nr,nrmax
      double precision ast
      double precision diet,dfg,dgr
      double precision pfs,prfrac
      double precision sum
      double precision zero
      parameter (zero=0.0d+0)
      dimension ast(nrmax)
      dimension dgr(nrmax,0:npmax)
      dimension dfg(nrmax,0:npmax)
      dimension nip(nrmax)
      dimension pfs(nrmax,nrmax)
      dimension prfrac(nrmax,0:npmax)
c
c...calculate agricultural dose
      do 30, j=1,nr
         sum=zero
         do 10, i=1,nr
            sum=sum+ast(i)*pfs(j,i)
   10    continue
         do 20, k=0,nip(j)
            dgr(j,k)=diet*prfrac(j,k)*dfg(j,k)*sum
   20    continue
   30 continue
      end
```

F.5 AOPER.FOR

```
c***********************************************************************
c
c  subroutine aoper; 5/93
c  C. David Updegraff, GRAM, Inc., Albuquerque, NM 87112.
c  Solves the 'a' operator for decay for a time period using
c  equation 2.5, equation B.2, and Table 2.4. Uses the subroutine
c  'kcoef' to generate the coefficients.
c***********************************************************************
c
      subroutine aoper(c,a,lambdr,d,t,nr)

      double precision a
      double precision c
      double precision d
      double precision k
      double precision lambdr
      double precision one
      double precision t
      double precision zero
      integer j
      integer maxr
      integer n,nr
      parameter (maxr=20)
      parameter (one=1.0d+0,zero=0.0d+0)
      dimension a(maxr)
      dimension c(maxr)
      dimension d(maxr,maxr)
      dimension k(maxr,maxr)
      dimension lambdr(maxr)
c
      call kcoef(c,lambdr,d,k,nr,maxr)
c
      do 20, j=1,nr
         a(j)=zero
         do 10, n=1,j
            a(j)=a(j)+k(j,n)*exp(-lambdr(n)*t)
   10    continue
         a(j)=max(zero,lambdr(j)*a(j))
   20 continue
      end
```

F.6 AQUATC.FOR

```
c
c...this subroutine calculates the dose due to ingestion of fish
c   according to Eqn.5.7 and 5.78
c
      subroutine aquatc(nrmax,npmax,awt,baf,dfg,
     $   nip,nr,prfrac,tf,uf,dar)

      double precision aff,awt
      double precision baf
      double precision dar,dfg
      double precision prfrac
      double precision tf
      double precision uf
      double precision year
      double precision zero
      integer j
      integer k
      integer nip,npmax,nr,nrmax
      parameter (year=365.25d+0)
      parameter (zero=0.0d+0)
      dimension awt(nrmax)
      dimension baf(nrmax)
      dimension dar(nrmax,0:npmax)
      dimension dfg(nrmax,0:npmax)
      dimension nip(nrmax)
      dimension prfrac(nrmax,0:npmax)
c
c...calculate dose due to ingestion of fish
      do 20, j=1,nr
         aff=awt(j)*uf*tf*baf(j)/year
         do 10, k=0,nip(j)
            dar(j,k)=prfrac(j,k)*dfg(j,k)*aff
   10    continue
   20 continue
      end
```

F.7 AROPER.FOR

```
c
c       calculate the soil activity of each radionuclide at the
c       beginning of the time of interest. See Section B.7.
c

      subroutine aroper(maxpar,maxprg,nprog,k,root,t,c)

      double complex c
      double complex k
      double complex root
      integer ibox,ipar,iprog,iroot
      integer maxpar,maxprg
      integer nbox,nprog,nroot
      double precision t
      double precision zero
      dimension c(3,maxprg)
      dimension k(3,3,maxpar,maxprg)
      dimension root(3,maxpar)
      parameter (nbox=3,nroot=3)
      parameter (zero=0.0d+0)
      do 40, iprog=1,nprog
         do 30, ibox=1,nbox
            c(ibox,iprog)=zero
            do 20, ipar=1,iprog
               do 10, iroot=1,nroot
                  c(ibox,iprog)=c(ibox,iprog)+k(ibox,iroot,ipar,iprog)*
     $               cdexp(root(iroot,ipar)*t)
   10          continue
   20       continue
CCC...added a check for small numbers that turn negative;
CCC...zero those numbers out
            if (dble(c(ibox,iprog)) .lt. zero) c(ibox,iprog)=zero
   30    continue
   40 continue
      end
```

F.8 BLDDEF.FOR

```
c
c...this subroutine reads the building occupancy scenario default
c   parameter file. The variables read in are:
c   time in building (t), occupancy period (tt), breathing rate (v),
c   resuspension factor (dust), ingestion rate (g), start time (tstart),
c   end time (tend), time step size (dt), and history file time step
c   skipping factor (tskip).
c
      subroutine blddef(in,t,tt,v,dust,g,tstart,tend,dt,skip,ioerr)

      double precision dust
      double precision g
      double precision t,tt
      double precision v
      double precision dt
      double precision tend,tstart,tskip
      integer in,ioerr,skip

      read(in,*,iostat=ioerr) t,tt,v,dust,g,tstart,tend,dt,tskip

      if (ioerr .ne. 0) then
            call ioreturn(ioerr)
            return
      endif

      skip = tskip
      end
```

F.9 BOXSET.FOR

```
c
c......set up the coefficients for the 3-box model
c

      include "cdecl.inc"

      subroutine boxset(rc,mxbox,mxpar,nr,kcoef,root,iconn,
     $   co,d,112,123,lambda,fr,wr,ioerr)

      character*80 status_msg
      double complex root
      double complex kcoef
      double precision co,coi
      double precision d
      double precision fr
      double precision 112,123,lambda
      double precision wr
      double precision zero
      integer ibox,iconn,ioerr,ipar,iprog,iroot
      integer nbox,nr,nroot
      integer maxpar,mxbox,mxpar
      integer rc
      parameter (maxpar=20)
      dimension co(mxbox,mxpar),coi(3,maxpar)
      dimension d(mxpar*(mxpar-1)/2)
      dimension iconn(mxpar*(mxpar-1)/2)
      dimension kcoef(3,3,mxpar,mxpar)
      dimension 112(mxpar),123(mxpar)
      dimension lambda(mxpar)
      dimension root(3,mxpar)
      parameter (nbox=3,nroot=3)
      parameter (zero=(0.0d+0,0.0d+0))
c
c...determine all possible route pairs from one radionuclide
c   to another
      call connct(mxpar,iconn)

c
c...calculate roots of laplace transform solution to 3 box problem
c......initialize roots to zero
      do 20, iprog=1,mxpar

         do 10, iroot=1,3
            root(iroot,iprog)=zero
   10    continue

   20 continue


c......calculate roots
      do 30, ipar=1,nr
         call cubic(root(1,ipar),fr,wr,112(ipar),
     $       123(ipar),lambda(ipar))
   30 continue

c
c...initialize initial condition array
```

```
      do 50, ipar=1,nr

         do 40, ibox=1,nbox
            coi(ibox,ipar)=co(ibox,ipar)/lambda(ipar)
   40    continue

   50 continue
c
c...calculate coefficients for exponentials
      do 80, iprog=1,nr

         write(status_msg(1:10),'(i10)',iostat=ioerr) iprog

          if (ioerr .ne. 0) then
            call ioretint(ioerr)
            return
          endif

         status_msg = status_msg(1:10)//'  coefficient(s) out of 'C

         write(status_msg(34:36),'(i3)',iostat=ioerr) nr

          if (ioerr .ne. 0) then
            call ioretint(ioerr)
            return
          endif


         status_msg = status_msg(1:36)//' calculating...'C
         call CMessage(status_msg,rc)

         if (rc .eq. 0) then
             return
         endif

         do 70, ipar=1,iprog

            do 60, iroot=1,nroot

               call coeff(rc,mxpar,iroot,ipar,iprog,iconn,
     $            root(iroot,ipar),root,coi,d,lambda,l12,l23,fr,wr,
     $            kcoef(1,iroot,ipar,iprog))

               if (rc .eq. 0) then
                  return
               endif

   60       continue
   70    continue
   80 continue
      end
```

F.10 C1MEXP

```
c     This function is used by the Sr operator
c     described in equations in Section B.7.
c     It is called by the sroper subroutine.

      double complex function c1mexp(x)

      double complex one
      double complex x
      double complex y
      double precision count
      double precision epslon
      double precision small
      parameter (epslon=1.0d-17,one=(1.0d+0,0.0d+0),small=1.06d-1)
      if (cdabs(x) .gt. small) then
         c1mexp=one-exp(-x)
      else
         count=one
         c1mexp=x
         y=x
    5    count=count+one
         y=-y*x/count
         c1mexp=c1mexp+y
         if (cdabs(y) .gt. epslon*cdabs(c1mexp)) go to 5
      endif
      end
```

F.11 CDECL.INC

```
 INTERFACE TO SUBROUTINE
$     CMessage [C,ALIAS:'_CMessage'] (str,rc)
 character*(*) str
 integer rc [reference]
 END

 INTERFACE TO SUBROUTINE
$     CProcess [C,ALIAS:'_CProcess'] (rc)
 integer rc [reference]
 END
```

F.12 CLOSE.FOR

```
c
c close all files opened in the main program
c
c
      subroutine closefiles(startfn, nofiles)


      integer startfn, nofiles
      integer i, endfn,ioerr

      endfn = startfn + nofiles

      do 10, i=startfn,endfn
          close(i,iostat=ioerr)

          if (ioerr .ne. 0) then
            call ioreturn(ioerr)
            return
          endif

10    continue
      end
```

F.13 COEFF.FOR

```
c
c...this subroutine calculates the k(ibox,ipar) coefficient for the
c   iprog'th progeny radionuclide in the three box model.
c

ccc       include "cdecl.inc"

      subroutine coeff(rc,maxprg,indrot,indpar,indprg,
     $   iconn,alpha,root,coi,d,lambda,112,123,fr,wr,kcoef)

      double complex aadj,alpha
      double complex coef,coef1
      double complex denom
      double complex kcoef
      double complex root
      double complex zero
      double precision coi
      double precision d
      double precision fr
      double precision 112,123,lambda
      double precision one
      double precision wr
      integer ibox,iconn,indpar,indprg,indrot,ipair
      integer ipar,iparvl,iprog,ipth,iroot,iroute,irout1
      integer maxprg
      integer nbox,nroot
      integer rc
      dimension aadj(3,3)
      dimension coef(3),coef1(3)
      dimension coi(3,maxprg)
      dimension d(maxprg*(maxprg-1)/2)
      dimension iconn(maxprg*(maxprg-1)/2)
      dimension kcoef(3)
      dimension 112(maxprg),123(maxprg),lambda(maxprg)
      dimension root(3,maxprg)
      parameter (nbox=3,nroot=3)
      parameter (one=1.0d+0)
      parameter (zero=(0.0d+0,0.0d+0))
c
c...initialize k coefficient
      do 10, ibox=1,nbox
         kcoef(ibox)=zero
   10 continue
c
c...initialize coef array with the initial box concentration
      do 140, iroute=2**(indprg-1),2**indprg-1
c
c...call back to the GUI to process Aborts
c
ccc         call CProcess(rc)
         if (rc .eq. 0) then
             return
         endif

         do 30, ipar=1,indprg
            iparvl=2**(ipar-1)
            if (iand(iroute,iparvl) .eq. iparvl) then
```

```
                do 20, ibox=1,3
                   coef(ibox)=coi(ibox,ipar)
   20           continue
                goto 35
             endif
   30     continue
   35     continue
c
c...update coef array with fractional yield
c   based on connection in pathway
          irout1=iroute
          do 70, ipth=1,indprg-1
             do 60, ipair=1,indprg*(indprg-1)/2
                if (iand(irout1,iconn(ipair)) .eq. iconn(ipair)) then
                   do 40, ibox=1,nbox
                      coef(ibox)=coef(ibox)*d(ipair)
   40              continue
                   do 50, ipar=1,indprg
                      iparvl=2**(ipar-1)
                      if (iand(irout1,iparvl) .eq. iparvl) then
                         irout1=ieor(irout1,iparvl)
                         goto 65
                      endif
   50              continue
                endif
   60        continue
   65        continue
   70     enddo
c
c...update coef array with decay constant and adjoint matrix
c   based on route from parent to progeny
          do 120, ipar=1,indprg
             iparvl=2**(ipar-1)
             if (iand(iroute,iparvl) .eq. iparvl) then
                do 80, ibox=1,nbox
                   coef(ibox)=coef(ibox)*lambda(ipar)
   80           continue
                call adj(alpha,l12(ipar),l23(ipar),lambda(ipar),
     $             fr,wr,aadj)
                call mvmltc(3,3,aadj,coef,coef1)
                do 90, ibox=1,nbox
                   coef(ibox)=coef1(ibox)
   90           continue
             else
                do 110, ibox=1,nbox
                   do 100, iroot=1,nbox
                      coef(ibox)=coef(ibox)*(alpha-root(iroot,ipar))
  100              continue
  110           continue
             endif
  120     continue
c
c...increment the kcoef array with the coef array
          do 130, ibox=1,nbox
             kcoef(ibox)=kcoef(ibox)+coef(ibox)
  130     enddo
  140 enddo
c
c...calculate denominator
```

```
      denom=one
      do 160, iroot=1,nroot
         do 150, iprog=1,indprg
            if (iroot .ne. indrot .or. iprog .ne. indpar)
     $         denom=denom*(alpha-root(iroot,iprog))
  150    continue
  160 continue
c
c...divide denominator into k coefficient
      do 170, ibox=1,nbox
         kcoef(ibox)=kcoef(ibox)/denom
  170 continue
      end
```

F.14 CONCEN.FOR

```
c
c...sets up the initial activity for a chain.  if the chain is
c   not in equilibrium, the activity of the first radionuclide is set
c   to the value read in the input file and the rest are set to zero.
c   if the chain is an equilibrium chain, the activity of the first
c   is set to the activity read in the input file and the rest are
c   set to the parent concentration times the fractional yield of the
c   parent times ( lambda divided by lambda minus lambda of the first
c   radionuclide in the chain ).

      subroutine concen(mxnr,mxnip,nr,echain,distrb,lambda,ipar,
     $    frac,prfrac,nip,conc,con)

      double precision con,conc
      double precision frac
      double precision lambda
      double precision prfrac
      double precision small,sum
      double precision zero
      integer i,ipar
      integer j
      integer mxnip,mxnr
      integer nip,nr
      logical distrb
      logical echain
      dimension con(mxnr)
      dimension frac(2,mxnr)
      dimension ipar(2,mxnr)
      dimension lambda(mxnr)
      dimension nip(mxnr)
      dimension prfrac(mxnr,0:mxnip)
      parameter (small=1.0d-50)
      parameter (zero=0.0d+0)
c
c...set initial activity for first radionuclide
      con(1)=conc
c...if chain is an equilibrium chain, set the initial activities of
c   all remaining radionuclides in the chain to the equilibrium
c   value
      if (echain) then
         do 10, i=2,nr
            con(i)=con(ipar(1,i))*frac(1,i)
            if (ipar(2,i) .gt. 0)
     $         con(i)=con(i)+con(ipar(2,i))*frac(2,i)
            con(i)=con(i)*(lambda(i)/(lambda(i)-lambda(1)))
   10    continue
         if (distrb) then
            sum=zero
            do 30, i=1,nr
               do 20, j=0,nip(i)
                  sum=sum+prfrac(i,j)*con(i)
   20          continue
   30       continue
            if (sum .gt. small) then
               do 40, i=1,nr
                  con(i)=con(i)*conc/sum
   40          continue
```

```
            else
               do 50, i=1,nr
                  con(i)=zero
   50          continue
            endif
         endif
c...if not an equilibrium chain, set all remaining initial
c   activities to zero
      else
         do 60, i=2,nr
            con(i)=zero
   60    continue
      endif
      end
```

F.15 CONCLC.FOR

```
c
c...this subroutine calculates the value of the derivatives
c   (i.e., dy/dt=f) from the values of the dependent variable, y,
c   at the current time, t. this subroutine is called by 'lsodes',
c   which is a stiff differential equation solver. the subroutine
c   takes advantage of the sparsity structure of the differential
c   equations.
c

      subroutine conclc(neq,t,y,ydot)

      double precision ddd
      double precision t
      double precision y,ydot
      double precision zero
      integer i,ia,ichn
      integer j,ja
      integer maxbox,maxchn,maxneq,maxnnz,maxnuc
      integer neq
      parameter (maxbox=12,maxchn=22,maxnuc=20)
      parameter (maxneq=maxbox*maxnuc)
      parameter (maxnnz=maxbox*maxnuc*(maxnuc+3)/2)
      parameter (zero=0.0d+0)
      dimension ddd(maxnnz,maxchn)
      dimension ia(maxneq+1,maxchn)
      dimension ja(maxnnz,maxchn)
      dimension y(neq),ydot(neq)
      common /chnnum/ ichn
      common /ddd/ ddd
      common /ia/ ia
      common /ja/ ja
c
c...initialize ydot array
      do 10, i=1,neq
         ydot(i)=zero
   10 continue
c
c...calculate ydot array
      do 30, i=1,neq
         do 20, j=ia(i,ichn),ia(i+1,ichn)-1
            ydot(ja(j,ichn))=ydot(ja(j,ichn))+y(i)*ddd(j,ichn)
   20    continue
   30 continue
      end
```

F.16 CONCRD.FOR

```
c
c...this subroutine reads the list of radionuclides in the
c   'import concentration' data file.  the first radionuclide in each
c   chain is assigned the chain name.  the routine does some simple
c   checking to make sure array sizes are not exceeded, end of files
c   are not reached, and the radionuclide name are valid.
c
      subroutine concrd(con,out,maxchn,maxnam,maxpar,
     $   nnam,nucnam,chain,ierr,nchn,nr,nucrd,ioerr)

      character*6 nucnam,nucrd
      character*8 chain
      integer con
      integer ichn,ierr,ioerr,ioerr2,ir
      integer maxchn,maxnam,maxpar
      integer nchn,nnam,nr
      integer out
      logical isanuc
      dimension chain(maxchn)
      dimension nr(maxchn)
      dimension nucnam(maxnam)
      dimension nucrd(maxpar,maxchn)
c
      ierr=0
c...read number of chains
      read(con,*,iostat=ioerr) nchn

      if (ioerr .lt. 0) then

         write(out,'(2a)',iostat=ioerr2) 'End of file reached reading ',
     $      '''nchn'' record in import concentration file'

          if (ioerr2 .ne. 0) then
              call ioreturn(ioerr2)
              ioerr = ioerr2
              return
          endif

         ierr=101

      elseif (ioerr .gt. 0) then

         write(out,'(2a)',iostat=ioerr2) 'Error reading ''nchn''',
     $      ' record in import concentration file'

          if (ioerr2 .ne. 0) then
              call ioreturn(ioerr2)
              ioerr = ioerr2
              return
          endif

         ierr=102

      endif

      if (ierr .ne. 0) return
```

```
      if (nchn .gt. maxchn) then

         write(out,'(2(a,i2),a)',iostat=ioerr2)
     $      'Number of chains in import concentration file (',nchn,
     $      ') exceeds maximum allowable (',maxchn,')'

          if (ioerr2 .ne. 0) then
              call ioreturn(ioerr2)
              ioerr = ioerr2
              return
          endif

         ierr=103

      endif

      if (ierr .ne. 0) return
c
      do 20, ichn=1,nchn
c......read number of explicit radionuclides in chain
         read(con,*,iostat=ioerr) nr(ichn)

         if (ioerr .lt. 0) then
            write(out,'(2a,i2,a)',iostat=ioerr2)
     $         'End of file reached reading ''nr''',
     $         ' record for chain ',ichn,' in import concentration file'

              if (ioerr2 .ne. 0) then
                  call ioreturn(ioerr2)
                  ioerr = ioerr2
                  return
              endif

            ierr=104

         elseif (ioerr .gt. 0) then

            write(out,'(2a,i2,a)',iostat=ioerr2)
     $         'Error reading ''nr'' record ',
     $         'for chain ',ichn,' in import concentration file'

              if (ioerr2 .ne. 0) then
                  call ioreturn(ioerr2)
                  ioerr = ioerr2
                  return
              endif

            ierr=105

         endif

         if (ierr .ne. 0) return

         if (nr(ichn) .gt. maxpar) then

            write(out,'(3(a,i2),a)',iostat=ioerr2)
     $         'Number of explicit radionuclides in chain ',ichn,
     $         ' of import concentration file (',nr(ichn),
     $         ') exceeds maximum allowable (',maxpar,')'
```

```
            if (ioerr2 .ne. 0) then
                call ioreturn(ioerr2)
                ioerr = ioerr2
                return
            endif

          ierr=106

        endif

        if (ierr .ne. 0) return

c......read names of explicit radionuclides in chain
        do 10, ir=1,nr(ichn)

          read(con,'(a)',iostat=ioerr) nucrd(ir,ichn)

          if (ioerr .lt. 0) then
             write(out,'(2a,2(i2,a))',iostat=ioerr2)
     $           'End of file reached ',
     $           'reading radionuclide ',ir,' of chain ',ichn,
     $           ' in import concentration file'

            if (ioerr2 .ne. 0) then
                call ioreturn(ioerr2)
                ioerr = ioerr2
                return
            endif

             ierr=107

          elseif (ioerr .gt. 0) then

             write(out,'(2a,3(i2,a))',iostat=ioerr2) 'Error ',
     $           'reading radionuclide ',ir,' out of ',nr(ichn),
     $           ' of chain ',ichn,
     $           ' in import concentration file'

            if (ioerr2 .ne. 0) then
                call ioreturn(ioerr2)
                ioerr = ioerr2
                return
            endif

             ierr=108

          endif

          if (ierr .ne. 0) return

          if (.not. isanuc(nnam,nucrd(ir,ichn),nucnam)) then

             write(out,'(4a)',iostat=ioerr2) 'Radionuclide name ''',
     $           nucrd(ir,ichn),''' from import concentration',
     $           ' file is not a valid radionuclide name'

            if (ioerr2 .ne. 0) then
                call ioreturn(ioerr2)
                ioerr = ioerr2
```

```
            return
          endif

           ierr=109

        endif

        if (ierr .ne. 0) return

   10   continue

c......assign first radionuclide in chain to chain name
        chain(ichn)=nucrd(1,ichn)

   20 continue
      end
```

F.17 CONNCT.FOR

```
c
c...this subroutine sets up the ordered pair of connections between
c   two radionuclides in a chain.  the subroutine generates integers
c   whose binary representation indicates the connections.  For
c   instance, the integer 5, whose binary representation is (1 0 1)
c   indicates a connection between radionuclide 1 and radionuclide 3
c   since the first and third bits of the binary representation are
c   set.
c
      subroutine connct(maxrad,iconn)

      integer i,iconn,icount,ival
      integer j
      integer maxrad
      dimension iconn(maxrad*(maxrad-1)/2)
      icount=0
      do 20, i=1,maxrad
         ival=2**(i-1)
         do 10, j=1,i-1
            icount=icount+1
            iconn(icount)=ival+2**(j-1)
   10    continue
   20 continue
      end
```

F.18 CONRES.FOR

```
c...this subroutine sets up the initial concentration of each
c   radionuclide in a chain into a format useable by the
c   residential scenario analytical solution
c

      subroutine conres(maxbox,maxnr,nbox,nr,con,conr)

      double precision con,conr
      double precision zero
      integer ibox,ir
      integer maxbox,maxnr
      integer nbox,nr
      dimension con(maxnr)
      dimension conr(maxbox,maxnr)
      parameter (zero=0.0d+0)
c
c...initialize all concentrations in initial concentration array to zero
      do 20, ir=1,maxnr
         do 10, ibox=1,nbox
            conr(ibox,ir)=zero
   10    continue
   20 continue
c
c...store initial concentrations in surface layer
      do 30, ir=1,nr
         conr(1,ir)=con(ir)
   30 continue
      end
```

F.19 CONTST.FOR

```
c......if an import concentration file is used, check for chain
c      compatibility; if not, stop

      subroutine contst(out,maxpar,nr,nrrd,nuclid,nucrd,ioerr)

      character*6 nuclid,nucrd
      integer ierr,ioerr,ir
      integer maxpar
      integer nr,nrrd
      integer out
      dimension nuclid(maxpar),nucrd(maxpar)
c
      ierr=0

      if (nr .ne. nrrd) then
         write(out,'(5a)',iostat=ioerr)
     $      'Length of chain in database does not ',
     $      'match length of chain in import concentration file ',
     $      'for chain ''',nuclid(1),''''

          if (ioerr .ne. 0) then
              call ioreturn(ioerr)
              return
          endif


         ierr=301

      endif

      if (ierr .ne. 0) then
           ioerr=ierr
           return
      endif

      do 10, ir=1,nr

         if (nuclid(ir) .ne. nucrd(ir)) then
            write(out,'(5a)',iostat=ioerr)
     $         'Radionuclide list in database does ',
     $         'not match radionuclide list in import ',
     $         'concentration data file for chain ''',nuclid(1),''''

              if (ioerr .ne. 0) then
                  call ioreturn(ioerr)
                  return
              endif


            write(out,'(5a)',iostat=ioerr) 'Found ''',nucrd(ir),
     $         ''' when expecting to find ''',nuclid(ir),''''

              if (ioerr .ne. 0) then
                  call ioreturn(ioerr)
                  return
              endif
```

```
        ierr=302

      endif

       if (ierr .ne. 0) then
           ioerr=ierr
           return
       endif


10 continue
   end
```

F.20 CONVRT.FOR

```
c
c...this subroutine converts the Sv-Bq sytems of dose units to the
c   mrem-pCi system
c
      subroutine convrt(mxnr,mxnip,nr,nip,
     $   dfg,dfh,dfes,dfer)

      integer iip,ir
      integer nip,nr
      integer mxnip,mxnr
      double precision dfg,dfh,dfes,dfer
      double precision factr1,factr2,factr3,factr4
      dimension dfg(mxnr,0:mxnip)
      dimension dfh(mxnr,0:mxnip)
      dimension dfes(mxnr,0:mxnip)
      dimension dfer(mxnr,0:mxnip)
      dimension nip(mxnr)
      data factr1 /3.7d+3/, factr2 /3.7d+3/
      data factr3 /6.944d+3/, factr4/2.505d+8/
      do 20, ir=1,nr
         do 10, iip=0,nip(ir)
            dfg(ir,iip)=factr1*dfg(ir,iip)
            dfh(ir,iip)=factr2*dfh(ir,iip)
            dfes(ir,iip)=factr3*dfes(ir,iip)
            dfer(ir,iip)=factr4*dfer(ir,iip)
   10    continue
   20 continue
      end
```

F.21 CORH.FOR

```
c
c...this subroutine determines if a radionuclide is either
c   carbon-14 (c-14) or tritium (h-3) for using the special
c   carbon or tritium models
c
      subroutine corh(mxchn,nchn,nuclid,carbon,hydrog)

      character*6 nuclid
      integer ichn
      integer mxchn
      integer nchn
      logical carbon
      logical hydrog
      dimension carbon(mxchn)
      dimension hydrog(mxchn)
      dimension nuclid(mxchn)
c
      do 10, ichn=1,nchn
         carbon(ichn)=.false.
         hydrog(ichn)=.false.
         if (nuclid(ichn) .eq. '14C') carbon(ichn)=.true.
         if (nuclid(ichn) .eq. '3H') hydrog(ichn)=.true.
   10 continue
      end
```

F.22 CUBIC.FOR

```
c
c...calculate roots of laplace transform solution to 3 box problem.
c   called by boxset.
c
      subroutine cubic(y,fr,wr,l12,l23,lambda)

      double complex abig
      double complex bbig,beta,bigadd,bigsub
      double complex x
      double complex y
      double precision a,alpha
      double precision b
      double precision four,fr
      double precision l12,l23,lambda
      double precision nine
      double precision one
      double precision p,pd3
      double precision q
      double precision r
      double precision third,three,twnty7,two
      double precision wr
      double precision zero
      integer i
      dimension x(3)
      dimension y(3)
      parameter (four=4.0d+0)
      parameter (nine=9.0d+0)
      parameter (one=1.0d+0)
      parameter (third=1.0d+0/3.0d+0,three=3.0d+0)
      parameter (twnty7=2.7d+1,two=2.0d+0)
      parameter (zero=0.0d+0)
      p=l12+l23+wr
      q=l12*l23+l12*wr+l23*wr
      r=(one-fr)*l12*l23*wr
      a=(three*q-p*p)/three
      b=((two*p*p-nine*q)*p+twnty7*r)/twnty7
      alpha=-b/two
      beta=cdsqrt(b*b/four+a*a*a/twnty7)
      abig=(alpha+beta)**third
      bbig=(alpha-beta)**third
      bigadd=abig+bbig
      bigsub=abig-bbig
      x(1)=bigadd
      x(2)=(-bigadd+bigsub*cdsqrt(-three))/two
      x(3)=(-bigadd-bigsub*cdsqrt(-three))/two
      pd3=p/three
      do 10, i=1,3
         y(i)=x(i)-pd3-lambda
   10 continue
      end
```

F.23 D1MEXP.FOR

```
c     This function is used by the R and Re operators
c     described in equations (B.2-B.5) and (B.15-B.18).
c     It is called by the roper and reoper subroutines.

      double precision function d1mexp(x)
      double precision count
      double precision epslon
      double precision one
      double precision small
      double precision x
      double precision y
      parameter (epslon=1.0e-17,one=1.0d+0,small=1.06d-1)
      if (x .gt. small) then
         d1mexp=one-exp(-x)
      else
         count=one
         d1mexp=x
         y=x
    5    count=count+one
         y=-y*x/count
         d1mexp=d1mexp+y
         if (abs(y) .gt. epslon*abs(d1mexp)) go to 5
      endif
      end
```

F.24 DDARAY.FOR

```
      subroutine ddaray(maxpar,maxvar,d,112,123,lambda,nr,dd)

      double precision d,dd
      double precision factor
      double precision 112,123,lambda
      double precision zero
      integer i
      integer j,j0,j1,j2,j3
      integer maxpar,maxvar
      integer n,n3,nr
      parameter (zero=0.0d+0)
      dimension d(maxpar,maxpar)
      dimension dd(maxvar,maxvar)
      dimension 112(maxpar),123(maxpar),lambda(maxpar)
c
c...initialize the dd array
c     call dinitz(nvmax*nvmax,dd)
      do 20, j=1,maxvar
         do 10, i=1,maxvar
            dd(i,j)=zero
   10    continue
   20 continue
c
c...fill in the dd array for the drinking water scenario, 3-box model
c   in a manaer similar to the d array
      do 40, j=1,nr
         j0=3*(j-1)
         j1=j0+1
         j2=j1+1
         j3=j2+1
         dd(j1,j2)=112(j)
         dd(j2,j3)=123(j)
         do 30, n=1,j-1
            if (d(n,j) .gt. zero) then
               n3=3*n
               factor=lambda(j)*d(n,j)
               dd(n3-2,j1)=factor
               dd(n3-1,j2)=factor
               dd(n3,j3)=factor
            endif
   30    continue
   40 continue
      end
```

F.25 DFOCON.FOR

```
c
c...this subroutine converts the organ dose factors from Sv/Bq
c   in table E.3 to mrem/pCi. Used by eq. (4.21) and (4.22).
c

      subroutine dfocon(mxnip,mxnr,mxorg,nip,norg,nr,dfo)

      double precision dfo
      double precision factor
      integer iip,iorg,ir
      integer mxnip,mxnr,mxorg
      integer nip,norg,nr
      parameter (factor=3.7d+3)
      dimension dfo(mxorg,mxnr,0:mxnip)
      dimension nip(mxnr)
      do 30, ir=1,nr
         do 20, iip=0,nip(ir)
            do 10, iorg=1,norg
               dfo(iorg,ir,iip)=dfo(iorg,ir,iip)*factor
   10       continue
   20    continue
   30 continue
      end
```

F.26 DLSODES.FOR

```
      subroutine lsodes (f, neq, y, t, tout, itol, rtol, atol, itask,
     1            istate, iopt, rwork, lrw, iwork, liw, jac, mf)
      external f, jac
      integer neq, itol, itask, istate, iopt, lrw, iwork, liw, mf
      double precision y, t, tout, rtol, atol, rwork
      dimension neq(1), y(1), rtol(1), atol(1), rwork(lrw), iwork(liw)
c-----------------------------------------------------------------------
c this is the march 30, 1987 version of
c lsodes.. livermore solver for ordinary differential equations
c          with general sparse jacobian matrices.
c this version is in double precision.
c
c lsodes solves the initial value problem for stiff or nonstiff
c systems of first order ode-s,
c     dy/dt = f(t,y) ,  or, in component form,
c     dy(i)/dt = f(i) = f(i,t,y(1),y(2),...,y(neq)) (i = 1,...,neq).
c lsodes is a variant of the lsode package, and is intended for
c problems in which the jacobian matrix df/dy has an arbitrary
c sparse structure (when the problem is stiff).
c
c authors..       alan c. hindmarsh,
c                 computing and mathematics research division, 1-316
c                 lawrence livermore national laboratory
c                 livermore, ca 94550.
c
c and             andrew h. sherman
c                 j. s. nolen and associates
c                 houston, tx 77084
c-----------------------------------------------------------------------
c references..
c 1.  alan c. hindmarsh,  odepack, a systematized collection of ode
c     solvers, in scientific computing, r. s. stepleman et al. (eds.),
c     north-holland, amsterdam, 1983, pp. 55-64.
c
c 2.  s. c. eisenstat, m. c. gursky, m. h. schultz, and a. h. sherman,
c     yale sparse matrix package.. i. the symmetric codes,
c     int. j. num. meth. eng., 18 (1982), pp. 1145-1151.
c
c 3.  s. c. eisenstat, m. c. gursky, m. h. schultz, and a. h. sherman,
c     yale sparse matrix package.. ii. the nonsymmetric codes,
c     research report no. 114, dept. of computer sciences, yale
c     university, 1977.
c-----------------------------------------------------------------------
c summary of usage.
c
c communication between the user and the lsodes package, for normal
c situations, is summarized here.  this summary describes only a subset
c of the full set of options available.  see the full description for
c details, including optional communication, nonstandard options,
c and instructions for special situations.  see also the example
c problem (with program and output) following this summary.
c
c a. first provide a subroutine of the form..
c                subroutine f (neq, t, y, ydot)
c                dimension y(neq), ydot(neq)
c which supplies the vector function f by loading ydot(i) with f(i).
c
```

```
c b. next determine (or guess) whether or not the problem is stiff.
c stiffness occurs when the jacobian matrix df/dy has an eigenvalue
c whose real part is negative and large in magnitude, compared to the
c reciprocal of the t span of interest.  if the problem is nonstiff,
c use a method flag mf = 10.  if it is stiff, there are two standard
c for the method flag, mf = 121 and mf = 222.  in both cases, lsodes
c requires the jacobian matrix in some form, and it treats this matrix
c in general sparse form, with sparsity structure determined internally.
c (for options where the user supplies the sparsity structure, see
c the full description of mf below.)
c
c c. if the problem is stiff, you are encouraged to supply the jacobian
c directly (mf = 121), but if this is not feasible, lsodes will
c compute it internally by difference quotients (mf = 222).
c if you are supplying the jacobian, provide a subroutine of the form..
c               subroutine jac (neq, t, y, j, ian, jan, pdj)
c               dimension y(1), ian(1), jan(1), pdj(1)
c here neq, t, y, and j are input arguments, and the jac routine is to
c load the array pdj (of length neq) with the j-th column of df/dy.
c i.e., load pdj(i) with df(i)/dy(j) for all relevant values of i.
c the arguments ian and jan should be ignored for normal situations.
c lsodes will call the jac routine with j = 1,2,...,neq.
c only nonzero elements need be loaded.  usually, a crude approximation
c to df/dy, possibly with fewer nonzero elements, will suffice.
c
c d. write a main program which calls subroutine lsodes once for
c each point at which answers are desired.  this should also provide
c for possible use of logical unit 6 for output of error messages
c by lsodes.  on the first call to lsodes, supply arguments as follows..
c f      = name of subroutine for right-hand side vector f.
c          this name must be declared external in calling program.
c neq    = number of first order ode-s.
c y      = array of initial values, of length neq.
c t      = the initial value of the independent variable.
c tout   = first point where output is desired (.ne. t).
c itol   = 1 or 2 according as atol (below) is a scalar or array.
c rtol   = relative tolerance parameter (scalar).
c atol   = absolute tolerance parameter (scalar or array).
c          the estimated local error in y(i) will be controlled so as
c          to be roughly less (in magnitude) than
c             ewt(i) = rtol*abs(y(i)) + atol     if itol = 1, or
c             ewt(i) = rtol*abs(y(i)) + atol(i)  if itol = 2.
c          thus the local error test passes if, in each component,
c          either the absolute error is less than atol (or atol(i)),
c          or the relative error is less than rtol.
c          use rtol = 0.0 for pure absolute error control, and
c          use atol = 0.0 (or atol(i) = 0.0) for pure relative error
c          control.  caution.. actual (global) errors may exceed these
c          local tolerances, so choose them conservatively.
c itask  = 1 for normal computation of output values of y at t = tout.
c istate = integer flag (input and output).  set istate = 1.
c iopt   = 0 to indicate no optional inputs used.
c rwork  = real work array of length at least..
c             20 + 16*neq                for mf = 10,
c             20 + (2 + 1./lenrat)*nnz + (11 + 9./lenrat)*neq
c                                        for mf = 121 or 222,
c          where..
c          nnz    = the number of nonzero elements in the sparse
c                   jacobian (if this is unknown, use an estimate), and
```

```
c          lenrat = the real to integer wordlength ratio (usually 1 in
c                   single precision and 2 in double precision).
c          in any case, the required size of rwork cannot generally
c          be predicted in advance if mf = 121 or 222, and the value
c          above is a rough estimate of a crude lower bound.  some
c          experimentation with this size may be necessary.
c          (when known, the correct required length is an optional
c          output, available in iwork(17).)
c lrw    = declared length of rwork (in user-s dimension).
c iwork  = integer work array of length at least 30.
c liw    = declared length of iwork (in user-s dimension).
c jac    = name of subroutine for jacobian matrix (mf = 121).
c          if used, this name must be declared external in calling
c          program.  if not used, pass a dummy name.
c mf     = method flag.  standard values are..
c          10  for nonstiff (adams) method, no jacobian used.
c          121 for stiff (bdf) method, user-supplied sparse jacobian.
c          222 for stiff method, internally generated sparse jacobian.
c note that the main program must declare arrays y, rwork, iwork,
c and possibly atol.
c
c e. the output from the first call (or any call) is..
c      y = array of computed values of y(t) vector.
c      t = corresponding value of independent variable (normally tout).
c istate = 2  if lsodes was successful, negative otherwise.
c          -1 means excess work done on this call (perhaps wrong mf).
c          -2 means excess accuracy requested (tolerances too small).
c          -3 means illegal input detected (see printed message).
c          -4 means repeated error test failures (check all inputs).
c          -5 means repeated convergence failures (perhaps bad jacobian
c             supplied or wrong choice of mf or tolerances).
c          -6 means error weight became zero during problem. (solution
c             component i vanished, and atol or atol(i) = 0.)
c          -7 means a fatal error return flag came from the sparse
c             solver cdrv by way of prjs or slss.  should never happen.
c          a return with istate = -1, -4, or -5 may result from using
c          an inappropriate sparsity structure, one that is quite
c          different from the initial structure.  consider calling
c          lsodes again with istate = 3 to force the structure to be
c          reevaluated.  see the full description of istate below.
c
c f. to continue the integration after a successful return, simply
c reset tout and call lsodes again.  no other parameters need be reset.
c
c-----------------------------------------------------------------------
c example problem.
c
c the following is a simple example problem, with the coding
c needed for its solution by lsodes.  the problem is from chemical
c kinetics, and consists of the following 12 rate equations..
c    dy1/dt  = -rk1*y1
c    dy2/dt  = rk1*y1 + rk11*rk14*y4 + rk19*rk14*y5
c                - rk3*y2*y3 - rk15*y2*y12 - rk2*y2
c    dy3/dt  = rk2*y2 - rk5*y3 - rk3*y2*y3 - rk7*y10*y3
c                + rk11*rk14*y4 + rk12*rk14*y6
c    dy4/dt  = rk3*y2*y3 - rk11*rk14*y4 - rk4*y4
c    dy5/dt  = rk15*y2*y12 - rk19*rk14*y5 - rk16*y5
c    dy6/dt  = rk7*y10*y3 - rk12*rk14*y6 - rk8*y6
c    dy7/dt  = rk17*y10*y12 - rk20*rk14*y7 - rk18*y7
```

```
c     dy8/dt  = rk9*y10 - rk13*rk14*y8 - rk10*y8
c     dy9/dt  = rk4*y4 + rk16*y5 + rk8*y6 + rk18*y7
c     dy10/dt = rk5*y3 + rk12*rk14*y6 + rk20*rk14*y7
c                 + rk13*rk14*y8 - rk7*y10*y3 - rk17*y10*y12
c                 - rk6*y10 - rk9*y10
c     dy11/dt = rk10*y8
c     dy12/dt = rk6*y10 + rk19*rk14*y5 + rk20*rk14*y7
c                 - rk15*y2*y12 - rk17*y10*y12
c
c with rk1 = rk5 = 0.1,  rk4 = rk8 = rk16 = rk18 = 2.5,
c      rk10 = 5.0,  rk2 = rk6 = 10.0,  rk14 = 30.0,
c      rk3 = rk7 = rk9 = rk11 = rk12 = rk13 = rk19 = rk20 = 50.0,
c      rk15 = rk17 = 100.0.
c
c the t interval is from 0 to 1000, and the initial conditions
c are y1 = 1, y2 = y3 = ... = y12 = 0.  the problem is stiff.
c
c the following coding solves this problem with lsodes, using mf = 121
c and printing results at t = .1, 1., 10., 100., 1000.  it uses
c itol = 1 and mixed relative/absolute tolerance controls.
c during the run and at the end, statistical quantities of interest
c are printed (see optional outputs in the full description below).
c
c     external fex, jex
c     double precision atol, rtol, rwork, t, tout, y
c     dimension y(12), rwork(500), iwork(30)
c     data lrw/500/, liw/30/
c     neq = 12
c     do 10 i = 1,neq
c 10    y(i) = 0.0d0
c     y(1) = 1.0d0
c     t = 0.0d0
c     tout = 0.1d0
c     itol = 1
c     rtol = 1.0d-4
c     atol = 1.0d-6
c     itask = 1
c     istate = 1
c     iopt = 0
c     mf = 121
c     do 40 iout = 1,5
c       call lsodes (fex, neq, y, t, tout, itol, rtol, atol,
c    1     itask, istate, iopt, rwork, lrw, iwork, liw, jex, mf)
c       write(6,30)t,iwork(11),rwork(11),(y(i),i=1,neq)
c 30    format(//7h at t =,e11.3,4x,
c    1    12h no. steps =,i5,4x,12h last step =,e11.3/
c    2    13h  y array =  ,4e14.5/13x,4e14.5/13x,4e14.5)
c       if (istate .lt. 0) go to 80
c       tout = tout*10.0d0
c 40    continue
c     lenrw = iwork(17)
c     leniw = iwork(18)
c     nst = iwork(11)
c     nfe = iwork(12)
c     nje = iwork(13)
c     nlu = iwork(21)
c     nnz = iwork(19)
c     nnzlu = iwork(25) + iwork(26) + neq
c     write (6,70) lenrw,leniw,nst,nfe,nje,nlu,nnz,nnzlu
```

```
c 70  format(//22h required rwork size =,i4,15h   iwork size =,i4/
c    1   12h no. steps =,i4,12h   no. f-s =,i4,12h   no. j-s =,i4,
c    2   13h   no. lu-s =,i4/23h no. of nonzeros in j =,i5,
c    3   26h   no. of nonzeros in lu =,i5)
c     stop
c 80  write(6,90)istate
c 90  format(///22h error halt.. istate =,i3)
c     stop
c     end
c
c     subroutine fex (neq, t, y, ydot)
c     double precision t, y, ydot
c     double precision rk1, rk2, rk3, rk4, rk5, rk6, rk7, rk8, rk9,
c    1   rk10, rk11, rk12, rk13, rk14, rk15, rk16, rk17
c     dimension y(12), ydot(12)
c     data rk1/0.1d0/, rk2/10.0d0/, rk3/50.0d0/, rk4/2.5d0/, rk5/0.1d0/,
c    1   rk6/10.0d0/, rk7/50.0d0/, rk8/2.5d0/, rk9/50.0d0/, rk10/5.0d0/,
c    2   rk11/50.0d0/, rk12/50.0d0/, rk13/50.0d0/, rk14/30.0d0/,
c    3   rk15/100.0d0/, rk16/2.5d0/, rk17/100.0d0/, rk18/2.5d0/,
c    4   rk19/50.0d0/, rk20/50.0d0/
c     ydot(1)  = -rk1*y(1)
c     ydot(2)  = rk1*y(1) + rk11*rk14*y(4) + rk19*rk14*y(5)
c    1           - rk3*y(2)*y(3) - rk15*y(2)*y(12) - rk2*y(2)
c     ydot(3)  = rk2*y(2) - rk5*y(3) - rk3*y(2)*y(3) - rk7*y(10)*y(3)
c    1           + rk11*rk14*y(4) + rk12*rk14*y(6)
c     ydot(4)  = rk3*y(2)*y(3) - rk11*rk14*y(4) - rk4*y(4)
c     ydot(5)  = rk15*y(2)*y(12) - rk19*rk14*y(5) - rk16*y(5)
c     ydot(6)  = rk7*y(10)*y(3) - rk12*rk14*y(6) - rk8*y(6)
c     ydot(7)  = rk17*y(10)*y(12) - rk20*rk14*y(7) - rk18*y(7)
c     ydot(8)  = rk9*y(10) - rk13*rk14*y(8) - rk10*y(8)
c     ydot(9)  = rk4*y(4) + rk16*y(5) + rk8*y(6) + rk18*y(7)
c     ydot(10) = rk5*y(3) + rk12*rk14*y(6) + rk20*rk14*y(7)
c    1           + rk13*rk14*y(8) - rk7*y(10)*y(3) - rk17*y(10)*y(12)
c    2           - rk6*y(10) - rk9*y(10)
c     ydot(11) = rk10*y(8)
c     ydot(12) = rk6*y(10) + rk19*rk14*y(5) + rk20*rk14*y(7)
c    1           - rk15*y(2)*y(12) - rk17*y(10)*y(12)
c     return
c     end
c
c     subroutine jex (neq, t, y, j, ia, ja, pdj)
c     double precision t, y, pdj
c     double precision rk1, rk2, rk3, rk4, rk5, rk6, rk7, rk8, rk9,
c    1   rk10, rk11, rk12, rk13, rk14, rk15, rk16, rk17
c     dimension y(1), ia(1), ja(1), pdj(1)
c     data rk1/0.1d0/, rk2/10.0d0/, rk3/50.0d0/, rk4/2.5d0/, rk5/0.1d0/,
c    1   rk6/10.0d0/, rk7/50.0d0/, rk8/2.5d0/, rk9/50.0d0/, rk10/5.0d0/,
c    2   rk11/50.0d0/, rk12/50.0d0/, rk13/50.0d0/, rk14/30.0d0/,
c    3   rk15/100.0d0/, rk16/2.5d0/, rk17/100.0d0/, rk18/2.5d0/,
c    4   rk19/50.0d0/, rk20/50.0d0/
c     go to (1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12), j
c 1   pdj(1) = -rk1
c     pdj(2) = rk1
c     return
c 2   pdj(2) = -rk3*y(3) - rk15*y(12) - rk2
c     pdj(3) = rk2 - rk3*y(3)
c     pdj(4) = rk3*y(3)
c     pdj(5) = rk15*y(12)
c     pdj(12) = -rk15*y(12)
```

```
c       return
c 3     pdj(2) = -rk3*y(2)
c       pdj(3) = -rk5 - rk3*y(2) - rk7*y(10)
c       pdj(4) = rk3*y(2)
c       pdj(6) = rk7*y(10)
c       pdj(10) = rk5 - rk7*y(10)
c       return
c 4     pdj(2) = rk11*rk14
c       pdj(3) = rk11*rk14
c       pdj(4) = -rk11*rk14 - rk4
c       pdj(9) = rk4
c       return
c 5     pdj(2) = rk19*rk14
c       pdj(5) = -rk19*rk14 - rk16
c       pdj(9) = rk16
c       pdj(12) = rk19*rk14
c       return
c 6     pdj(3) = rk12*rk14
c       pdj(6) = -rk12*rk14 - rk8
c       pdj(9) = rk8
c       pdj(10) = rk12*rk14
c       return
c 7     pdj(7) = -rk20*rk14 - rk18
c       pdj(9) = rk18
c       pdj(10) = rk20*rk14
c       pdj(12) = rk20*rk14
c       return
c 8     pdj(8) = -rk13*rk14 - rk10
c       pdj(10) = rk13*rk14
c       pdj(11) = rk10
c 9     return
c 10    pdj(3) = -rk7*y(3)
c       pdj(6) = rk7*y(3)
c       pdj(7) = rk17*y(12)
c       pdj(8) = rk9
c       pdj(10) = -rk7*y(3) - rk17*y(12) - rk6 - rk9
c       pdj(12) = rk6 - rk17*y(12)
c 11    return
c 12    pdj(2) = -rk15*y(2)
c       pdj(5) = rk15*y(2)
c       pdj(7) = rk17*y(10)
c       pdj(10) = -rk17*y(10)
c       pdj(12) = -rk15*y(2) - rk17*y(10)
c       return
c       end
c
c the output of this program (on a cray-1 in single precision)
c is as follows..
c
c
c at t =  1.000e-01     no. steps =   12     last step =  1.515e-02
c  y array =     9.90050e-01   6.28228e-03   3.65313e-03   7.51934e-07
c                1.12167e-09   1.18458e-09   1.77291e-12   3.26476e-07
c                5.46720e-08   9.99500e-06   4.48483e-08   2.76398e-06
c
c
c at t =  1.000e+00     no. steps =   33     last step =  7.880e-02
c  y array =     9.04837e-01   9.13105e-03   8.20622e-02   2.49177e-05
c                1.85055e-06   1.96797e-06   1.46157e-07   2.39557e-05
```

```
c                 3.26306e-05   7.21621e-04   5.06433e-05   3.05010e-03
c
c
c at t =  1.000e+01     no. steps =   48     last step =  1.239e+00
c  y array =     3.67876e-01   3.68958e-03   3.65133e-01   4.48325e-05
c                6.10798e-05   4.33148e-05   5.90211e-05   1.18449e-04
c                3.15235e-03   3.56531e-03   4.15520e-03   2.48741e-01
c
c
c at t =  1.000e+02     no. steps =   91     last step =  3.764e+00
c  y array =     4.44981e-05   4.42666e-07   4.47273e-04  -3.53257e-11
c                2.81577e-08  -9.67741e-11   2.77615e-07   1.45322e-07
c                1.56230e-02   4.37394e-06   1.60104e-02   9.52246e-01
c
c
c at t =  1.000e+03     no. steps =  111     last step =  4.156e+02
c  y array =    -2.65492e-13   2.60539e-14  -8.59563e-12   6.29355e-14
c               -1.78066e-13   5.71471e-13  -1.47561e-12   4.58078e-15
c                1.56314e-02   1.37878e-13   1.60184e-02   9.52719e-01
c
c
c required rwork size = 442   iwork size =  30
c no. steps = 111   no. f-s = 142   no. j-s =   2   no. lu-s =  20
c no. of nonzeros in j =   44   no. of nonzeros in lu =   50
c-----------------------------------------------------------------------
c full description of user interface to lsodes.
c
c the user interface to lsodes consists of the following parts.
c
c i.   the call sequence to subroutine lsodes, which is a driver
c      routine for the solver.  this includes descriptions of both
c      the call sequence arguments and of user-supplied routines.
c      following these descriptions is a description of
c      optional inputs available through the call sequence, and then
c      a description of optional outputs (in the work arrays).
c
c ii.  descriptions of other routines in the lsodes package that may be
c      (optionally) called by the user.  these provide the ability to
c      alter error message handling, save and restore the internal
c      common, and obtain specified derivatives of the solution y(t).
c
c iii. descriptions of common blocks to be declared in overlay
c      or similar environments, or to be saved when doing an interrupt
c      of the    oblem and continued solution later.
c
c iv.  description of two routines in the lsodes package, either of
c      which the user may replace with his own version, if desired.
c      these relate to the measurement of errors.
c
c-----------------------------------------------------------------------
c part i.  call sequence.
c
c the call sequence parameters used for input only are
c     f, neq, tout, itol, rtol, atol, itask, iopt, lrw, liw, jac, mf,
c and those used for both input and output are
c     y, t, istate.
c the work arrays rwork and iwork are also used for conditional and
c optional inputs and optional outputs.  (the term output here refers
c to the return from subroutine lsodes to the user-s calling program.)
```

```
c
c the legality of input parameters will be thoroughly checked on the
c initial call for the problem, but not checked thereafter unless a
c change in input parameters is flagged by istate = 3 on input.
c
c the descriptions of the call arguments are as follows.
c
c f      = the name of the user-supplied subroutine defining the
c          ode system.  the system must be put in the first-order
c          form dy/dt = f(t,y), where f is a vector-valued function
c          of the scalar t and the vector y.  subroutine f is to
c          compute the function f.  it is to have the form
c               subroutine f (neq, t, y, ydot)
c               dimension y(1), ydot(1)
c          where neq, t, and y are input, and the array ydot = f(t,y)
c          is output.  y and ydot are arrays of length neq.
c          (in the dimension statement above, 1 is a dummy
c          dimension.. it can be replaced by any value.)
c          subroutine f should not alter y(1),...,y(neq).
c          f must be declared external in the calling program.
c
c          subroutine f may access user-defined quantities in
c          neq(2),... and/or in y(neq(1)+1),... if neq is an array
c          (dimensioned in f) and/or y has length exceeding neq(1).
c          see the descriptions of neq and y below.
c
c          if quantities computed in the f routine are needed
c          externally to lsodes, an extra call to f should be made
c          for this purpose, for consistent and accurate results.
c          if only the derivative dy/dt is needed, use intdy instead.
c
c neq    = the size of the ode system (number of first order
c          ordinary differential equations).  used only for input.
c          neq may be decreased, but not increased, during the problem.
c          if neq is decreased (with istate = 3 on input), the
c          remaining components of y should be left undisturbed, if
c          these are to be accessed in f and/or jac.
c
c          normally, neq is a scalar, and it is generally referred to
c          as a scalar in this user interface description.  however,
c          neq may be an array, with neq(1) set to the system size.
c          (the lsodes package accesses only neq(1).)  in either case,
c          this parameter is passed as the neq argument in all calls
c          to f and jac.  hence, if it is an array, locations
c          neq(2),... may be used to store other integer data and pass
c          it to f and/or jac.  subroutines f and/or jac must include
c          neq in a dimension statement in that case.
c
c y      = a real array for the vector of dependent variables, of
c          length neq or more.  used for both input and output on the
c          first call (istate = 1), and only for output on other calls.
c          on the first call, y must contain the vector of initial
c          values.  on output, y contains the computed solution vector,
c          evaluated at t.  if desired, the y array may be used
c          for other purposes between calls to the solver.
c
c          this array is passed as the y argument in all calls to
c          f and jac.  hence its length may exceed neq, and locations
c          y(neq+1),... may be used to store other real data and
```

```
c          pass it to f and/or jac.  (the lsodes package accesses only
c          y(1),...,y(neq).)
c
c t      = the independent variable.  on input, t is used only on the
c          first call, as the initial point of the integration.
c          on output, after each call, t is the value at which a
c          computed solution y is evaluated (usually the same as tout).
c          on an error return, t is the farthest point reached.
c
c tout   = the next value of t at which a computed solution is desired.
c          used only for input.
c
c          when starting the problem (istate = 1), tout may be equal
c          to t for one call, then should .ne. t for the next call.
c          for the initial t, an input value of tout .ne. t is used
c          in order to determine the direction of the integration
c          (i.e. the algebraic sign of the step sizes) and the rough
c          scale of the problem.  integration in either direction
c          (forward or backward in t) is permitted.
c
c          if itask = 2 or 5 (one-step modes), tout is ignored after
c          the first call (i.e. the first call with tout .ne. t).
c          otherwise, tout is required on every call.
c
c          if itask = 1, 3, or 4, the values of tout need not be
c          monotone, but a value of tout which backs up is limited
c          to the current internal t interval, whose endpoints are
c          tcur - hu and tcur (see optional outputs, below, for
c          tcur and hu).
c
c itol   = an indicator for the type of error control.  see
c          description below under atol.  used only for input.
c
c rtol   = a relative error tolerance parameter, either a scalar or
c          an array of length neq.  see description below under atol.
c          input only.
c
c atol   = an absolute error tolerance parameter, either a scalar or
c          an array of length neq.  input only.
c
c             the input parameters itol, rtol, and atol determine
c          the error control performed by the solver.  the solver will
c          control the vector e = (e(i)) of estimated local errors
c          in y, according to an inequality of the form
c                      rms-norm of ( e(i)/ewt(i) )   .le.   1,
c          where       ewt(i) = rtol(i)*abs(y(i)) + atol(i),
c          and the rms-norm (root-mean-square norm) here is
c          rms-norm(v) = sqrt(sum v(i)**2 / neq).  here ewt = (ewt(i))
c          is a vector of weights which must always be positive, and
c          the values of rtol and atol should all be non-negative.
c          the following table gives the types (scalar/array) of
c          rtol and atol, and the corresponding form of ewt(i).
c
c             itol    rtol       atol          ewt(i)
c              1     scalar     scalar     rtol*abs(y(i)) + atol
c              2     scalar     array      rtol*abs(y(i)) + atol(i)
c              3     array      scalar     rtol(i)*abs(y(i)) + atol
c              4     array      array      rtol(i)*abs(y(i)) + atol(i)
c
```

```
c          when either of these parameters is a scalar, it need not
c          be dimensioned in the user-s calling program.
c
c          if none of the above choices (with itol, rtol, and atol
c          fixed throughout the problem) is suitable, more general
c          error controls can be obtained by substituting
c          user-supplied routines for the setting of ewt and/or for
c          the norm calculation.  see part iv below.
c
c          if global errors are to be estimated by making a repeated
c          run on the same problem with smaller tolerances, then all
c          components of rtol and atol (i.e. of ewt) should be scaled
c          down uniformly.
c
c itask  = an index specifying the task to be performed.
c          input only.  itask has the following values and meanings.
c          1  means normal computation of output values of y(t) at
c             t = tout (by overshooting and interpolating).
c          2  means take one step only and return.
c          3  means stop at the first internal mesh point at or
c             beyond t = tout and return.
c          4  means normal computation of output values of y(t) at
c             t = tout but without overshooting t = tcrit.
c             tcrit must be input as rwork(1).  tcrit may be equal to
c             or beyond tout, but not behind it in the direction of
c             integration.  this option is useful if the problem
c             has a singularity at or beyond t = tcrit.
c          5  means take one step, without passing tcrit, and return.
c             tcrit must be input as rwork(1).
c
c          note..  if itask = 4 or 5 and the solver reaches tcrit
c          (within roundoff), it will return t = tcrit (exactly) to
c          indicate this (unless itask = 4 and tout comes before tcrit,
c          in which case answers at t = tout are returned first).
c
c istate = an index used for input and output to specify the
c          the state of the calculation.
c
c          on input, the values of istate are as follows.
c          1  means this is the first call for the problem
c             (initializations will be done).  see note below.
c          2  means this is not the first call, and the calculation
c             is to continue normally, with no change in any input
c             parameters except possibly tout and itask.
c             (if itol, rtol, and/or atol are changed between calls
c             with istate = 2, the new values will be used but not
c             tested for legality.)
c          3  means this is not the first call, and the
c             calculation is to continue normally, but with
c             a change in input parameters other than
c             tout and itask.  changes are allowed in
c             neq, itol, rtol, atol, iopt, lrw, liw, mf,
c             the conditional inputs ia and ja,
c             and any of the optional inputs except h0.
c             in particular, if miter = 1 or 2, a call with istate = 3
c             will cause the sparsity structure of the problem to be
c             recomputed (or reread from ia and ja if moss = 0).
c          note..  a preliminary call with tout = t is not counted
c          as a first call here, as no initialization or checking of
```

```
c          input is done.  (such a call is sometimes useful for the
c          purpose of outputting the initial conditions.)
c          thus the first call for which tout .ne. t requires
c          istate = 1 on input.
c
c          on output, istate has the following values and meanings.
c           1  means nothing was done, as tout was equal to t with
c              istate = 1 on input.  (however, an internal counter was
c              set to detect and prevent repeated calls of this type.)
c           2  means the integration was performed successfully.
c          -1  means an excessive amount of work (more than mxstep
c              steps) was done on this call, before completing the
c              requested task, but the integration was otherwise
c              successful as far as t.  (mxstep is an optional input
c              and is normally 500.)  to continue, the user may
c              simply reset istate to a value .gt. 1 and call again
c              (the excess work step counter will be reset to 0).
c              in addition, the user may increase mxstep to avoid
c              this error return (see below on optional inputs).
c          -2  means too much accuracy was requested for the precision
c              of the machine being used.  this was detected before
c              completing the requested task, but the integration
c              was successful as far as t.  to continue, the tolerance
c              parameters must be reset, and istate must be set
c              to 3.  the optional output tolsf may be used for this
c              purpose.  (note.. if this condition is detected before
c              taking any steps, then an illegal input return
c              (istate = -3) occurs instead.)
c          -3  means illegal input was detected, before taking any
c              integration steps.  see written message for details.
c              note..  if the solver detects an infinite loop of calls
c              to the solver with illegal input, it will cause
c              the run to stop.
c          -4  means there were repeated error test failures on
c              one attempted step, before completing the requested
c              task, but the integration was successful as far as t.
c              the problem may have a singularity, or the input
c              may be inappropriate.
c          -5  means there were repeated convergence test failures on
c              one attempted step, before completing the requested
c              task, but the integration was successful as far as t.
c              this may be caused by an inaccurate jacobian matrix,
c              if one is being used.
c          -6  means ewt(i) became zero for some i during the
c              integration.  pure relative error control (atol(i)=0.0)
c              was requested on a variable which has now vanished.
c              the integration was successful as far as t.
c          -7  means a fatal error return flag came from the sparse
c              solver cdrv by way of prjs or slss (numerical
c              factorization or backsolve).  this should never happen.
c              the integration was successful as far as t.
c
c          note.. an error return with istate = -1, -4, or -5 and with
c          miter = 1 or 2 may mean that the sparsity structure of the
c          problem has changed significantly since it was last
c          determined (or input).  in that case, one can attempt to
c          complete the integration by setting istate = 3 on the next
c          call, so that a new structure determination is done.
c
```

```
c          note..  since the normal output value of istate is 2,
c          it does not need to be reset for normal continuation.
c          also, since a negative input value of istate will be
c          regarded as illegal, a negative output value requires the
c          user to change it, and possibly other inputs, before
c          calling the solver again.
c
c iopt   = an integer flag to specify whether or not any optional
c          inputs are being used on this call.  input only.
c          the optional inputs are listed separately below.
c          iopt = 0 means no optional inputs are being used.
c                   default values will be used in all cases.
c          iopt = 1 means one or more optional inputs are being used.
c
c rwork  = a work array used for a mixture of real (double precision)
c          and integer work space.
c          the length of rwork (in real words) must be at least
c             20 + nyh*(maxord + 1) + 3*neq + lwm    where
c          nyh    = the initial value of neq,
c          maxord = 12 (if meth = 1) or 5 (if meth = 2) (unless a
c                   smaller value is given as an optional input),
c          lwm = 0                                     if miter = 0,
c          lwm = 2*nnz + 2*neq + (nnz+9*neq)/lenrat    if miter = 1,
c          lwm = 2*nnz + 2*neq + (nnz+10*neq)/lenrat   if miter = 2,
c          lwm = neq + 2                               if miter = 3.
c          in the above formulas,
c          nnz    = number of nonzero elements in the jacobian matrix.
c          lenrat = the real to integer wordlength ratio (usually 1 in
c                   single precision and 2 in double precision).
c          (see the mf description for meth and miter.)
c          thus if maxord has its default value and neq is constant,
c          the minimum length of rwork is..
c             20 + 16*neq        for mf = 10,
c             20 + 16*neq + lwm  for mf = 11, 111, 211, 12, 112, 212,
c             22 + 17*neq        for mf = 13,
c             20 +  9*neq        for mf = 20,
c             20 +  9*neq + lwm  for mf = 21, 121, 221, 22, 122, 222,
c             22 + 10*neq        for mf = 23.
c          if miter = 1 or 2, the above formula for lwm is only a
c          crude lower bound.  the required length of rwork cannot
c          be readily predicted in general, as it depends on the
c          sparsity structure of the problem.  some experimentation
c          may be necessary.
c
c          the first 20 words of rwork are reserved for conditional
c          and optional inputs and optional outputs.
c
c          the following word in rwork is a conditional input..
c            rwork(1) = tcrit = critical value of t which the solver
c                       is not to overshoot.  required if itask is
c                       4 or 5, and ignored otherwise.  (see itask.)
c
c lrw    = the length of the array rwork, as declared by the user.
c          (this will be checked by the solver.)
c
c iwork  = an integer work array.  the length of iwork must be at least
c             31 + neq + nnz   if moss = 0 and miter = 1 or 2, or
c             30               otherwise.
c          (nnz is the number of nonzero elements in df/dy.)
```

```
c
c          in lsodes, iwork is used only for conditional and
c          optional inputs and optional outputs.
c
c          the following two blocks of words in iwork are conditional
c          inputs, required if moss = 0 and miter = 1 or 2, but not
c          otherwise (see the description of mf for moss).
c            iwork(30+j) = ia(j)      (j=1,...,neq+1)
c            iwork(31+neq+k) = ja(k) (k=1,...,nnz)
c          the two arrays ia and ja describe the sparsity structure
c          to be assumed for the jacobian matrix.  ja contains the row
c          indices where nonzero elements occur, reading in columnwise
c          order, and ia contains the starting locations in ja of the
c          descriptions of columns 1,...,neq, in that order, with
c          ia(1) = 1.  thus, for each column index j = 1,...,neq, the
c          values of the row index i in column j where a nonzero
c          element may occur are given by
c            i = ja(k),  where   ia(j) .le. k .lt. ia(j+1).
c          if nnz is the total number of nonzero locations assumed,
c          then the length of the ja array is nnz, and ia(neq+1) must
c          be nnz + 1.  duplicate entries are not allowed.
c
c liw    = the length of the array iwork, as declared by the user.
c          (this will be checked by the solver.)
c
c note..  the work arrays must not be altered between calls to lsodes
c for the same problem, except possibly for the conditional and
c optional inputs, and except for the last 3*neq words of rwork.
c the latter space is used for internal scratch space, and so is
c available for use by the user outside lsodes between calls, if
c desired (but not for use by f or jac).
c
c jac    = name of user-supplied routine (miter = 1 or moss = 1) to
c          compute the jacobian matrix, df/dy, as a function of
c          the scalar t and the vector y.  it is to have the form
c               subroutine jac (neq, t, y, j, ian, jan, pdj)
c               dimension y(1), ian(1), jan(1), pdj(1)
c          where neq, t, y, j, ian, and jan are input, and the array
c          pdj, of length neq, is to be loaded with column j
c          of the jacobian on output.  thus df(i)/dy(j) is to be
c          loaded into pdj(i) for all relevant values of i.
c          here t and y have the same meaning as in subroutine f,
c          and j is a column index (1 to neq).  ian and jan are
c          undefined in calls to jac for structure determination
c          (moss = 1).  otherwise, ian and jan are structure
c          descriptors, as defined under optional outputs below, and
c          so can be used to determine the relevant row indices i, if
c          desired.  (in the dimension statement above, 1 is a
c          dummy dimension.. it can be replaced by any value.)
c               jac need not provide df/dy exactly.  a crude
c          approximation (possibly with greater sparsity) will do.
c               in any case, pdj is preset to zero by the solver,
c          so that only the nonzero elements need be loaded by jac.
c          calls to jac are made with j = 1,...,neq, in that order, and
c          each such set of calls is preceded by a call to f with the
c          same arguments neq, t, and y.  thus to gain some efficiency,
c          intermediate quantities shared by both calculations may be
c          saved in a user common block by f and not recomputed by jac,
c          if desired.  jac must not alter its input arguments.
```

```
c          jac must be declared external in the calling program.
c               subroutine jac may access user-defined quantities in
c          neq(2),... and y(neq(1)+1),... if neq is an array
c          (dimensioned in jac) and y has length exceeding neq(1).
c          see the descriptions of neq and y above.
c
c mf     = the method flag.  used only for input.
c          mf has three decimal digits-- moss, meth, miter--
c             mf = 100*moss + 10*meth + miter.
c          moss indicates the method to be used to obtain the sparsity
c          structure of the jacobian matrix if miter = 1 or 2..
c            moss = 0 means the user has supplied ia and ja
c                     (see descriptions under iwork above).
c            moss = 1 means the user has supplied jac (see below)
c                     and the structure will be obtained from neq
c                     initial calls to jac.
c            moss = 2 means the structure will be obtained from neq+1
c                     initial calls to f.
c          meth indicates the basic linear multistep method..
c            meth = 1 means the implicit adams method.
c            meth = 2 means the method based on backward
c                     differentiation formulas (bdf-s).
c          miter indicates the corrector iteration method..
c            miter = 0 means functional iteration (no jacobian matrix
c                      is involved).
c            miter = 1 means chord iteration with a user-supplied
c                      sparse jacobian, given by subroutine jac.
c            miter = 2 means chord iteration with an internally
c                      generated (difference quotient) sparse jacobian
c                      (using ngp extra calls to f per df/dy value,
c                      where ngp is an optional output described below.)
c            miter = 3 means chord iteration with an internally
c                      generated diagonal jacobian approximation.
c                      (using 1 extra call to f per df/dy evaluation).
c          if miter = 1 or moss = 1, the user must supply a subroutine
c          jac (the name is arbitrary) as described above under jac.
c          otherwise, a dummy argument can be used.
c
c          the standard choices for mf are..
c            mf = 10  for a nonstiff problem,
c            mf = 21 or 22 for a stiff problem with ia/ja supplied
c                     (21 if jac is supplied, 22 if not),
c            mf = 121 for a stiff problem with jac supplied,
c                     but not ia/ja,
c            mf = 222 for a stiff problem with neither ia/ja nor
c                     jac supplied.
c          the sparseness structure can be changed during the
c          problem by making a call to lsodes with istate = 3.
c-----------------------------------------------------------------------
c optional inputs.
c
c the following is a list of the optional inputs provided for in the
c call sequence.  (see also part ii.)  for each such input variable,
c this table lists its name as used in this documentation, its
c location in the call sequence, its meaning, and the default value.
c the use of any of these inputs requires iopt = 1, and in that
c case all of these inputs are examined.  a value of zero for any
c of these optional inputs will cause the default value to be used.
c thus to use a subset of the optional inputs, simply preload
```

```
c locations 5 to 10 in rwork and iwork to 0.0 and 0 respectively, and
c then set those of interest to nonzero values.
c
c name     location       meaning and default value
c
c h0       rwork(5)   the step size to be attempted on the first step.
c                     the default value is determined by the solver.
c
c hmax     rwork(6)   the maximum absolute step size allowed.
c                     the default value is infinite.
c
c hmin     rwork(7)   the minimum absolute step size allowed.
c                     the default value is 0.  (this lower bound is not
c                     enforced on the final step before reaching tcrit
c                     when itask = 4 or 5.)
c
c seth     rwork(8)   the element threshhold for sparsity determination
c                     when moss = 1 or 2.  if the absolute value of
c                     an estimated jacobian element is .le. seth, it
c                     will be assumed to be absent in the structure.
c                     the default value of seth is 0.
c
c maxord   iwork(5)   the maximum order to be allowed.  the default
c                     value is 12 if meth = 1, and 5 if meth = 2.
c                     if maxord exceeds the default value, it will
c                     be reduced to the default value.
c                     if maxord is changed during the problem, it may
c                     cause the current order to be reduced.
c
c mxstep   iwork(6)   maximum number of (internally defined) steps
c                     allowed during one call to the solver.
c                     the default value is 500.
c
c mxhnil   iwork(7)   maximum number of messages printed (per problem)
c                     warning that t + h = t on a step (h = step size).
c                     this must be positive to result in a non-default
c                     value.  the default value is 10.
c-----------------------------------------------------------------------
c optional outputs.
c
c as optional additional output from lsodes, the variables listed
c below are quantities related to the performance of lsodes
c which are available to the user.  these are communicated by way of
c the work arrays, but also have internal mnemonic names as shown.
c except where stated otherwise, all of these outputs are defined
c on any successful return from lsodes, and on any return with
c istate = -1, -2, -4, -5, or -6.  on an illegal input return
c (istate = -3), they will be unchanged from their existing values
c (if any), except possibly for tolsf, lenrw, and leniw.
c on any error return, outputs relevant to the error will be defined,
c as noted below.
c
c name     location       meaning
c
c hu       rwork(11) the step size in t last used (successfully).
c
c hcur     rwork(12) the step size to be attempted on the next step.
c
c tcur     rwork(13) the current value of the independent variable
```

```
c                    which the solver has actually reached, i.e. the
c                    current internal mesh point in t.  on output, tcur
c                    will always be at least as far as the argument
c                    t, but may be farther (if interpolation was done).
c
c tolsf   rwork(14) a tolerance scale factor, greater than 1.0,
c                    computed when a request for too much accuracy was
c                    detected (istate = -3 if detected at the start of
c                    the problem, istate = -2 otherwise).  if itol is
c                    left unaltered but rtol and atol are uniformly
c                    scaled up by a factor of tolsf for the next call,
c                    then the solver is deemed likely to succeed.
c                    (the user may also ignore tolsf and alter the
c                    tolerance parameters in any other way appropriate.)
c
c nst     iwork(11) the number of steps taken for the problem so far.
c
c nfe     iwork(12) the number of f evaluations for the problem so far,
c                    excluding those for structure determination
c                    (moss = 2).
c
c nje     iwork(13) the number of jacobian evaluations for the problem
c                    so far, excluding those for structure determination
c                    (moss = 1).
c
c nqu     iwork(14) the method order last used (successfully).
c
c nqcur   iwork(15) the order to be attempted on the next step.
c
c imxer   iwork(16) the index of the component of largest magnitude in
c                    the weighted local error vector ( e(i)/ewt(i) ),
c                    on an error return with istate = -4 or -5.
c
c lenrw   iwork(17) the length of rwork actually required.
c                    this is defined on normal returns and on an illegal
c                    input return for insufficient storage.
c
c leniw   iwork(18) the length of iwork actually required.
c                    this is defined on normal returns and on an illegal
c                    input return for insufficient storage.
c
c nnz     iwork(19) the number of nonzero elements in the jacobian
c                    matrix, including the diagonal (miter = 1 or 2).
c                    (this may differ from that given by ia(neq+1)-1
c                    if moss = 0, because of added diagonal entries.)
c
c ngp     iwork(20) the number of groups of column indices, used in
c                    difference quotient jacobian aproximations if
c                    miter = 2.  this is also the number of extra f
c                    evaluations needed for each jacobian evaluation.
c
c nlu     iwork(21) the number of sparse lu decompositions for the
c                    problem so far.
c
c lyh     iwork(22) the base address in rwork of the history array yh,
c                    described below in this list.
c
c ipian   iwork(23) the base address of the structure descriptor array
c                    ian, described below in this list.
```

```
c
c ipjan   iwork(24) the base address of the structure descriptor array
c                   jan, described below in this list.
c
c nzl     iwork(25) the number of nonzero elements in the strict lower
c                   triangle of the lu factorization used in the chord
c                   iteration (miter = 1 or 2).
c
c nzu     iwork(26) the number of nonzero elements in the strict upper
c                   triangle of the lu factorization used in the chord
c                   iteration (miter = 1 or 2).
c                   the total number of nonzeros in the factorization
c                   is therefore nzl + nzu + neq.
c
c the following four arrays are segments of the rwork array which
c may also be of interest to the user as optional outputs.
c for each array, the table below gives its internal name,
c its base address, and its description.
c for yh and acor, the base addresses are in rwork (a real array).
c the integer arrays ian and jan are to be obtained by declaring an
c integer array iwk and identifying iwk(1) with rwork(21), using either
c an equivalence statement or a subroutine call.  then the base
c addresses ipian (of ian) and ipjan (of jan) in iwk are to be obtained
c as optional outputs iwork(23) and iwork(24), respectively.
c thus ian(1) is iwk(ipian), etc.
c
c name    base address      description
c
c ian    ipian (in iwk)  structure descriptor array of size neq + 1.
c jan    ipjan (in iwk)  structure descriptor array of size nnz.
c         (see above)    ian and jan together describe the sparsity
c                        structure of the jacobian matrix, as used by
c                        lsodes when miter = 1 or 2.
c                        jan contains the row indices of the nonzero
c                        locations, reading in columnwise order, and
c                        ian contains the starting locations in jan of
c                        the descriptions of columns 1,...,neq, in
c                        that order, with ian(1) = 1.  thus for each
c                        j = 1,...,neq, the row indices i of the
c                        nonzero locations in column j are
c                        i = jan(k),  ian(j) .le. k .lt. ian(j+1).
c                        note that ian(neq+1) = nnz + 1.
c                        (if moss = 0, ian/jan may differ from the
c                        input ia/ja because of a different ordering
c                        in each column, and added diagonal entries.)
c
c yh      lyh            the nordsieck history array, of size nyh by
c          (optional     (nqcur + 1), where nyh is the initial value
c          output)       of neq.  for j = 0,1,...,nqcur, column j+1
c                        of yh contains hcur**j/factorial(j) times
c                        the j-th derivative of the interpolating
c                        polynomial currently representing the solution,
c                        evaluated at t = tcur.  the base address lyh
c                        is another optional output, listed above.
c
c acor     lenrw-neq+1   array of size neq used for the accumulated
c                        corrections on each step, scaled on output
c                        to represent the estimated local error in y
c                        on the last step.  this is the vector e in
```

```
c                        the description of the error control.  it is
c                        defined only on a successful return from
c                        lsodes.
c
c-----------------------------------------------------------------------
c part ii.  other routines callable.
c
c the following are optional calls which the user may make to
c gain additional capabilities in conjunction with lsodes.
c (the routines xsetun and xsetf are designed to conform to the
c slatec error handling package.)
c
c     form of call                  function
c   call xsetun(lun)          set the logical unit number, lun, for
c                             output of messages from lsodes, if
c                             the default is not desired.
c                             the default value of lun is 6.
c
c   call xsetf(mflag)         set a flag to control the printing of
c                             messages by lsodes.
c                             mflag = 0 means do not print. (danger..
c                             this risks losing valuable information.)
c                             mflag = 1 means print (the default).
c
c                             either of the above calls may be made at
c                             any time and will take effect immediately.
c
c   call srcms(rsav,isav,job) saves and restores the contents of
c                             the internal common blocks used by
c                             lsodes (see part iii below).
c                             rsav must be a real array of length 224
c                             or more, and isav must be an integer
c                             array of length 75 or more.
c                             job=1 means save common into rsav/isav.
c                             job=2 means restore common from rsav/isav.
c                                srcms is useful if one is
c                             interrupting a run and restarting
c                             later, or alternating between two or
c                             more problems solved with lsodes.
c
c   call intdy(,,,,,)         provide derivatives of y, of various
c        (see below)          orders, at a specified point t, if
c                             desired.  it may be called only after
c                             a successful return from lsodes.
c
c the detailed instructions for using intdy are as follows.
c the form of the call is..
c
c   lyh = iwork(22)
c   call intdy (t, k, rwork(lyh), nyh, dky, iflag)
c
c the input parameters are..
c
c t         = value of independent variable where answers are desired
c             (normally the same as the t last returned by lsodes).
c             for valid results, t must lie between tcur - hu and tcur.
c             (see optional outputs for tcur and hu.)
c k         = integer order of the derivative desired.  k must satisfy
c             0 .le. k .le. nqcur, where nqcur is the current order
```

```
c              (see optional outputs).  the capability corresponding
c              to k = 0, i.e. computing y(t), is already provided
c              by lsodes directly.  since nqcur .ge. 1, the first
c              derivative dy/dt is always available with intdy.
c lyh        = the base address of the history array yh, obtained
c              as an optional output as shown above.
c nyh        = column length of yh, equal to the initial value of neq.
c
c the output parameters are..
c
c dky        = a real array of length neq containing the computed value
c              of the k-th derivative of y(t).
c iflag      = integer flag, returned as 0 if k and t were legal,
c              -1 if k was illegal, and -2 if t was illegal.
c              on an error return, a message is also written.
c-----------------------------------------------------------------------
c part iii.  common blocks.
c
c if lsodes is to be used in an overlay situation, the user
c must declare, in the primary overlay, the variables in..
c   (1) the call sequence to lsodes,
c   (2) the three internal common blocks
c         /ls0001/  of length  257  (218 double precision words
c                        followed by 39 integer words),
c         /lss001/  of length  40    ( 6 double precision words
c                        followed by 34 integer words),
c         /eh0001/  of length  2 (integer words).
c
c if lsodes is used on a system in which the contents of internal
c common blocks are not preserved between calls, the user should
c declare the above three common blocks in his main program to insure
c that their contents are preserved.
c
c if the solution of a given problem by lsodes is to be interrupted
c and then later continued, such as when restarting an interrupted run
c or alternating between two or more problems, the user should save,
c following the return from the last lsodes call prior to the
c interruption, the contents of the call sequence variables and the
c internal common blocks, and later restore these values before the
c next lsodes call for that problem.  to save and restore the common
c blocks, use subroutine srcms (see part ii above).
c
c-----------------------------------------------------------------------
c part iv.  optionally replaceable solver routines.
c
c below are descriptions of two routines in the lsodes package which
c relate to the measurement of errors.  either routine can be
c replaced by a user-supplied version, if desired.  however, since such
c a replacement may have a major impact on performance, it should be
c done only when absolutely necessary, and only with great caution.
c (note.. the means by which the package version of a routine is
c superseded by the user-s version may be system-dependent.)
c
c (a) ewset.
c the following subroutine is called just before each internal
c integration step, and sets the array of error weights, ewt, as
c described under itol/rtol/atol above..
c     subroutine ewset (neq, itol, rtol, atol, ycur, ewt)
c where neq, itol, rtol, and atol are as in the lsodes call sequence,
```

```
c ycur contains the current dependent variable vector, and
c ewt is the array of weights set by ewset.
c
c if the user supplies this subroutine, it must return in ewt(i)
c (i = 1,...,neq) a positive quantity suitable for comparing errors
c in y(i) to.  the ewt array returned by ewset is passed to the
c vnorm routine (see below), and also used by lsodes in the computation
c of the optional output imxer, the diagonal jacobian approximation,
c and the increments for difference quotient jacobians.
c
c in the user-supplied version of ewset, it may be desirable to use
c the current values of derivatives of y.  derivatives up to order nq
c are available from the history array yh, described above under
c optional outputs.  in ewset, yh is identical to the ycur array,
c extended to nq + 1 columns with a column length of nyh and scale
c factors of h**j/factorial(j).  on the first call for the problem,
c given by nst = 0, nq is 1 and h is temporarily set to 1.0.
c the quantities nq, nyh, h, and nst can be obtained by including
c in ewset the statements..
c     double precision h, rls
c     common /ls0001/ rls(218),ils(39)
c     nq = ils(35)
c     nyh = ils(14)
c     nst = ils(36)
c     h = rls(212)
c thus, for example, the current value of dy/dt can be obtained as
c ycur(nyh+i)/h  (i=1,...,neq)  (and the division by h is
c unnecessary when nst = 0).
c
c (b) vnorm.
c the following is a real function routine which computes the weighted
c root-mean-square norm of a vector v..
c     d = vnorm (n, v, w)
c where..
c   n = the length of the vector,
c   v = real array of length n containing the vector,
c   w = real array of length n containing weights,
c   d = sqrt( (1/n) * sum(v(i)*w(i))**2 ).
c vnorm is called with n = neq and with w(i) = 1.0/ewt(i), where
c ewt is as set by subroutine ewset.
c
c if the user supplies this function, it should return a non-negative
c value of vnorm suitable for use in the error control in lsodes.
c none of the arguments should be altered by vnorm.
c for example, a user-supplied vnorm routine might..
c   -substitute a max-norm of (v(i)*w(i)) for the rms-norm, or
c   -ignore some components of v in the norm, with the effect of
c    suppressing the error control on those components of y.
c-----------------------------------------------------------------------
c-----------------------------------------------------------------------
c other routines in the lsodes package.
c
c in addition to subroutine lsodes, the lsodes package includes the
c following subroutines and function routines..
c  iprep    acts as an iterface between lsodes and prep, and also does
c           adjusting of work space pointers and work arrays.
c  prep     is called by iprep to compute sparsity and do sparse matrix
c           preprocessing if miter = 1 or 2.
c  jgroup   is called by prep to compute groups of jacobian column
```

```
c            indices for use when miter = 2.
c  adjlr     adjusts the length of required sparse matrix work space.
c            it is called by prep.
c  cntnzu    is called by prep and counts the nonzero elements in the
c            strict upper triangle of j + j-transpose, where j = df/dy.
c  intdy     computes an interpolated value of the y vector at t = tout.
c  stode     is the core integrator, which does one step of the
c            integration and the associated error control.
c  cfode     sets all method coefficients and test constants.
c  prjs      computes and preprocesses the jacobian matrix j = df/dy
c            and the newton iteration matrix p = i - h*10*j.
c  slss      manages solution of linear system in chord iteration.
c  ewset     sets the error weight vector ewt before each step.
c  vnorm     computes the weighted r.m.s. norm of a vector.
c  srcms     is a user-callable routine to save and restore
c            the contents of the internal common blocks.
c  odrv      constructs a reordering of the rows and columns of
c            a matrix by the minimum degree algorithm.  odrv is a
c            driver routine which calls subroutines md, mdi, mdm,
c            mdp, mdu, and sro.  see ref. 2 for details.  (the odrv
c            module has been modified since ref. 2, however.)
c  cdrv      performs reordering, symbolic factorization, numerical
c            factorization, or linear system solution operations,
c            depending on a path argument ipath.  cdrv is a
c            driver routine which calls subroutines nroc, nsfc,
c            nnfc, nnsc, and nntc.  see ref. 3 for details.
c            lsodes uses cdrv to solve linear systems in which the
c            coefficient matrix is  p = i - con*j, where i is the
c            identity, con is a scalar, and j is an approximation to
c            the jacobian df/dy.  because cdrv deals with rowwise
c            sparsity descriptions, cdrv works with p-transpose, not p.
c  dlmach    computes the unit roundoff in a machine-independent manner.
c  xerrwv, xsetun, and xsetf   handle the printing of all error
c            messages and warnings.  xerrwv is machine-dependent.
c note..  vnorm and dlmach are function routines.
c all the others are subroutines.
c
c the intrinsic and external routines used by lsodes are..
c dabs, dmax1, dmin1, dfloat, max0, min0, mod, dsign, dsqrt, and write.
c
c a block data subprogram is also included with the package,
c for loading some of the variables in internal common.
c
c-----------------------------------------------------------------------
c the following card is for optimized compilation on lll compilers.
cllll. optimize
c-----------------------------------------------------------------------
      external prjs, slss
      integer illin, init, lyh, lewt, lacor, lsavf, lwm, liwm,
     1   mxstep, mxhnil, nhnil, ntrep, nslast, nyh, iowns
      integer icf, ierpj, iersl, jcur, jstart, kflag, l, meth, miter,
     1   maxord, maxcor, msbp, mxncf, n, nq, nst, nfe, nje, nqu
      integer iplost, iesp, istatc, iys, iba, ibian, ibjan, ibjgp,
     1   ipian, ipjan, ipjgp, ipigp, ipr, ipc, ipic, ipisp, iprsp, ipa,
     2   lenyh, lenyhm, lenwk, lreq, lrat, lrest, lwmin, moss, msbj,
     3   nslj, ngp, nlu, nnz, nsp, nzl, nzu
      integer i, i1, i2, iflag, imax, imul, imxer, ipflag, ipgo, irem,
     1   j, kgo, lenrat, lenyht, leniw, lenrw, lf0, lia, lja,
     2   lrtem, lwtem, lyhd, lyhn, mfl, mord, mxhnl0, mxstp0, ncolm
```

```
      double precision rowns,
     1   ccmax, el0, h, hmin, hmxi, hu, rc, tn, uround
      double precision con0, conmin, ccmxj, psmall, rbig, seth
      double precision atoli, ayi, big, ewti, h0, hmax, hmx, rh, rtoli,
     1   tcrit, tdist, tnext, tol, tolsf, tp, size, sum, w0,
     2   d1mach, vnorm
      dimension mord(2)
      logical ihit
c-----------------------------------------------------------------------
c the following two internal common blocks contain
c (a) variables which are local to any subroutine but whose values must
c     be preserved between calls to the routine (own variables), and
c (b) variables which are communicated between subroutines.
c the structure of each block is as follows..  all real variables are
c listed first, followed by all integers.  within each type, the
c variables are grouped with those local to subroutine lsodes first,
c then those local to subroutine stode or subroutine prjs
c (no other routines have own variables), and finally those used
c for communication.  the block ls0001 is declared in subroutines
c lsodes, iprep, prep, intdy, stode, prjs, and slss.  the block lss001
c is declared in subroutines lsodes, iprep, prep, prjs, and slss.
c groups of variables are replaced by dummy arrays in the common
c declarations in routines where those variables are not used.
c-----------------------------------------------------------------------
      common /ls0001/ rowns(209),
     1   ccmax, el0, h, hmin, hmxi, hu, rc, tn, uround,
     2   illin, init, lyh, lewt, lacor, lsavf, lwm, liwm,
     3   mxstep, mxhnil, nhnil, ntrep, nslast, nyh, iowns(6),
     4   icf, ierpj, iersl, jcur, jstart, kflag, l, meth, miter,
     5   maxord, maxcor, msbp, mxncf, n, nq, nst, nfe, nje, nqu
c
      common /lss001/ con0, conmin, ccmxj, psmall, rbig, seth,
     1   iplost, iesp, istatc, iys, iba, ibian, ibjan, ibjgp,
     2   ipian, ipjan, ipjgp, ipigp, ipr, ipc, ipic, ipisp, iprsp, ipa,
     3   lenyh, lenyhm, lenwk, lreq, lrat, lrest, lwmin, moss, msbj,
     4   nslj, ngp, nlu, nnz, nsp, nzl, nzu
c
      data mord(1),mord(2)/12,5/, mxstp0/500/, mxhnl0/10/
c-----------------------------------------------------------------------
c in the data statement below, set lenrat equal to the ratio of
c the wordlength for a real number to that for an integer.  usually,
c lenrat = 1 for single precision and 2 for double precision.  if the
c true ratio is not an integer, use the next smaller integer (.ge. 1).
c-----------------------------------------------------------------------
      data lenrat/2/
c-----------------------------------------------------------------------
c block a.
c this code block is executed on every call.
c it tests istate and itask for legality and branches appropriately.
c if istate .gt. 1 but the flag init shows that initialization has
c not yet been done, an error return occurs.
c if istate = 1 and tout = t, jump to block g and return immediately.
c-----------------------------------------------------------------------
      if (istate .lt. 1 .or. istate .gt. 3) go to 601
      if (itask .lt. 1 .or. itask .gt. 5) go to 602
      if (istate .eq. 1) go to 10
      if (init .eq. 0) go to 603
      if (istate .eq. 2) go to 200
      go to 20
```

```
 10   init = 0
      if (tout .eq. t) go to 430
 20   ntrep = 0
c-----------------------------------------------------------------------
c block b.
c the next code block is executed for the initial call (istate = 1),
c or for a continuation call with parameter changes (istate = 3).
c it contains checking of all inputs and various initializations.
c if istate = 1, the final setting of work space pointers, the matrix
c preprocessing, and other initializations are done in block c.
c
c first check legality of the non-optional inputs neq, itol, iopt,
c mf, ml, and mu.
c-----------------------------------------------------------------------
      if (neq(1) .le. 0) go to 604
      if (istate .eq. 1) go to 25
      if (neq(1) .gt. n) go to 605
 25   n = neq(1)
      if (itol .lt. 1 .or. itol .gt. 4) go to 606
      if (iopt .lt. 0 .or. iopt .gt. 1) go to 607
      moss = mf/100
      mf1 = mf - 100*moss
      meth = mf1/10
      miter = mf1 - 10*meth
      if (moss .lt. 0 .or. moss .gt. 2) go to 608
      if (meth .lt. 1 .or. meth .gt. 2) go to 608
      if (miter .lt. 0 .or. miter .gt. 3) go to 608
      if (miter .eq. 0 .or. miter .eq. 3) moss = 0
c next process and check the optional inputs. --------------------------
      if (iopt .eq. 1) go to 40
      maxord = mord(meth)
      mxstep = mxstp0
      mxhnil = mxhnl0
      if (istate .eq. 1) h0 = 0.0d0
      hmxi = 0.0d0
      hmin = 0.0d0
      seth = 0.0d0
      go to 60
 40   maxord = iwork(5)
      if (maxord .lt. 0) go to 611
      if (maxord .eq. 0) maxord = 100
      maxord = min0(maxord,mord(meth))
      mxstep = iwork(6)
      if (mxstep .lt. 0) go to 612
      if (mxstep .eq. 0) mxstep = mxstp0
      mxhnil = iwork(7)
      if (mxhnil .lt. 0) go to 613
      if (mxhnil .eq. 0) mxhnil = mxhnl0
      if (istate .ne. 1) go to 50
      h0 = rwork(5)
      if ((tout - t)*h0 .lt. 0.0d0) go to 614
 50   hmax = rwork(6)
      if (hmax .lt. 0.0d0) go to 615
      hmxi = 0.0d0
      if (hmax .gt. 0.0d0) hmxi = 1.0d0/hmax
      hmin = rwork(7)
      if (hmin .lt. 0.0d0) go to 616
      seth = rwork(8)
      if (seth .lt. 0.0d0) go to 609
```

```
c check rtol and atol for legality. ----------------------------------
 60   rtoli = rtol(1)
      atoli = atol(1)
      do 65 i = 1,n
        if (itol .ge. 3) rtoli = rtol(i)
        if (itol .eq. 2 .or. itol .eq. 4) atoli = atol(i)
        if (rtoli .lt. 0.0d0) go to 619
        if (atoli .lt. 0.0d0) go to 620
 65     continue
c-----------------------------------------------------------------------
c compute required work array lengths, as far as possible, and test
c these against lrw and liw.  then set tentative pointers for work
c arrays.  pointers to rwork/iwork segments are named by prefixing l to
c the name of the segment.  e.g., the segment yh starts at rwork(lyh).
c segments of rwork (in order) are denoted  wm, yh, savf, ewt, acor.
c if miter = 1 or 2, the required length of the matrix work space wm
c is not yet known, and so a crude minimum value is used for the
c initial tests of lrw and liw, and yh is temporarily stored as far
c to the right in rwork as possible, to leave the maximum amount
c of space for wm for matrix preprocessing.  thus if miter = 1 or 2
c and moss .ne. 2, some of the segments of rwork are temporarily
c omitted, as they are not needed in the preprocessing.  these
c omitted segments are.. acor if istate = 1, ewt and acor if istate = 3
c and moss = 1, and savf, ewt, and acor if istate = 3 and moss = 0.
c-----------------------------------------------------------------------
      lrat = lenrat
      if (istate .eq. 1) nyh = n
      lwmin = 0
      if (miter .eq. 1) lwmin = 4*n + 10*n/lrat
      if (miter .eq. 2) lwmin = 4*n + 11*n/lrat
      if (miter .eq. 3) lwmin = n + 2
      lenyh = (maxord+1)*nyh
      lrest = lenyh + 3*n
      lenrw = 20 + lwmin + lrest
      iwork(17) = lenrw
      leniw = 30
      if (moss .eq. 0 .and. miter .ne. 0 .and. miter .ne. 3)
     1   leniw = leniw + n + 1
      iwork(18) = leniw
      if (lenrw .gt. lrw) go to 617
      if (leniw .gt. liw) go to 618
      lia = 31
      if (moss .eq. 0 .and. miter .ne. 0 .and. miter .ne. 3)
     1   leniw = leniw + iwork(lia+n) - 1
      iwork(18) = leniw
      if (leniw .gt. liw) go to 618
      lja = lia + n + 1
      lia = min0(lia,liw)
      lja = min0(lja,liw)
      lwm = 21
      if (istate .eq. 1) nq = 1
      ncolm = min0(nq+1,maxord+2)
      lenyhm = ncolm*nyh
      lenyht = lenyh
      if (miter .eq. 1 .or. miter .eq. 2) lenyht = lenyhm
      imul = 2
      if (istate .eq. 3) imul = moss
      if (moss .eq. 2) imul = 3
      lrtem = lenyht + imul*n
```

```
      lwtem = lwmin
      if (miter .eq. 1 .or. miter .eq. 2) lwtem = lrw - 20 - lrtem
      lenwk = lwtem
      lyhn = lwm + lwtem
      lsavf = lyhn + lenyht
      lewt = lsavf + n
      lacor = lewt + n
      istatc = istate
      if (istate .eq. 1) go to 100
c-----------------------------------------------------------------------
c istate = 3.  move yh to its new location.
c note that only the part of yh needed for the next step, namely
c min(nq+1,maxord+2) columns, is actually moved.
c a temporary error weight array ewt is loaded if moss = 2.
c sparse matrix processing is done in iprep/prep if miter = 1 or 2.
c if maxord was reduced below nq, then the pointers are finally set
c so that savf is identical to yh(*,maxord+2).
c-----------------------------------------------------------------------
      lyhd = lyh - lyhn
      imax = lyhn - 1 + lenyhm
c move yh.  branch for move right, no move, or move left. --------------
      if (lyhd) 70,80,74
 70   do 72 i = lyhn,imax
        j = imax + lyhn - i
 72     rwork(j) = rwork(j+lyhd)
      go to 80
 74   do 76 i = lyhn,imax
 76     rwork(i) = rwork(i+lyhd)
 80   lyh = lyhn
      iwork(22) = lyh
      if (miter .eq. 0 .or. miter .eq. 3) go to 92
      if (moss .ne. 2) go to 85
c temporarily load ewt if miter = 1 or 2 and moss = 2. ----------------
      call ewset (n, itol, rtol, atol, rwork(lyh), rwork(lewt))
      do 82 i = 1,n
        if (rwork(i+lewt-1) .le. 0.0d0) go to 621
 82     rwork(i+lewt-1) = 1.0d0/rwork(i+lewt-1)
 85   continue
c iprep and prep do sparse matrix preprocessing if miter = 1 or 2. -----
      lsavf = min0(lsavf,lrw)
      lewt = min0(lewt,lrw)
      lacor = min0(lacor,lrw)
      call iprep (neq, y, rwork, iwork(lia), iwork(lja), ipflag, f, jac)
      lenrw = lwm - 1 + lenwk + lrest
      iwork(17) = lenrw
      if (ipflag .ne. -1) iwork(23) = ipian
      if (ipflag .ne. -1) iwork(24) = ipjan
      ipgo = -ipflag + 1
      go to (90, 628, 629, 630, 631, 632, 633), ipgo
 90   iwork(22) = lyh
      if (lenrw .gt. lrw) go to 617
c set flag to signal parameter changes to stode. -----------------------
 92   jstart = -1
      if (n .eq. nyh) go to 200
c neq was reduced.  zero part of yh to avoid undefined references. -----
      i1 = lyh + l*nyh
      i2 = lyh + (maxord + 1)*nyh - 1
      if (i1 .gt. i2) go to 200
      do 95 i = i1,i2
```

```
 95     rwork(i) = 0.0d0
      go to 200
c-----------------------------------------------------------------------
c block c.
c the next block is for the initial call only (istate = 1).
c it contains all remaining initializations, the initial call to f,
c the sparse matrix preprocessing (miter = 1 or 2), and the
c calculation of the initial step size.
c the error weights in ewt are inverted after being loaded.
c-----------------------------------------------------------------------
 100  continue
      lyh = lyhn
      iwork(22) = lyh
      tn = t
      nst = 0
      h = 1.0d0
      nnz = 0
      ngp = 0
      nzl = 0
      nzu = 0
c load the initial value vector in yh. --------------------------------
      do 105 i = 1,n
 105    rwork(i+lyh-1) = y(i)
c initial call to f.  (lf0 points to yh(*,2).) -------------------------
      lf0 = lyh + nyh
      call f (neq, t, y, rwork(lf0))
      nfe = 1
c load and invert the ewt array.  (h is temporarily set to 1.0.) -------
      call ewset (n, itol, rtol, atol, rwork(lyh), rwork(lewt))
      do 110 i = 1,n
        if (rwork(i+lewt-1) .le. 0.0d0) go to 621
 110    rwork(i+lewt-1) = 1.0d0/rwork(i+lewt-1)
      if (miter .eq. 0 .or. miter .eq. 3) go to 120
c iprep and prep do sparse matrix preprocessing if miter = 1 or 2. -----
      lacor = min0(lacor,lrw)
      call iprep (neq, y, rwork, iwork(lia), iwork(lja), ipflag, f, jac)
      lenrw = lwm - 1 + lenwk + lrest
      iwork(17) = lenrw
      if (ipflag .ne. -1) iwork(23) = ipian
      if (ipflag .ne. -1) iwork(24) = ipjan
      ipgo = -ipflag + 1
      go to (115, 628, 629, 630, 631, 632, 633), ipgo
 115  iwork(22) = lyh
      if (lenrw .gt. lrw) go to 617
c check tcrit for legality (itask = 4 or 5). ---------------------------
 120  continue
      if (itask .ne. 4 .and. itask .ne. 5) go to 125
      tcrit = rwork(1)
      if ((tcrit - tout)*(tout - t) .lt. 0.0d0) go to 625
      if (h0 .ne. 0.0d0 .and. (t + h0 - tcrit)*h0 .gt. 0.0d0)
     1   h0 = tcrit - t
c initialize all remaining parameters. --------------------------------
 125  uround = d1mach(4)
      jstart = 0
      if (miter .ne. 0) rwork(lwm) = dsqrt(uround)
      msbj = 50
      nslj = 0
      ccmxj = 0.2d0
      psmall = 1000.0d0*uround
```

```
      rbig = 0.01d0/psmall
      nhnil = 0
      nje = 0
      nlu = 0
      nslast = 0
      hu = 0.0d0
      nqu = 0
      ccmax = 0.3d0
      maxcor = 3
      msbp = 20
      mxncf = 10
c-----------------------------------------------------------------------
c the coding below computes the step size, h0, to be attempted on the
c first step, unless the user has supplied a value for this.
c first check that tout - t differs significantly from zero.
c a scalar tolerance quantity tol is computed, as max(rtol(i))
c if this is positive, or max(atol(i)/abs(y(i))) otherwise, adjusted
c so as to be between 100*uround and 1.0e-3.
c then the computed value h0 is given by..
c                                          neq
c   h0**2 = tol / ( w0**-2 + (1/neq) * sum ( f(i)/ywt(i) )**2  )
c                                           1
c where   w0     = max ( abs(t), abs(tout) ),
c         f(i)   = i-th component of initial value of f,
c         ywt(i) = ewt(i)/tol  (a weight for y(i)).
c the sign of h0 is inferred from the initial values of tout and t.
c-----------------------------------------------------------------------
      lf0 = lyh + nyh
      if (h0 .ne. 0.0d0) go to 180
      tdist = dabs(tout - t)
      w0 = dmax1(dabs(t),dabs(tout))
      if (tdist .lt. 2.0d0*uround*w0) go to 622
      tol = rtol(1)
      if (itol .le. 2) go to 140
      do 130 i = 1,n
 130    tol = dmax1(tol,rtol(i))
 140  if (tol .gt. 0.0d0) go to 160
      atoli = atol(1)
      do 150 i = 1,n
        if (itol .eq. 2 .or. itol .eq. 4) atoli = atol(i)
        ayi = dabs(y(i))
        if (ayi .ne. 0.0d0) tol = dmax1(tol,atoli/ayi)
 150    continue
 160  tol = dmax1(tol,100.0d0*uround)
      tol = dmin1(tol,0.001d0)
      sum = vnorm (n, rwork(lf0), rwork(lewt))
      sum = 1.0d0/(tol*w0*w0) + tol*sum**2
      h0 = 1.0d0/dsqrt(sum)
      h0 = dmin1(h0,tdist)
      h0 = dsign(h0,tout-t)
c adjust h0 if necessary to meet hmax bound. --------------------------
 180  rh = dabs(h0)*hmxi
      if (rh .gt. 1.0d0) h0 = h0/rh
c load h with h0 and scale yh(*,2) by h0. ----------------------------
      h = h0
      do 190 i = 1,n
 190    rwork(i+lf0-1) = h0*rwork(i+lf0-1)
      go to 270
c-----------------------------------------------------------------------
```

```
c block d.
c the next code block is for continuation calls only (istate = 2 or 3)
c and is to check stop conditions before taking a step.
c-----------------------------------------------------------------------
 200  nslast = nst
      go to (210, 250, 220, 230, 240), itask
 210  if ((tn - tout)*h .lt. 0.0d0) go to 250
      call intdy (tout, 0, rwork(lyh), nyh, y, iflag)
      if (iflag .ne. 0) go to 627
      t = tout
      go to 420
 220  tp = tn - hu*(1.0d0 + 100.0d0*uround)
      if ((tp - tout)*h .gt. 0.0d0) go to 623
      if ((tn - tout)*h .lt. 0.0d0) go to 250
      go to 400
 230  tcrit = rwork(1)
      if ((tn - tcrit)*h .gt. 0.0d0) go to 624
      if ((tcrit - tout)*h .lt. 0.0d0) go to 625
      if ((tn - tout)*h .lt. 0.0d0) go to 245
      call intdy (tout, 0, rwork(lyh), nyh, y, iflag)
      if (iflag .ne. 0) go to 627
      t = tout
      go to 420
 240  tcrit = rwork(1)
      if ((tn - tcrit)*h .gt. 0.0d0) go to 624
 245  hmx = dabs(tn) + dabs(h)
      ihit = dabs(tn - tcrit) .le. 100.0d0*uround*hmx
      if (ihit) go to 400
      tnext = tn + h*(1.0d0 + 4.0d0*uround)
      if ((tnext - tcrit)*h .le. 0.0d0) go to 250
      h = (tcrit - tn)*(1.0d0 - 4.0d0*uround)
      if (istate .eq. 2) jstart = -2
c-----------------------------------------------------------------------
c block e.
c the next block is normally executed for all calls and contains
c the call to the one-step core integrator stode.
c
c this is a looping point for the integration steps.
c
c first check for too many steps being taken, update ewt (if not at
c start of problem), check for too much accuracy being requested, and
c check for h below the roundoff level in t.
c-----------------------------------------------------------------------
 250  continue
      if ((nst-nslast) .ge. mxstep) go to 500
      call ewset (n, itol, rtol, atol, rwork(lyh), rwork(lewt))
      do 260 i = 1,n
        if (rwork(i+lewt-1) .le. 0.0d0) go to 510
 260    rwork(i+lewt-1) = 1.0d0/rwork(i+lewt-1)
 270  tolsf = uround*vnorm (n, rwork(lyh), rwork(lewt))
      if (tolsf .le. 1.0d0) go to 280
      tolsf = tolsf*2.0d0
      if (nst .eq. 0) go to 626
      go to 520
 280  if ((tn + h) .ne. tn) go to 290
      nhnil = nhnil + 1
      if (nhnil .gt. mxhnil) go to 290

CCC got rid of error messages - Anne 8/2/95
```

```
CCC      call xerrwv('lsodes-- warning..internal t (=r1) and h (=r2) are',
CCC     1   50, 101, 0, 0, 0, 0, 0, 0.0d0, 0.0d0)
CCC      call xerrwv(
CCC     1  '      such that in the machine, t + h = t on the next step  ',
CCC     1   60, 101, 0, 0, 0, 0, 0, 0.0d0, 0.0d0)
CCC      call xerrwv('      (h = step size). solver will continue anyway',
CCC     1   50, 101, 0, 0, 0, 0, 2, tn, h)
      if (nhnil .lt. mxhnil) go to 290

CCC got rid of error messages - Anne 8/2/95
CCC      call xerrwv('lsodes-- above warning has been issued i1 times.  ',
CCC     1   50, 102, 0, 0, 0, 0, 0, 0.0d0, 0.0d0)
CCC      call xerrwv('      it will not be issued again for this problem',
CCC     1   50, 102, 0, 1, mxhnil, 0, 0, 0.0d0, 0.0d0)
 290  continue
c-----------------------------------------------------------------------
c    call stode(neq,y,yh,nyh,yh,ewt,savf,acor,wm,wm,f,jac,prjs,slss)
c-----------------------------------------------------------------------
      call stode (neq, y, rwork(lyh), nyh, rwork(lyh), rwork(lewt),
     1   rwork(lsavf), rwork(lacor), rwork(lwm), rwork(lwm),
     2   f, jac, prjs, slss)
      kgo = 1 - kflag
      go to (300, 530, 540, 550), kgo
c-----------------------------------------------------------------------
c block f.
c the following block handles the case of a successful return from the
c core integrator (kflag = 0).  test for stop conditions.
c-----------------------------------------------------------------------
 300  init = 1
      go to (310, 400, 330, 340, 350), itask
c itask = 1.  if tout has been reached, interpolate. -------------------
 310  if ((tn - tout)*h .lt. 0.0d0) go to 250
      call intdy (tout, 0, rwork(lyh), nyh, y, iflag)
      t = tout
      go to 420
c itask = 3.  jump to exit if tout was reached. ------------------------
 330  if ((tn - tout)*h .ge. 0.0d0) go to 400
      go to 250
c itask = 4.  see if tout or tcrit was reached.  adjust h if necessary.
 340  if ((tn - tout)*h .lt. 0.0d0) go to 345
      call intdy (tout, 0, rwork(lyh), nyh, y, iflag)
      t = tout
      go to 420
 345  hmx = dabs(tn) + dabs(h)
      ihit = dabs(tn - tcrit) .le. 100.0d0*uround*hmx
      if (ihit) go to 400
      tnext = tn + h*(1.0d0 + 4.0d0*uround)
      if ((tnext - tcrit)*h .le. 0.0d0) go to 250
      h = (tcrit - tn)*(1.0d0 - 4.0d0*uround)
      jstart = -2
      go to 250
c itask = 5.  see if tcrit was reached and jump to exit. ---------------
 350  hmx = dabs(tn) + dabs(h)
      ihit = dabs(tn - tcrit) .le. 100.0d0*uround*hmx
c-----------------------------------------------------------------------
c block g.
c the following block handles all successful returns from lsodes.
c if itask .ne. 1, y is loaded from yh and t is set accordingly.
c istate is set to 2, the illegal input counter is zeroed, and the
```

```
c optional outputs are loaded into the work arrays before returning.
c if istate = 1 and tout = t, there is a return with no action taken,
c except that if this has happened repeatedly, the run is terminated.
c-----------------------------------------------------------------------
 400  do 410 i = 1,n
 410    y(i) = rwork(i+lyh-1)
      t = tn
      if (itask .ne. 4 .and. itask .ne. 5) go to 420
      if (ihit) t = tcrit
 420  istate = 2
      illin = 0
      rwork(11) = hu
      rwork(12) = h
      rwork(13) = tn
      iwork(11) = nst
      iwork(12) = nfe
      iwork(13) = nje
      iwork(14) = nqu
      iwork(15) = nq
      iwork(19) = nnz
      iwork(20) = ngp
      iwork(21) = nlu
      iwork(25) = nzl
      iwork(26) = nzu
      return
c
 430  ntrep = ntrep + 1
CCC changed the 5 to a 23 to allow for more radionuclides to
CCC    be covered in any simulation - David Updegraf 7/7/98
CCC      if (ntrep .lt. 5) return
      if (ntrep .lt. 23)return
CCC got rid of error messages - Anne 8/2/95
CCC      call xerrwv(
CCC     1  'lsodes-- repeated calls with istate = 1 and tout = t (=r1)
CCC     1   60, 301, 0, 0, 0, 0, 1, t, 0.0d0)
CCC changed 'go to 800' to 'go to 710'
CCC      go to 800
      go to 710
c-----------------------------------------------------------------------
c block h.
c the following block handles all unsuccessful returns other than
c those for illegal input.  first the error message routine is called.
c if there was an error test or convergence test failure, imxer is set.
c then y is loaded from yh, t is set to tn, and the illegal input
c counter illin is set to 0.  the optional outputs are loaded into
c the work arrays before returning.
c-----------------------------------------------------------------------
c the maximum number of steps was taken before reaching tout. ----------

CCC got rid of error messages - Anne 8/2/95
CCC
CCC 500  call xerrwv('lsodes-- at current t (=r1), mxstep (=i1) steps   ',
CCCC     1   50, 201, 0, 0, 0, 0, 0, 0.0d0, 0.0d0)
CCC      call xerrwv('      taken on this call before reaching tout     ',
CCC     1   50, 201, 0, 1, mxstep, 0, 1, tn, 0.0d0)
500      istate = -1
      go to 580
c ewt(i) .le. 0.0 for some i (not at start of problem). ----------------
```

```
510  ewti = rwork(lewt+i-1)
CCC got rid of error messages - Anne 8/2/95
CCC       call xerrwv('lsodes-- at t (=r1), ewt(i1) has become r2 .le. 0.',
CCC      1   50, 202, 0, 1, i, 0, 2, tn, ewti)
      istate = -6
      go to 580
c too much accuracy requested for machine precision. -------------------

CCC got rid of error messages - Anne 8/2/95
CCC 520  call xerrwv('lsodes-- at t (=r1), too much accuracy requested  ',
CCC      1   50, 203, 0, 0, 0, 0, 0, 0.0d0, 0.0d0)
CCC       call xerrwv('      for precision of machine..  see tolsf (=r2) ',
CCC      1   50, 203, 0, 0, 0, 0, 2, tn, tolsf)
520   rwork(14) = tolsf
      istate = -2
      go to 580
c kflag = -1.  error test failed repeatedly or with abs(h) = hmin. -----

CCC got rid of error messages - Anne 8/2/95
CCC 530  call xerrwv('lsodes-- at t(=r1) and step size h(=r2), the error',
CCC      1   50, 204, 0, 0, 0, 0, 0, 0.0d0, 0.0d0)
CCC       call xerrwv('      test failed repeatedly or with abs(h) = hmin',
CCC      1   50, 204, 0, 0, 0, 0, 2, tn, h)
530   istate = -4
      go to 560
c kflag = -2.  convergence failed repeatedly or with abs(h) = hmin. ----
CCC 540  call xerrwv('lsodes-- at t (=r1) and step size h (=r2), the    ',
CCC      1   50, 205, 0, 0, 0, 0, 0, 0.0d0, 0.0d0)
CCC       call xerrwv('      corrector convergence failed repeatedly     ',
CCC      1   50, 205, 0, 0, 0, 0, 0, 0.0d0, 0.0d0)
CCC       call xerrwv('      or with abs(h) = hmin   ',
CCC      1   30, 205, 0, 0, 0, 0, 2, tn, h)
540   istate = -5
      go to 560
c kflag = -3.  fatal error flag returned by prjs or slss (cdrv). -------
CCC 550  call xerrwv('lsodes-- at t (=r1) and step size h (=r2), a fatal',
CCC      1   50, 207, 0, 0, 0, 0, 0, 0.0d0, 0.0d0)
CCC       call xerrwv('      error flag was returned by cdrv (by way of  ',
CCC      1   50, 207, 0, 0, 0, 0, 0, 0.0d0, 0.0d0)
CCC       call xerrwv('      subroutine prjs or slss)',
CCC      1   30, 207, 0, 0, 0, 0, 2, tn, h)
550   istate = -7
      go to 580
c compute imxer if relevant. ----------------------------------------
560  big = 0.0d0
      imxer = 1
      do 570 i = 1,n
        size = dabs(rwork(i+lacor-1)*rwork(i+lewt-1))
        if (big .ge. size) go to 570
        big = size
        imxer = i
570    continue
      iwork(16) = imxer
c set y vector, t, illin, and optional outputs. ------------------------
580  do 590 i = 1,n
590    y(i) = rwork(i+lyh-1)
      t = tn
      illin = 0
      rwork(11) = hu
```

```
      rwork(12) = h
      rwork(13) = tn
      iwork(11) = nst
      iwork(12) = nfe
      iwork(13) = nje
      iwork(14) = nqu
      iwork(15) = nq
      iwork(19) = nnz
      iwork(20) = ngp
      iwork(21) = nlu
      iwork(25) = nzl
      iwork(26) = nzu
      return
c-----------------------------------------------------------------------
c block i.
c the following block handles all error returns due to illegal input
c (istate = -3), as detected before calling the core integrator.
c first the error message routine is called.  then if there have been
c 5 consecutive such returns just before this call to the solver,
c the run is halted.
c-----------------------------------------------------------------------
CCC 601  call xerrwv('lsodes-- istate (=i1) illegal ',
CCC     1   30, 1, 0, 1, istate, 0, 0, 0.0d0, 0.0d0)
601   go to 700
CCC 602  call xerrwv('lsodes-- itask (=i1) illegal  ',
CCC     1   30, 2, 0, 1, itask, 0, 0, 0.0d0, 0.0d0)
602   go to 700
CCC 603  call xerrwv('lsodes-- istate .gt. 1 but lsodes not initialized ',
CCC     1   50, 3, 0, 0, 0, 0, 0, 0.0d0, 0.0d0)
603   go to 700
CCC 604  call xerrwv('lsodes-- neq (=i1) .lt. 1     ',
CCC     1   30, 4, 0, 1, neq(1), 0, 0, 0.0d0, 0.0d0)
604   go to 700
CCC 605  call xerrwv('lsodes-- istate = 3 and neq increased (i1 to i2)  ',
CCC     1   50, 5, 0, 2, n, neq(1), 0, 0.0d0, 0.0d0)
605   go to 700
CCC 606  call xerrwv('lsodes-- itol (=i1) illegal   ',
CCC     1   30, 6, 0, 1, itol, 0, 0, 0.0d0, 0.0d0)
606   go to 700
CCC 607  call xerrwv('lsodes-- iopt (=i1) illegal   ',
CCC     1   30, 7, 0, 1, iopt, 0, 0, 0.0d0, 0.0d0)
607   go to 700
CCC 608  call xerrwv('lsodes-- mf (=i1) illegal     ',
CCC     1   30, 8, 0, 1, mf, 0, 0, 0.0d0, 0.0d0)
608   go to 700
CCC 609  call xerrwv('lsodes-- seth (=r1) .lt. 0.0  ',
CCC     1   30, 9, 0, 0, 0, 0, 1, seth, 0.0d0)
609   go to 700
CCC 611  call xerrwv('lsodes-- maxord (=i1) .lt. 0  ',
CCC     1   30, 11, 0, 1, maxord, 0, 0, 0.0d0, 0.0d0)
611   go to 700
CCC 612  call xerrwv('lsodes-- mxstep (=i1) .lt. 0  ',
CCC     1   30, 12, 0, 1, mxstep, 0, 0, 0.0d0, 0.0d0)
612   go to 700
CCC 613  call xerrwv('lsodes-- mxhnil (=i1) .lt. 0  ',
CCC     1   30, 13, 0, 1, mxhnil, 0, 0, 0.0d0, 0.0d0)
613   go to 700
CCC 614  call xerrwv('lsodes-- tout (=r1) behind t (=r2)
CCC     1   40, 14, 0, 0, 0, 0, 2, tout, t)
```

```
CCC      call xerrwv('      integration direction is given by h0 (=r1)  ',
CCC     1   50, 14, 0, 0, 0, 0, 1, h0, 0.0d0)
614   go to 700
CCC 615  call xerrwv('lsodes-- hmax (=r1) .lt. 0.0  ',
CCC     1   30, 15, 0, 0, 0, 0, 1, hmax, 0.0d0)
615   go to 700
CCC 616  call xerrwv('lsodes-- hmin (=r1) .lt. 0.0  ',
CCC     1   30, 16, 0, 0, 0, 0, 1, hmin, 0.0d0)
616   go to 700
CCC 617  call xerrwv('lsodes-- rwork length is insufficient to proceed. ',
CCC     1   50, 17, 0, 0, 0, 0, 0, 0.0d0, 0.0d0)
CCC      call xerrwv(
CCC     1  '        length needed is .ge. lenrw (=i1), exceeds lrw (=i2)',
CCC     1   60, 17, 0, 2, lenrw, lrw, 0, 0.0d0, 0.0d0)
617   go to 700
CCC 618  call xerrwv('lsodes-- iwork length is insufficient to proceed. ',
CCC     1   50, 18, 0, 0, 0, 0, 0, 0.0d0, 0.0d0)
CCC      call xerrwv(
CCC     1  '        length needed is .ge. leniw (=i1), exceeds liw (=i2)',
CCC     1   60, 18, 0, 2, leniw, liw, 0, 0.0d0, 0.0d0)
618   go to 700
CCC 619  call xerrwv('lsodes-- rtol(i1) is r1 .lt. 0.0        ',
CCC     1   40, 19, 0, 1, i, 0, 1, rtoli, 0.0d0)
619   go to 700
CCC 620  call xerrwv('lsodes-- atol(i1) is r1 .lt. 0.0        ',
CCC     1   40, 20, 0, 1, i, 0, 1, atoli, 0.0d0)
620   go to 700

 621  ewti = rwork(lewt+i-1)
CCC      call xerrwv('lsodes-- ewt(i1) is r1 .le. 0.0         ',
CCC     1   40, 21, 0, 1, i, 0, 1, ewti, 0.0d0)
      go to 700
CCC 622  call xerrwv(
CCC     1  'lsodes-- tout (=r1) too close to t(=r2) to start integration',
CCC     1   60, 22, 0, 0, 0, 0, 2, tout, t)
622   go to 700
CCC 623  call xerrwv(
CCC     1  'lsodes-- itask = i1 and tout (=r1) behind tcur - hu (= r2)  ',
CCC     1   60, 23, 0, 1, itask, 0, 2, tout, tp)
623   go to 700
CCC 624  call xerrwv(
CCC     1  'lsodes-- itask = 4 or 5 and tcrit (=r1) behind tcur (=r2)   ',
CCC     1   60, 24, 0, 0, 0, 0, 2, tcrit, tn)
624   go to 700
CCC 625  call xerrwv(
CCC     1  'lsodes-- itask = 4 or 5 and tcrit (=r1) behind tout (=r2)   ',
CCC     1   60, 25, 0, 0, 0, 0, 2, tcrit, tout)
625   go to 700
CCC 626  call xerrwv('lsodes-- at start of problem, too much accuracy   ',
CCC     1   50, 26, 0, 0, 0, 0, 0, 0.0d0, 0.0d0)
CCC      call xerrwv(
CCC     1  '      requested for precision of machine..  see tolsf (=r1) ',
CCC     1   60, 26, 0, 0, 0, 0, 1, tolsf, 0.0d0)
626   rwork(14) = tolsf
      go to 700
CCC 627  call xerrwv('lsodes-- trouble from intdy. itask = i1, tout = r1',
CCC     1   50, 27, 0, 1, itask, 0, 1, tout, 0.0d0)
627   go to 700
CCC 628  call xerrwv(
```

```
CCC     1  'lsodes-- rwork length insufficient (for subroutine prep).    ',
CCC     1   60, 28, 0, 0, 0, 0, 0, 0.0d0, 0.0d0)
CCC      call xerrwv(
CCC     1  '        length needed is .ge. lenrw (=i1), exceeds lrw (=i2)',
CCC     1   60, 28, 0, 2, lenrw, lrw, 0, 0.0d0, 0.0d0)
628   go to 700
CCC 629  call xerrwv(
CCC     1  'lsodes-- rwork length insufficient (for subroutine jgroup). ',
CCC     1   60, 29, 0, 0, 0, 0, 0, 0.0d0, 0.0d0)
CCC      call xerrwv(
CCC     1  '        length needed is .ge. lenrw (=i1), exceeds lrw (=i2)',
CCC     1   60, 29, 0, 2, lenrw, lrw, 0, 0.0d0, 0.0d0)
629   go to 700
CCC 630  call xerrwv(
CCC     1  'lsodes-- rwork length insufficient (for subroutine odrv).    ',
CCC     1   60, 30, 0, 0, 0, 0, 0, 0.0d0, 0.0d0)
CCC      call xerrwv(
CCC     1  '        length needed is .ge. lenrw (=i1), exceeds lrw (=i2)',
CCC     1   60, 30, 0, 2, lenrw, lrw, 0, 0.0d0, 0.0d0)
630   go to 700

CCC 631  call xerrwv(
CCC     1  'lsodes-- error from odrv in yale sparse matrix package       ',
CCC     1   60, 31, 0, 0, 0, 0, 0, 0.0d0, 0.0d0)
631   imul = (iys - 1)/n
      irem = iys - imul*n
CCC      call xerrwv(
CCC     1  '      at t (=r1), odrv returned error flag = i1*neq + i2.    ',
CCC     1   60, 31, 0, 2, imul, irem, 1, tn, 0.0d0)
      go to 700

CCC 632  call xerrwv(
CCC     1  'lsodes-- rwork length insufficient (for subroutine cdrv).    ',
CCC     1   60, 32, 0, 0, 0, 0, 0, 0.0d0, 0.0d0)
CCC      call xerrwv(
CCC     1  '        length needed is .ge. lenrw (=i1), exceeds lrw (=i2)',
CCC     1   60, 32, 0, 2, lenrw, lrw, 0, 0.0d0, 0.0d0)
632   go to 700

CCC 633  call xerrwv(
CCC     1  'lsodes-- error from cdrv in yale sparse matrix package       ',
CCC     1   60, 33, 0, 0, 0, 0, 0, 0.0d0, 0.0d0)
633   imul = (iys - 1)/n
      irem = iys - imul*n
CCC      call xerrwv(
CCC     1  '      at t (=r1), cdrv returned error flag = i1*neq + i2.    ',
CCC     1   60, 33, 0, 2, imul, irem, 1, tn, 0.0d0)
CCCC       if (imul .eq. 2) call xerrwv(
CCC     1  '        duplicate entry in sparsity structure descriptors    ',
CCC     1   60, 33, 0, 0, 0, 0, 0, 0.0d0, 0.0d0)
CCC      if (imul .eq. 3 .or. imul .eq. 6) call xerrwv(
CCC     1  '        insufficient storage for nsfc (called by cdrv)       ',
CCC     1   60, 33, 0, 0, 0, 0, 0, 0.0d0, 0.0d0)
c
 700  if (illin .eq. 5) go to 710
      illin = illin + 1
      istate = -3
      return
```

```
CCC710
CCC 710  call xerrwv('lsodes-- repeated occurrences of illegal input    ',
CCC     1   50, 302, 0, 0, 0, 0, 0, 0.0d0, 0.0d0)
c
CCC800
CCC 800  call xerrwv('lsodes-- run aborted.. apparent infinite loop     ',
CCC     1   50, 303, 2, 0, 0, 0, 0, 0.0d0, 0.0d0)
ccc   this used to be the plave where a stop would occur but changed
ccc   to flag istate as a fatal error stop
710   istate = -8
      return
c----------------------- end of subroutine lsodes ----------------------
      end


      subroutine iprep (neq, y, rwork, ia, ja, ipflag, f, jac)
cl11. optimize
      external f, jac
      integer neq, ia, ja, ipflag
      integer illin, init, lyh, lewt, lacor, lsavf, lwm, liwm,
     1   mxstep, mxhnil, nhnil, ntrep, nslast, nyh, iowns
      integer icf, ierpj, iersl, jcur, jstart, kflag, l, meth, miter,
     1   maxord, maxcor, msbp, mxncf, n, nq, nst, nfe, nje, nqu
      integer iplost, iesp, istatc, iys, iba, ibian, ibjan, ibjgp,
     1   ipian, ipjan, ipjgp, ipigp, ipr, ipc, ipic, ipisp, iprsp, ipa,
     2   lenyh, lenyhm, lenwk, lreq, lrat, lrest, lwmin, moss, msbj,
     3   nslj, ngp, nlu, nnz, nsp, nzl, nzu
      integer i, imax, lewtn, lyhd, lyhn
      double precision y, rwork
      double precision rowns,
     1   ccmax, el0, h, hmin, hmxi, hu, rc, tn, uround
      double precision rlss
      dimension neq(1), y(1), rwork(1), ia(1), ja(1)
      common /ls0001/ rowns(209),
     1   ccmax, el0, h, hmin, hmxi, hu, rc, tn, uround,
     2   illin, init, lyh, lewt, lacor, lsavf, lwm, liwm,
     3   mxstep, mxhnil, nhnil, ntrep, nslast, nyh, iowns(6),
     4   icf, ierpj, iersl, jcur, jstart, kflag, l, meth, miter,
     5   maxord, maxcor, msbp, mxncf, n, nq, nst, nfe, nje, nqu
      common /lss001/ rlss(6),
     1   iplost, iesp, istatc, iys, iba, ibian, ibjan, ibjgp,
     2   ipian, ipjan, ipjgp, ipigp, ipr, ipc, ipic, ipisp, iprsp, ipa,
     3   lenyh, lenyhm, lenwk, lreq, lrat, lrest, lwmin, moss, msbj,
     4   nslj, ngp, nlu, nnz, nsp, nzl, nzu
c-----------------------------------------------------------------------
c this routine serves as an interface between the driver and
c subroutine prep.  it is called only if miter is 1 or 2.
c tasks performed here are..
c  * call prep,
c  * reset the required wm segment length lenwk,
c  * move yh back to its final location (following wm in rwork),
c  * reset pointers for yh, savf, ewt, and acor, and
c  * move ewt to its new position if istate = 1.
c ipflag is an output error indication flag.  ipflag = 0 if there was
c no trouble, and ipflag is the value of the prep error flag ipper
c if there was trouble in subroutine prep.
c-----------------------------------------------------------------------
      ipflag = 0
c call prep to do matrix preprocessing operations. ---------------------
```

```
      call prep (neq, y, rwork(lyh), rwork(lsavf), rwork(lewt),
     1   rwork(lacor), ia, ja, rwork(lwm), rwork(lwm), ipflag, f, jac)
      lenwk = max0(lreq,lwmin)
      if (ipflag .lt. 0) return
c if prep was successful, move yh to end of required space for wm. -----
      lyhn = lwm + lenwk
      if (lyhn .gt. lyh) return
      lyhd = lyh - lyhn
      if (lyhd .eq. 0) go to 20
      imax = lyhn - 1 + lenyhm
      do 10 i = lyhn,imax
 10     rwork(i) = rwork(i+lyhd)
      lyh = lyhn
c reset pointers for savf, ewt, and acor. ------------------------------
 20   lsavf = lyh + lenyh
      lewtn = lsavf + n
      lacor = lewtn + n
      if (istatc .eq. 3) go to 40
c if istate = 1, move ewt (left) to its new position. ------------------
      if (lewtn .gt. lewt) return
      do 30 i = 1,n
 30     rwork(i+lewtn-1) = rwork(i+lewt-1)
 40   lewt = lewtn
      return
c----------------------- end of subroutine iprep -----------------------
      end


      subroutine prep (neq, y, yh, savf, ewt, ftem, ia, ja,
     1                     wk, iwk, ipper, f, jac)
clll. optimize
      external f,jac
      integer neq, ia, ja, iwk, ipper
      integer iownd, iowns,
     1   icf, ierpj, iersl, jcur, jstart, kflag, l, meth, miter,
     2   maxord, maxcor, msbp, mxncf, n, nq, nst, nfe, nje, nqu
      integer iplost, iesp, istatc, iys, iba, ibian, ibjan, ibjgp,
     1   ipian, ipjan, ipjgp, ipigp, ipr, ipc, ipic, ipisp, iprsp, ipa,
     2   lenyh, lenyhm, lenwk, lreq, lrat, lrest, lwmin, moss, msbj,
     3   nslj, ngp, nlu, nnz, nsp, nzl, nzu
      integer i, ibr, ier, ipil, ipiu, iptt1, iptt2, j, jfound, k,
     1   knew, kmax, kmin, ldif, lenigp, liwk, maxg, npl, nzsut
      double precision y, yh, savf, ewt, ftem, wk
      double precision rowns,
     1   ccmax, el0, h, hmin, hmxi, hu, rc, tn, uround
      double precision con0, conmin, ccmxj, psmall, rbig, seth
      double precision dq, dyj, erwt, fac, yj
      dimension neq(1), y(1), yh(1), savf(1), ewt(1), ftem(1),
     1   ia(1), ja(1), wk(1), iwk(1)
      common /ls0001/ rowns(209),
     2   ccmax, el0, h, hmin, hmxi, hu, rc, tn, uround,
     3   iownd(14), iowns(6),
     4   icf, ierpj, iersl, jcur, jstart, kflag, l, meth, miter,
     5   maxord, maxcor, msbp, mxncf, n, nq, nst, nfe, nje, nqu
      common /lss001/ con0, conmin, ccmxj, psmall, rbig, seth,
     1   iplost, iesp, istatc, iys, iba, ibian, ibjan, ibjgp,
     2   ipian, ipjan, ipjgp, ipigp, ipr, ipc, ipic, ipisp, iprsp, ipa,
     3   lenyh, lenyhm, lenwk, lreq, lrat, lrest, lwmin, moss, msbj,
     4   nslj, ngp, nlu, nnz, nsp, nzl, nzu
```

```
c-----------------------------------------------------------------------
c this routine performs preprocessing related to the sparse linear
c systems that must be solved if miter = 1 or 2.
c the operations that are performed here are..
c  * compute sparseness structure of jacobian according to moss,
c  * compute grouping of column indices (miter = 2),
c  * compute a new ordering of rows and columns of the matrix,
c  * reorder ja corresponding to the new ordering,
c  * perform a symbolic lu factorization of the matrix, and
c  * set pointers for segments of the iwk/wk array.
c in addition to variables described previously, prep uses the
c following for communication..
c yh     = the history array.  only the first column, containing the
c          current y vector, is used.  used only if moss .ne. 0.
c savf   = a work array of length neq, used only if moss .ne. 0.
c ewt    = array of length neq containing (inverted) error weights.
c          used only if moss = 2 or if istate = moss = 1.
c ftem   = a work array of length neq, identical to acor in the driver,
c          used only if moss = 2.
c wk     = a real work array of length lenwk, identical to wm in
c          the driver.
c iwk    = integer work array, assumed to occupy the same space as wk.
c lenwk  = the length of the work arrays wk and iwk.
c istatc = a copy of the driver input argument istate (= 1 on the
c          first call, = 3 on a continuation call).
c iys    = flag value from odrv or cdrv.
c ipper  = output error flag with the following values and meanings..
c           0  no error.
c          -1  insufficient storage for internal structure pointers.
c          -2  insufficient storage for jgroup.
c          -3  insufficient storage for odrv.
c          -4  other error flag from odrv (should never occur).
c          -5  insufficient storage for cdrv.
c          -6  other error flag from cdrv.
c-----------------------------------------------------------------------
      ibian = lrat*2
      ipian = ibian + 1
      npl = n + 1
      ipjan = ipian + npl
      ibjan = ipjan - 1
      liwk = lenwk*lrat
      if (ipjan+n-1 .gt. liwk) go to 210
      if (moss .eq. 0) go to 30
c
      if (istatc .eq. 3) go to 20
c istate = 1 and moss .ne. 0.  perturb y for structure determination. --
      do 10 i = 1,n
        erwt = 1.0d0/ewt(i)
        fac = 1.0d0 + 1.0d0/(dfloat(i)+1.0d0)
        y(i) = y(i) + fac*dsign(erwt,y(i))
 10     continue
      go to (70, 100), moss
c
 20   continue
c istate = 3 and moss .ne. 0.  load y from yh(*,1). --------------------
      do 25 i = 1,n
 25     y(i) = yh(i)
      go to (70, 100), moss
c
```

```
c moss = 0.  process user-s ia,ja.  add diagonal entries if necessary. -
 30   knew = ipjan
      kmin = ia(1)
      iwk(ipian) = 1
      do 60 j = 1,n
        jfound = 0
        kmax = ia(j+1) - 1
        if (kmin .gt. kmax) go to 45
        do 40 k = kmin,kmax
          i = ja(k)
          if (i .eq. j) jfound = 1
          if (knew .gt. liwk) go to 210
          iwk(knew) = i
          knew = knew + 1
 40       continue
        if (jfound .eq. 1) go to 50
 45     if (knew .gt. liwk) go to 210
        iwk(knew) = j
        knew = knew + 1
 50     iwk(ipian+j) = knew + 1 - ipjan
        kmin = kmax + 1
 60     continue
      go to 140
c
c moss = 1.  compute structure from user-supplied jacobian routine jac.
 70   continue
c a dummy call to f allows user to create temporaries for use in jac. --
      call f (neq, tn, y, savf)
      k = ipjan
      iwk(ipian) = 1
      do 90 j = 1,n
        if (k .gt. liwk) go to 210
        iwk(k) = j
        k = k + 1
        do 75 i = 1,n
 75       savf(i) = 0.0d0
        call jac (neq, tn, y, j, iwk(ipian), iwk(ipjan), savf)
        do 80 i = 1,n
          if (dabs(savf(i)) .le. seth) go to 80
          if (i .eq. j) go to 80
          if (k .gt. liwk) go to 210
          iwk(k) = i
          k = k + 1
 80       continue
        iwk(ipian+j) = k + 1 - ipjan
 90     continue
      go to 140
c
c moss = 2.  compute structure from results of n + 1 calls to f. -------
 100  k = ipjan
      iwk(ipian) = 1
      call f (neq, tn, y, savf)
      do 120 j = 1,n
        if (k .gt. liwk) go to 210
        iwk(k) = j
        k = k + 1
        yj = y(j)
        erwt = 1.0d0/ewt(j)
        dyj = dsign(erwt,yj)
```

```
        y(j) = yj + dyj
        call f (neq, tn, y, ftem)
        y(j) = yj
        do 110 i = 1,n
          dq = (ftem(i) - savf(i))/dyj
          if (dabs(dq) .le. seth) go to 110
          if (i .eq. j) go to 110
          if (k .gt. liwk) go to 210
          iwk(k) = i
          k = k + 1
 110      continue
        iwk(ipian+j) = k + 1 - ipjan
 120    continue
c
 140  continue
      if (moss .eq. 0 .or. istatc .ne. 1) go to 150
c if istate = 1 and moss .ne. 0, restore y from yh. --------------------
      do 145 i = 1,n
 145    y(i) = yh(i)
 150  nnz = iwk(ipian+n) - 1
      lenigp = 0
      ipigp = ipjan + nnz
      if (miter .ne. 2) go to 160
c
c compute grouping of column indices (miter = 2). ----------------------
      maxg = np1
      ipjgp = ipjan + nnz
      ibjgp = ipjgp - 1
      ipigp = ipjgp + n
      iptt1 = ipigp + np1
      iptt2 = iptt1 + n
      lreq = iptt2 + n - 1
      if (lreq .gt. liwk) go to 220
      call jgroup (n, iwk(ipian), iwk(ipjan), maxg, ngp, iwk(ipigp),
     1   iwk(ipjgp), iwk(iptt1), iwk(iptt2), ier)
      if (ier .ne. 0) go to 220
      lenigp = ngp + 1
c
c compute new ordering of rows/columns of jacobian. --------------------
 160  ipr = ipigp + lenigp
      ipc = ipr
      ipic = ipc + n
      ipisp = ipic + n
      iprsp = (ipisp - 2)/lrat + 2
      iesp = lenwk + 1 - iprsp
      if (iesp .lt. 0) go to 230
      ibr = ipr - 1
      do 170 i = 1,n
 170    iwk(ibr+i) = i
      nsp = liwk + 1 - ipisp
      call odrv (n, iwk(ipian), iwk(ipjan), wk, iwk(ipr), iwk(ipic),
     1   nsp, iwk(ipisp), 1, iys)
      if (iys .eq. 11*n+1) go to 240
      if (iys .ne. 0) go to 230
c
c reorder jan and do symbolic lu factorization of matrix. --------------
      ipa = lenwk + 1 - nnz
      nsp = ipa - iprsp
      lreq = max0(12*n/lrat, 6*n/lrat+2*n+nnz) + 3
```

```
      lreq = lreq + iprsp - 1 + nnz
      if (lreq .gt. lenwk) go to 250
      iba = ipa - 1
      do 180 i = 1,nnz
 180    wk(iba+i) = 0.0d0
      ipisp = lrat*(iprsp - 1) + 1
      call cdrv (n,iwk(ipr),iwk(ipc),iwk(ipic),iwk(ipian),iwk(ipjan),
     1   wk(ipa),wk(ipa),wk(ipa),nsp,iwk(ipisp),wk(iprsp),iesp,5,iys)
      lreq = lenwk - iesp
      if (iys .eq. 10*n+1) go to 250
      if (iys .ne. 0) go to 260
      ipil = ipisp
      ipiu = ipil + 2*n + 1
      nzu = iwk(ipil+n) - iwk(ipil)
      nzl = iwk(ipiu+n) - iwk(ipiu)
      if (lrat .gt. 1) go to 190
      call adjlr (n, iwk(ipisp), ldif)
      lreq = lreq + ldif
 190  continue
      if (lrat .eq. 2 .and. nnz .eq. n) lreq = lreq + 1
      nsp = nsp + lreq - lenwk
      ipa = lreq + 1 - nnz
      iba = ipa - 1
      ipper = 0
      return
c
 210  ipper = -1
      lreq = 2 + (2*n + 1)/lrat
      lreq = max0(lenwk+1,lreq)
      return
c
 220  ipper = -2
      lreq = (lreq - 1)/lrat + 1
      return
c
 230  ipper = -3
      call cntnzu (n, iwk(ipian), iwk(ipjan), nzsut)
      lreq = lenwk - iesp + (3*n + 4*nzsut - 1)/lrat + 1
      return
c
 240  ipper = -4
      return
c
 250  ipper = -5
      return
c
 260  ipper = -6
      lreq = lenwk
      return
c----------------------- end of subroutine prep ------------------------
      end
      subroutine jgroup (n,ia,ja,maxg,ngrp,igp,jgp,incl,jdone,ier)
cl11. optimize
      integer n, ia, ja, maxg, ngrp, igp, jgp, incl, jdone, ier
      dimension ia(1), ja(1), igp(1), jgp(n), incl(n), jdone(n)
c-----------------------------------------------------------------------
c this subroutine constructs groupings of the column indices of
c the jacobian matrix, used in the numerical evaluation of the
c jacobian by finite differences.
```

```
c
c input..
c n       = the order of the matrix.
c ia,ja   = sparse structure descriptors of the matrix by rows.
c maxg    = length of available storate in the igp array.
c
c output..
c ngrp    = number of groups.
c jgp     = array of length n containing the column indices by groups.
c igp     = pointer array of length ngrp + 1 to the locations in jgp
c           of the beginning of each group.
c ier     = error indicator.  ier = 0 if no error occurred, or 1 if
c           maxg was insufficient.
c
c incl and jdone are working arrays of length n.
c-----------------------------------------------------------------------
      integer i, j, k, kmin, kmax, ncol, ng
c
      ier = 0
      do 10 j = 1,n
 10     jdone(j) = 0
      ncol = 1
      do 60 ng = 1,maxg
        igp(ng) = ncol
        do 20 i = 1,n
 20       incl(i) = 0
        do 50 j = 1,n
c reject column j if it is already in a group.-------------------------
          if (jdone(j) .eq. 1) go to 50
          kmin = ia(j)
          kmax = ia(j+1) - 1
          do 30 k = kmin,kmax
c reject column j if it overlaps any column already in this group.------
            i = ja(k)
            if (incl(i) .eq. 1) go to 50
 30         continue
c accept column j into group ng.---------------------------------------
          jgp(ncol) = j
          ncol = ncol + 1
          jdone(j) = 1
          do 40 k = kmin,kmax
            i = ja(k)
 40         incl(i) = 1
 50       continue
c stop if this group is empty (grouping is complete).-------------------
        if (ncol .eq. igp(ng)) go to 70
 60     continue
c error return if not all columns were chosen (maxg too small).---------
      if (ncol .le. n) go to 80
      ng = maxg
 70   ngrp = ng - 1
      return
 80   ier = 1
      return
c----------------------- end of subroutine jgroup ----------------------
      end
      subroutine adjlr (n, isp, ldif)
      integer n, isp, ldif
      dimension isp(1)
```

```
c-----------------------------------------------------------------------
c this routine computes an adjustment, ldif, to the required
c integer storage space in iwk (sparse matrix work space).
c it is called only if the word length ratio is lrat = 1.
c this is to account for the possibility that the symbolic lu phase
c may require more storage than the numerical lu and solution phases.
c-----------------------------------------------------------------------
      integer ip, jlmax, jumax, lnfc, lsfc, nzlu
c
      ip = 2*n + 1
c get jlmax = ijl(n) and jumax = iju(n) (sizes of jl and ju). ----------
      jlmax = isp(ip)
      jumax = isp(ip+ip)
c nzlu = (size of l) + (size of u) = (il(n+1)-il(1)) + (iu(n+1)-iu(1)).
      nzlu = isp(n+1) - isp(1) + isp(ip+n+1) - isp(ip+1)
      lsfc = 12*n + 3 + 2*max0(jlmax,jumax)
      lnfc = 9*n + 2 + jlmax + jumax + nzlu
      ldif = max0(0, lsfc - lnfc)
      return
c----------------------- end of subroutine adjlr -----------------------
      end
      subroutine cntnzu (n, ia, ja, nzsut)
      integer n, ia, ja, nzsut
      dimension ia(1), ja(1)
c-----------------------------------------------------------------------
c this routine counts the number of nonzero elements in the strict
c upper triangle of the matrix m + m(transpose), where the sparsity
c structure of m is given by pointer arrays ia and ja.
c this is needed to compute the storage requirements for the
c sparse matrix reordering operation in odrv.
c-----------------------------------------------------------------------
      integer ii, jj, j, jmin, jmax, k, kmin, kmax, num
c
      num = 0
      do 50 ii = 1,n
        jmin = ia(ii)
        jmax = ia(ii+1) - 1
        if (jmin .gt. jmax) go to 50
        do 40 j = jmin,jmax
          if (ja(j) - ii) 10, 40, 30
 10       jj =ja(j)
          kmin = ia(jj)
          kmax = ia(jj+1) - 1
          if (kmin .gt. kmax) go to 30
          do 20 k = kmin,kmax
            if (ja(k) .eq. ii) go to 40
 20         continue
 30       num = num + 1
 40       continue
 50     continue
      nzsut = num
      return
c----------------------- end of subroutine cntnzu ----------------------
      end
      subroutine intdy (t, k, yh, nyh, dky, iflag)
c111. optimize
      integer k, nyh, iflag
      integer iownd, iowns,
     1   icf, ierpj, iersl, jcur, jstart, kflag, l, meth, miter,
```

```
     2   maxord, maxcor, msbp, mxncf, n, nq, nst, nfe, nje, nqu
      integer i, ic, j, jb, jb2, jj, jj1, jp1
      double precision t, yh, dky
      double precision rowns,
     1   ccmax, el0, h, hmin, hmxi, hu, rc, tn, uround
      double precision c, r, s, tp
      dimension yh(nyh,1), dky(1)
      common /ls0001/ rowns(209),
     2   ccmax, el0, h, hmin, hmxi, hu, rc, tn, uround,
     3   iownd(14), iowns(6),
     4   icf, ierpj, iersl, jcur, jstart, kflag, l, meth, miter,
     5   maxord, maxcor, msbp, mxncf, n, nq, nst, nfe, nje, nqu
c-----------------------------------------------------------------------
c intdy computes interpolated values of the k-th derivative of the
c dependent variable vector y, and stores it in dky.  this routine
c is called within the package with k = 0 and t = tout, but may
c also be called by the user for any k up to the current order.
c (see detailed instructions in the usage documentation.)
c-----------------------------------------------------------------------
c the computed values in dky are gotten by interpolation using the
c nordsieck history array yh.  this array corresponds uniquely to a
c vector-valued polynomial of degree nqcur or less, and dky is set
c to the k-th derivative of this polynomial at t.
c the formula for dky is..
c               q
c  dky(i)  =  sum  c(j,k) * (t - tn)**(j-k) * h**(-j) * yh(i,j+1)
c              j=k
c where  c(j,k) = j*(j-1)*...*(j-k+1), q = nqcur, tn = tcur, h = hcur.
c the quantities  nq = nqcur, l = nq+1, n = neq, tn, and h are
c communicated by common.  the above sum is done in reverse order.
c iflag is returned negative if either k or t is out of bounds.
c-----------------------------------------------------------------------
      iflag = 0
      if (k .lt. 0 .or. k .gt. nq) go to 80
      tp = tn - hu -  100.0d0*uround*(tn + hu)
      if ((t-tp)*(t-tn) .gt. 0.0d0) go to 90
c
      s = (t - tn)/h
      ic = 1
      if (k .eq. 0) go to 15
      jj1 = l - k
      do 10 jj = jj1,nq
 10     ic = ic*jj
 15   c = dfloat(ic)
      do 20 i = 1,n
 20     dky(i) = c*yh(i,l)
      if (k .eq. nq) go to 55
      jb2 = nq - k
      do 50 jb = 1,jb2
        j = nq - jb
        jp1 = j + 1
        ic = 1
        if (k .eq. 0) go to 35
        jj1 = jp1 - k
        do 30 jj = jj1,j
 30       ic = ic*jj
 35     c = dfloat(ic)
        do 40 i = 1,n
 40       dky(i) = c*yh(i,jp1) + s*dky(i)
```

```
 50     continue
      if (k .eq. 0) return
 55   r = h**(-k)
      do 60 i = 1,n
 60     dky(i) = r*dky(i)
      return
c

CCC 80   call xerrwv('intdy--  k (=i1) illegal       ',
CCC     1   30, 51, 0, 1, k, 0, 0, 0.0d0, 0.0d0)
80    iflag = -1
      return

CCC 90   call xerrwv('intdy--  t (=r1) illegal       ',
CCC     1   30, 52, 0, 0, 0, 0, 1, t, 0.0d0)
CCC      call xerrwv(
CCC     1  '       t not in interval tcur - hu (= r1) to tcur (=r2)       ',
CCC     1   60, 52, 0, 0, 0, 0, 2, tp, tn)
90    iflag = -2
      return
c----------------------- end of subroutine intdy -----------------------
      end
      subroutine stode (neq, y, yh, nyh, yh1, ewt, savf, acor,
     1   wm, iwm, f, jac, pjac, slvs)
clll. optimize
      external f, jac, pjac, slvs
      integer neq, nyh, iwm
      integer iownd, ialth, ipup, lmax, meo, nqnyh, nslp,
     1   icf, ierpj, iersl, jcur, jstart, kflag, l, meth, miter,
     2   maxord, maxcor, msbp, mxncf, n, nq, nst, nfe, nje, nqu
      integer i, i1, iredo, iret, j, jb, m, ncf, newq
      double precision y, yh, yh1, ewt, savf, acor, wm
      double precision conit, crate, el, elco, hold, rmax, tesco,
     2   ccmax, el0, h, hmin, hmxi, hu, rc, tn, uround
      double precision dcon, ddn, del, delp, dsm, dup, exdn, exsm, exup,
     1   r, rh, rhdn, rhsm, rhup, told, vnorm
      dimension neq(1), y(1), yh(nyh,1), yh1(1), ewt(1), savf(1),
     1   acor(1), wm(1), iwm(1)
      common /ls0001/ conit, crate, el(13), elco(13,12),
     1   hold, rmax, tesco(3,12),
     2   ccmax, el0, h, hmin, hmxi, hu, rc, tn, uround, iownd(14),
     3   ialth, ipup, lmax, meo, nqnyh, nslp,
     4   icf, ierpj, iersl, jcur, jstart, kflag, l, meth, miter,
     5   maxord, maxcor, msbp, mxncf, n, nq, nst, nfe, nje, nqu
c-----------------------------------------------------------------------
c stode performs one step of the integration of an initial value
c problem for a system of ordinary differential equations.
c note.. stode is independent of the value of the iteration method
c indicator miter, when this is .ne. 0, and hence is independent
c of the type of chord method used, or the jacobian structure.
c communication with stode is done with the following variables..
c
c neq    = integer array containing problem size in neq(1), and
c          passed as the neq argument in all calls to f and jac.
c y      = an array of length .ge. n used as the y argument in
c          all calls to f and jac.
c yh     = an nyh by lmax array containing the dependent variables
c          and their approximate scaled derivatives, where
c          lmax = maxord + 1.  yh(i,j+1) contains the approximate
```

```
c          j-th derivative of y(i), scaled by h**j/factorial(j)
c          (j = 0,1,...,nq).  on entry for the first step, the first
c          two columns of yh must be set from the initial values.
c nyh    = a constant integer .ge. n, the first dimension of yh.
c yh1    = a one-dimensional array occupying the same space as yh.
c ewt    = an array of length n containing multiplicative weights
c          for local error measurements.  local errors in y(i) are
c          compared to 1.0/ewt(i) in various error tests.
c savf   = an array of working storage, of length n.
c          also used for input of yh(*,maxord+2) when jstart = -1
c          and maxord .lt. the current order nq.
c acor   = a work array of length n, used for the accumulated
c          corrections.  on a successful return, acor(i) contains
c          the estimated one-step local error in y(i).
c wm,iwm = real and integer work arrays associated with matrix
c          operations in chord iteration (miter .ne. 0).
c pjac   = name of routine to evaluate and preprocess jacobian matrix
c          and p = i - h*el0*jac, if a chord method is being used.
c slvs   = name of routine to solve linear system in chord iteration.
c ccmax  = maximum relative change in h*el0 before pjac is called.
c h      = the step size to be attempted on the next step.
c          h is altered by the error control algorithm during the
c          problem.  h can be either positive or negative, but its
c          sign must remain constant throughout the problem.
c hmin   = the minimum absolute value of the step size h to be used.
c hmxi   = inverse of the maximum absolute value of h to be used.
c          hmxi = 0.0 is allowed and corresponds to an infinite hmax.
c          hmin and hmxi may be changed at any time, but will not
c          take effect until the next change of h is considered.
c tn     = the independent variable. tn is updated on each step taken.
c jstart = an integer used for input only, with the following
c          values and meanings..
c               0  perform the first step.
c            .gt.0  take a new step continuing from the last.
c              -1  take the next step with a new value of h, maxord,
c                    n, meth, miter, and/or matrix parameters.
c              -2  take the next step with a new value of h,
c                    but with other inputs unchanged.
c          on return, jstart is set to 1 to facilitate continuation.
c kflag  = a completion code with the following meanings..
c               0  the step was succesful.
c              -1  the requested error could not be achieved.
c              -2  corrector convergence could not be achieved.
c              -3  fatal error in pjac or slvs.
c          a return with kflag = -1 or -2 means either
c          abs(h) = hmin or 10 consecutive failures occurred.
c          on a return with kflag negative, the values of tn and
c          the yh array are as of the beginning of the last
c          step, and h is the last step size attempted.
c maxord = the maximum order of integration method to be allowed.
c maxcor = the maximum number of corrector iterations allowed.
c msbp   = maximum number of steps between pjac calls (miter .gt. 0).
c mxncf  = maximum number of convergence failures allowed.
c meth/miter = the method flags.  see description in driver.
c n      = the number of first-order differential equations.
c-----------------------------------------------------------------------
      kflag = 0
      told = tn
      ncf = 0
```

```
      ierpj = 0
      iersl = 0
      jcur = 0
      icf = 0
      delp = 0.0d0
      if (jstart .gt. 0) go to 200
      if (jstart .eq. -1) go to 100
      if (jstart .eq. -2) go to 160
c-----------------------------------------------------------------------
c on the first call, the order is set to 1, and other variables are
c initialized.  rmax is the maximum ratio by which h can be increased
c in a single step.  it is initially 1.e4 to compensate for the small
c initial h, but then is normally equal to 10.  if a failure
c occurs (in corrector convergence or error test), rmax is set at 2
c for the next increase.
c-----------------------------------------------------------------------
      lmax = maxord + 1
      nq = 1
      l = 2
      ialth = 2
      rmax = 10000.0d0
      rc = 0.0d0
      el0 = 1.0d0
      crate = 0.7d0
      hold = h
      meo = meth
      nslp = 0
      ipup = miter
      iret = 3
      go to 140
c-----------------------------------------------------------------------
c the following block handles preliminaries needed when jstart = -1.
c ipup is set to miter to force a matrix update.
c if an order increase is about to be considered (ialth = 1),
c ialth is reset to 2 to postpone consideration one more step.
c if the caller has changed meth, cfode is called to reset
c the coefficients of the method.
c if the caller has changed maxord to a value less than the current
c order nq, nq is reduced to maxord, and a new h chosen accordingly.
c if h is to be changed, yh must be rescaled.
c if h or meth is being changed, ialth is reset to l = nq + 1
c to prevent further changes in h for that many steps.
c-----------------------------------------------------------------------
 100  ipup = miter
      lmax = maxord + 1
      if (ialth .eq. 1) ialth = 2
      if (meth .eq. meo) go to 110
      call cfode (meth, elco, tesco)
      meo = meth
      if (nq .gt. maxord) go to 120
      ialth = 1
      iret = 1
      go to 150
 110  if (nq .le. maxord) go to 160
 120  nq = maxord
      l = lmax
      do 125 i = 1,l
 125    el(i) = elco(i,nq)
      nqnyh = nq*nyh
```

```
      rc = rc*el(1)/el0
      el0 = el(1)
      conit = 0.5d0/dfloat(nq+2)
      ddn = vnorm (n, savf, ewt)/tesco(1,1)
      exdn = 1.0d0/dfloat(l)
      rhdn = 1.0d0/(1.3d0*ddn**exdn + 0.0000013d0)
      rh = dmin1(rhdn,1.0d0)
      iredo = 3
      if (h .eq. hold) go to 170
      rh = dmin1(rh,dabs(h/hold))
      h = hold
      go to 175
c-----------------------------------------------------------------------
c cfode is called to get all the integration coefficients for the
c current meth.  then the el vector and related constants are reset
c whenever the order nq is changed, or at the start of the problem.
c-----------------------------------------------------------------------
 140  call cfode (meth, elco, tesco)
 150  do 155 i = 1,l
 155    el(i) = elco(i,nq)
      nqnyh = nq*nyh
      rc = rc*el(1)/el0
      el0 = el(1)
      conit = 0.5d0/dfloat(nq+2)
      go to (160, 170, 200), iret
c-----------------------------------------------------------------------
c if h is being changed, the h ratio rh is checked against
c rmax, hmin, and hmxi, and the yh array rescaled.  ialth is set to
c l = nq + 1 to prevent a change of h for that many steps, unless
c forced by a convergence or error test failure.
c-----------------------------------------------------------------------
 160  if (h .eq. hold) go to 200
      rh = h/hold
      h = hold
      iredo = 3
      go to 175
 170  rh = dmax1(rh,hmin/dabs(h))
 175  rh = dmin1(rh,rmax)
      rh = rh/dmax1(1.0d0,dabs(h)*hmxi*rh)
      r = 1.0d0
      do 180 j = 2,l
        r = r*rh
        do 180 i = 1,n
 180      yh(i,j) = yh(i,j)*r
      h = h*rh
      rc = rc*rh
      ialth = l
      if (iredo .eq. 0) go to 690
c-----------------------------------------------------------------------
c this section computes the predicted values by effectively
c multiplying the yh array by the pascal triangle matrix.
c rc is the ratio of new to old values of the coefficient  h*el(1).
c when rc differs from 1 by more than ccmax, ipup is set to miter
c to force pjac to be called, if a jacobian is involved.
c in any case, pjac is called at least every msbp steps.
c-----------------------------------------------------------------------
 200  if (dabs(rc-1.0d0) .gt. ccmax) ipup = miter
      if (nst .ge. nslp+msbp) ipup = miter
      tn = tn + h
```

```
      i1 = nqnyh + 1
      do 215 jb = 1,nq
        i1 = i1 - nyh
cdir$ ivdep
        do 210 i = i1,nqnyh
 210      yh1(i) = yh1(i) + yh1(i+nyh)
 215    continue
c-----------------------------------------------------------------------
c up to maxcor corrector iterations are taken.  a convergence test is
c made on the r.m.s. norm of each correction, weighted by the error
c weight vector ewt.  the sum of the corrections is accumulated in the
c vector acor(i).  the yh array is not altered in the corrector loop.
c-----------------------------------------------------------------------
 220  m = 0
      do 230 i = 1,n
 230    y(i) = yh(i,1)
      call f (neq, tn, y, savf)
      nfe = nfe + 1
      if (ipup .le. 0) go to 250
c-----------------------------------------------------------------------
c if indicated, the matrix p = i - h*el(1)*j is reevaluated and
c preprocessed before starting the corrector iteration.  ipup is set
c to 0 as an indicator that this has been done.
c-----------------------------------------------------------------------
      call pjac (neq, y, yh, nyh, ewt, acor, savf, wm, iwm, f, jac)
      ipup = 0
      rc = 1.0d0
      nslp = nst
      crate = 0.7d0
      if (ierpj .ne. 0) go to 430
 250  do 260 i = 1,n
 260    acor(i) = 0.0d0
 270  if (miter .ne. 0) go to 350
c-----------------------------------------------------------------------
c in the case of functional iteration, update y directly from
c the result of the last function evaluation.
c-----------------------------------------------------------------------
      do 290 i = 1,n
        savf(i) = h*savf(i) - yh(i,2)
 290    y(i) = savf(i) - acor(i)
      del = vnorm (n, y, ewt)
      do 300 i = 1,n
        y(i) = yh(i,1) + el(1)*savf(i)
 300    acor(i) = savf(i)
      go to 400
c-----------------------------------------------------------------------
c in the case of the chord method, compute the corrector error,
c and solve the linear system with that as right-hand side and
c p as coefficient matrix.
c-----------------------------------------------------------------------
 350  do 360 i = 1,n
 360    y(i) = h*savf(i) - (yh(i,2) + acor(i))
      call slvs (wm, iwm, y, savf)
      if (iersl .lt. 0) go to 430
      if (iersl .gt. 0) go to 410
      del = vnorm (n, y, ewt)
      do 380 i = 1,n
        acor(i) = acor(i) + y(i)
 380    y(i) = yh(i,1) + el(1)*acor(i)
```

```
c-----------------------------------------------------------------------
c test for convergence.  if m.gt.0, an estimate of the convergence
c rate constant is stored in crate, and this is used in the test.
c-----------------------------------------------------------------------
 400  if (m .ne. 0) crate = dmax1(0.2d0*crate,del/delp)
      dcon = del*dmin1(1.0d0,1.5d0*crate)/(tesco(2,nq)*conit)
      if (dcon .le. 1.0d0) go to 450
      m = m + 1
      if (m .eq. maxcor) go to 410
      if (m .ge. 2 .and. del .gt. 2.0d0*delp) go to 410
      delp = del
      call f (neq, tn, y, savf)
      nfe = nfe + 1
      go to 270
c-----------------------------------------------------------------------
c the corrector iteration failed to converge.
c if miter .ne. 0 and the jacobian is out of date, pjac is called for
c the next try.  otherwise the yh array is retracted to its values
c before prediction, and h is reduced, if possible.  if h cannot be
c reduced or mxncf failures have occurred, exit with kflag = -2.
c-----------------------------------------------------------------------
 410  if (miter .eq. 0 .or. jcur .eq. 1) go to 430
      icf = 1
      ipup = miter
      go to 220
 430  icf = 2
      ncf = ncf + 1
      rmax = 2.0d0
      tn = told
      i1 = nqnyh + 1
      do 445 jb = 1,nq
        i1 = i1 - nyh
cdir$ ivdep
        do 440 i = i1,nqnyh
 440      yh1(i) = yh1(i) - yh1(i+nyh)
 445    continue
      if (ierpj .lt. 0 .or. iersl .lt. 0) go to 680
      if (dabs(h) .le. hmin*1.00001d0) go to 670
      if (ncf .eq. mxncf) go to 670
      rh = 0.25d0
      ipup = miter
      iredo = 1
      go to 170
c-----------------------------------------------------------------------
c the corrector has converged.  jcur is set to 0
c to signal that the jacobian involved may need updating later.
c the local error test is made and control passes to statement 500
c if it fails.
c-----------------------------------------------------------------------
 450  jcur = 0
      if (m .eq. 0) dsm = del/tesco(2,nq)
      if (m .gt. 0) dsm = vnorm (n, acor, ewt)/tesco(2,nq)
      if (dsm .gt. 1.0d0) go to 500
c-----------------------------------------------------------------------
c after a successful step, update the yh array.
c consider changing h if ialth = 1.  otherwise decrease ialth by 1.
c if ialth is then 1 and nq .lt. maxord, then acor is saved for
c use in a possible order increase on the next step.
c if a change in h is considered, an increase or decrease in order
```

```
c by one is considered also.  a change in h is made only if it is by a
c factor of at least 1.1.  if not, ialth is set to 3 to prevent
c testing for that many steps.
c-----------------------------------------------------------------------
      kflag = 0
      iredo = 0
      nst = nst + 1
      hu = h
      nqu = nq
      do 470 j = 1,l
        do 470 i = 1,n
 470      yh(i,j) = yh(i,j) + el(j)*acor(i)
      ialth = ialth - 1
      if (ialth .eq. 0) go to 520
      if (ialth .gt. 1) go to 700
      if (l .eq. lmax) go to 700
      do 490 i = 1,n
 490    yh(i,lmax) = acor(i)
      go to 700
c-----------------------------------------------------------------------
c the error test failed.  kflag keeps track of multiple failures.
c restore tn and the yh array to their previous values, and prepare
c to try the step again.  compute the optimum step size for this or
c one lower order.  after 2 or more failures, h is forced to decrease
c by a factor of 0.2 or less.
c-----------------------------------------------------------------------
 500  kflag = kflag - 1
      tn = told
      i1 = nqnyh + 1
      do 515 jb = 1,nq
        i1 = i1 - nyh
cdir$ ivdep
        do 510 i = i1,nqnyh
 510      yh1(i) = yh1(i) - yh1(i+nyh)
 515    continue
      rmax = 2.0d0
      if (dabs(h) .le. hmin*1.00001d0) go to 660
      if (kflag .le. -3) go to 640
      iredo = 2
      rhup = 0.0d0
      go to 540
c-----------------------------------------------------------------------
c regardless of the success or failure of the step, factors
c rhdn, rhsm, and rhup are computed, by which h could be multiplied
c at order nq - 1, order nq, or order nq + 1, respectively.
c in the case of failure, rhup = 0.0 to avoid an order increase.
c the largest of these is determined and the new order chosen
c accordingly.  if the order is to be increased, we compute one
c additional scaled derivative.
c-----------------------------------------------------------------------
 520  rhup = 0.0d0
      if (l .eq. lmax) go to 540
      do 530 i = 1,n
 530    savf(i) = acor(i) - yh(i,lmax)
      dup = vnorm (n, savf, ewt)/tesco(3,nq)
      exup = 1.0d0/dfloat(l+1)
      rhup = 1.0d0/(1.4d0*dup**exup + 0.0000014d0)
 540  exsm = 1.0d0/dfloat(l)
      rhsm = 1.0d0/(1.2d0*dsm**exsm + 0.0000012d0)
```

```
      rhdn = 0.0d0
      if (nq .eq. 1) go to 560
      ddn = vnorm (n, yh(1,1), ewt)/tesco(1,nq)
      exdn = 1.0d0/dfloat(nq)
      rhdn = 1.0d0/(1.3d0*ddn**exdn + 0.0000013d0)
 560  if (rhsm .ge. rhup) go to 570
      if (rhup .gt. rhdn) go to 590
      go to 580
 570  if (rhsm .lt. rhdn) go to 580
      newq = nq
      rh = rhsm
      go to 620
 580  newq = nq - 1
      rh = rhdn
      if (kflag .lt. 0 .and. rh .gt. 1.0d0) rh = 1.0d0
      go to 620
 590  newq = 1
      rh = rhup
      if (rh .lt. 1.1d0) go to 610
      r = el(1)/dfloat(1)
      do 600 i = 1,n
 600    yh(i,newq+1) = acor(i)*r
      go to 630
 610  ialth = 3
      go to 700
 620  if ((kflag .eq. 0) .and. (rh .lt. 1.1d0)) go to 610
      if (kflag .le. -2) rh = dmin1(rh,0.2d0)
c-----------------------------------------------------------------------
c if there is a change of order, reset nq, l, and the coefficients.
c in any case h is reset according to rh and the yh array is rescaled.
c then exit from 690 if the step was ok, or redo the step otherwise.
c-----------------------------------------------------------------------
      if (newq .eq. nq) go to 170
 630  nq = newq
      l = nq + 1
      iret = 2
      go to 150
c-----------------------------------------------------------------------
c control reaches this section if 3 or more failures have occured.
c if 10 failures have occurred, exit with kflag = -1.
c it is assumed that the derivatives that have accumulated in the
c yh array have errors of the wrong order.  hence the first
c derivative is recomputed, and the order is set to 1.  then
c h is reduced by a factor of 10, and the step is retried,
c until it succeeds or h reaches hmin.
c-----------------------------------------------------------------------
 640  if (kflag .eq. -10) go to 660
      rh = 0.1d0
      rh = dmax1(hmin/dabs(h),rh)
      h = h*rh
      do 645 i = 1,n
 645    y(i) = yh(i,1)
      call f (neq, tn, y, savf)
      nfe = nfe + 1
      do 650 i = 1,n
 650    yh(i,2) = h*savf(i)
      ipup = miter
      ialth = 5
      if (nq .eq. 1) go to 200
```

```
      nq = 1
      l = 2
      iret = 3
      go to 150
c-----------------------------------------------------------------------
c all returns are made through this section.  h is saved in hold
c to allow the caller to change h on the next step.
c-----------------------------------------------------------------------
 660  kflag = -1
      go to 720
 670  kflag = -2
      go to 720
 680  kflag = -3
      go to 720
 690  rmax = 10.0d0
 700  r = 1.0d0/tesco(2,nqu)
      do 710 i = 1,n
 710    acor(i) = acor(i)*r
 720  hold = h
      jstart = 1
      return
c----------------------- end of subroutine stode -----------------------
      end
      subroutine cfode (meth, elco, tesco)
c111. optimize
      integer meth
      integer i, ib, nq, nqm1, nqp1
      double precision elco, tesco
      double precision agamq, fnq, fnqm1, pc, pint, ragq,
     1   rqfac, rq1fac, tsign, xpin
      dimension elco(13,12), tesco(3,12)
c-----------------------------------------------------------------------
c cfode is called by the integrator routine to set coefficients
c needed there.  the coefficients for the current method, as
c given by the value of meth, are set for all orders and saved.
c the maximum order assumed here is 12 if meth = 1 and 5 if meth = 2.
c (a smaller value of the maximum order is also allowed.)
c cfode is called once at the beginning of the problem,
c and is not called again unless and until meth is changed.
c
c the elco array contains the basic method coefficients.
c the coefficients el(i), 1 .le. i .le. nq+1, for the method of
c order nq are stored in elco(i,nq).  they are given by a genetrating
c polynomial, i.e.,
c     l(x) = el(1) + el(2)*x + ... + el(nq+1)*x**nq.
c for the implicit adams methods, l(x) is given by
c     dl/dx = (x+1)*(x+2)*...*(x+nq-1)/factorial(nq-1),    l(-1) = 0.
c for the bdf methods, l(x) is given by
c     l(x) = (x+1)*(x+2)* ... *(x+nq)/k,
c where         k = factorial(nq)*(1 + 1/2 + ... + 1/nq).
c
c the tesco array contains test constants used for the
c local error test and the selection of step size and/or order.
c at order nq, tesco(k,nq) is used for the selection of step
c size at order nq - 1 if k = 1, at order nq if k = 2, and at order
c nq + 1 if k = 3.
c-----------------------------------------------------------------------
      dimension pc(12)
c
```

```
      go to (100, 200), meth
c
 100  elco(1,1) = 1.0d0
      elco(2,1) = 1.0d0
      tesco(1,1) = 0.0d0
      tesco(2,1) = 2.0d0
      tesco(1,2) = 1.0d0
      tesco(3,12) = 0.0d0
      pc(1) = 1.0d0
      rqfac = 1.0d0
      do 140 nq = 2,12
c----------------------------------------------------------------------
c the pc array will contain the coefficients of the polynomial
c     p(x) = (x+1)*(x+2)*...*(x+nq-1).
c initially, p(x) = 1.
c----------------------------------------------------------------------
        rq1fac = rqfac
        rqfac = rqfac/dfloat(nq)
        nqm1 = nq - 1
        fnqm1 = dfloat(nqm1)
        nqp1 = nq + 1
c form coefficients of p(x)*(x+nq-1). ---------------------------------
        pc(nq) = 0.0d0
        do 110 ib = 1,nqm1
          i = nqp1 - ib
 110      pc(i) = pc(i-1) + fnqm1*pc(i)
        pc(1) = fnqm1*pc(1)
c compute integral, -1 to 0, of p(x) and x*p(x). ----------------------
        pint = pc(1)
        xpin = pc(1)/2.0d0
        tsign = 1.0d0
        do 120 i = 2,nq
          tsign = -tsign
          pint = pint + tsign*pc(i)/dfloat(i)
 120      xpin = xpin + tsign*pc(i)/dfloat(i+1)
c store coefficients in elco and tesco. -------------------------------
        elco(1,nq) = pint*rq1fac
        elco(2,nq) = 1.0d0
        do 130 i = 2,nq
 130      elco(i+1,nq) = rq1fac*pc(i)/dfloat(i)
        agamq = rqfac*xpin
        ragq = 1.0d0/agamq
        tesco(2,nq) = ragq
        if (nq .lt. 12) tesco(1,nqp1) = ragq*rqfac/dfloat(nqp1)
        tesco(3,nqm1) = ragq
 140    continue
      return
c
 200  pc(1) = 1.0d0
      rq1fac = 1.0d0
      do 230 nq = 1,5
c----------------------------------------------------------------------
c the pc array will contain the coefficients of the polynomial
c     p(x) = (x+1)*(x+2)*...*(x+nq).
c initially, p(x) = 1.
c----------------------------------------------------------------------
        fnq = dfloat(nq)
        nqp1 = nq + 1
c form coefficients of p(x)*(x+nq). -----------------------------------
```

```
      pc(nqp1) = 0.0d0
      do 210 ib = 1,nq
        i = nq + 2 - ib
210     pc(i) = pc(i-1) + fnq*pc(i)
      pc(1) = fnq*pc(1)
c store coefficients in elco and tesco. -------------------------------
      do 220 i = 1,nqp1
220     elco(i,nq) = pc(i)/pc(2)
      elco(2,nq) = 1.0d0
      tesco(1,nq) = rq1fac
      tesco(2,nq) = dfloat(nqp1)/elco(1,nq)
      tesco(3,nq) = dfloat(nq+2)/elco(1,nq)
      rq1fac = rq1fac/fnq
230   continue
     return
c----------------------- end of subroutine cfode -----------------------
     end
     subroutine prjs (neq,y,yh,nyh,ewt,ftem,savf,wk,iwk,f,jac)
c111. optimize
     external f,jac
     integer neq, nyh, iwk
     integer iownd, iowns,
    1   icf, ierpj, iersl, jcur, jstart, kflag, l, meth, miter,
    2   maxord, maxcor, msbp, mxncf, n, nq, nst, nfe, nje, nqu
     integer iplost, iesp, istatc, iys, iba, ibian, ibjan, ibjgp,
    1   ipian, ipjan, ipjgp, ipigp, ipr, ipc, ipic, ipisp, iprsp, ipa,
    2   lenyh, lenyhm, lenwk, lreq, lrat, lrest, lwmin, moss, msbj,
    3   nslj, ngp, nlu, nnz, nsp, nzl, nzu
     integer i, imul, j, jj, jok, jmax, jmin, k, kmax, kmin, ng
     double precision y, yh, ewt, ftem, savf, wk
     double precision rowns,
    1   ccmax, el0, h, hmin, hmxi, hu, rc, tn, uround
     double precision con0, conmin, ccmxj, psmall, rbig, seth
     double precision con, di, fac, hl0, pij, r, r0, rcon, rcont,
    1   srur, vnorm
     dimension neq(1), y(1), yh(nyh,1), ewt(1), ftem(1), savf(1),
    1   wk(1), iwk(1)
     common /ls0001/ rowns(209),
    2   ccmax, el0, h, hmin, hmxi, hu, rc, tn, uround,
    3   iownd(14), iowns(6),
    4   icf, ierpj, iersl, jcur, jstart, kflag, l, meth, miter,
    5   maxord, maxcor, msbp, mxncf, n, nq, nst, nfe, nje, nqu
     common /lss001/ con0, conmin, ccmxj, psmall, rbig, seth,
    1   iplost, iesp, istatc, iys, iba, ibian, ibjan, ibjgp,
    2   ipian, ipjan, ipjgp, ipigp, ipr, ipc, ipic, ipisp, iprsp, ipa,
    3   lenyh, lenyhm, lenwk, lreq, lrat, lrest, lwmin, moss, msbj,
    4   nslj, ngp, nlu, nnz, nsp, nzl, nzu
c-----------------------------------------------------------------------
c prjs is called to compute and process the matrix
c p = i - h*el(1)*j , where j is an approximation to the jacobian.
c j is computed by columns, either by the user-supplied routine jac
c if miter = 1, or by finite differencing if miter = 2.
c if miter = 3, a diagonal approximation to j is used.
c if miter = 1 or 2, and if the existing value of the jacobian
c (as contained in p) is considered acceptable, then a new value of
c p is reconstructed from the old value.  in any case, when miter
c is 1 or 2, the p matrix is subjected to lu decomposition in cdrv.
c p and its lu decomposition are stored (separately) in wk.
c
```

```
c in addition to variables described previously, communication
c with prjs uses the following..
c y     = array containing predicted values on entry.
c ftem  = work array of length n (acor in stode).
c savf  = array containing f evaluated at predicted y.
c wk    = real work space for matrices.  on output it contains the
c         inverse diagonal matrix if miter = 3, and p and its sparse
c         lu decomposition if miter is 1 or 2.
c         storage of matrix elements starts at wk(3).
c         wk also contains the following matrix-related data..
c         wk(1) = sqrt(uround), used in numerical jacobian increments.
c         wk(2) = h*el0, saved for later use if miter = 3.
c iwk   = integer work space for matrix-related data, assumed to
c         be equivalenced to wk.  in addition, wk(iprsp) and iwk(ipisp)
c         are assumed to have identical locations.
c el0   = el(1) (input).
c ierpj = output error flag (in common).
c       = 0 if no error.
c       = 1  if zero pivot found in cdrv.
c       = 2  if a singular matrix arose with miter = 3.
c       = -1 if insufficient storage for cdrv (should not occur here).
c       = -2 if other error found in cdrv (should not occur here).
c jcur  = output flag = 1 to indicate that the jacobian matrix
c         (or approximation) is now current.
c this routine also uses other variables in common.
c-----------------------------------------------------------------------
      hl0 = h*el0
      con = -hl0
      if (miter .eq. 3) go to 300
c see whether j should be reevaluated (jok = 0) or not (jok = 1). ------
      jok = 1
      if (nst .eq. 0 .or. nst .ge. nslj+msbj) jok = 0
      if (icf .eq. 1 .and. dabs(rc - 1.0d0) .lt. ccmxj) jok = 0
      if (icf .eq. 2) jok = 0
      if (jok .eq. 1) go to 250
c
c miter = 1 or 2, and the jacobian is to be reevaluated. ---------------
 20   jcur = 1
      nje = nje + 1
      nslj = nst
      iplost = 0
      conmin = dabs(con)
      go to (100, 200), miter
c
c if miter = 1, call jac, multiply by scalar, and add identity. --------
 100  continue
      kmin = iwk(ipian)
      do 130 j = 1, n
        kmax = iwk(ipian+j) - 1
        do 110 i = 1,n
 110      ftem(i) = 0.0d0
        call jac (neq, tn, y, j, iwk(ipian), iwk(ipjan), ftem)
        do 120 k = kmin, kmax
          i = iwk(ibjan+k)
          wk(iba+k) = ftem(i)*con
          if (i .eq. j) wk(iba+k) = wk(iba+k) + 1.0d0
 120      continue
        kmin = kmax + 1
 130    continue
```

```
      go to 290
c
c if miter = 2, make ngp calls to f to approximate j and p. ------------
 200  continue
      fac = vnorm(n, savf, ewt)
      r0 = 1000.0d0 * dabs(h) * uround * dfloat(n) * fac
      if (r0 .eq. 0.0d0) r0 = 1.0d0
      srur = wk(1)
      jmin = iwk(ipigp)
      do 240 ng = 1,ngp
        jmax = iwk(ipigp+ng) - 1
        do 210 j = jmin,jmax
          jj = iwk(ibjgp+j)
          r = dmax1(srur*dabs(y(jj)),r0/ewt(jj))
 210      y(jj) = y(jj) + r
        call f (neq, tn, y, ftem)
        do 230 j = jmin,jmax
          jj = iwk(ibjgp+j)
          y(jj) = yh(jj,1)
          r = dmax1(srur*dabs(y(jj)),r0/ewt(jj))
          fac = -hl0/r
          kmin =iwk(ibian+jj)
          kmax =iwk(ibian+jj+1) - 1
          do 220 k = kmin,kmax
            i = iwk(ibjan+k)
            wk(iba+k) = (ftem(i) - savf(i))*fac
            if (i .eq. jj) wk(iba+k) = wk(iba+k) + 1.0d0
 220        continue
 230      continue
        jmin = jmax + 1
 240    continue
      nfe = nfe + ngp
      go to 290
c
c if jok = 1, reconstruct new p from old p. ---------------------------
 250  jcur = 0
      rcon = con/con0
      rcont = dabs(con)/conmin
      if (rcont .gt. rbig .and. iplost .eq. 1) go to 20
      kmin = iwk(ipian)
      do 275 j = 1,n
        kmax = iwk(ipian+j) - 1
        do 270 k = kmin,kmax
          i = iwk(ibjan+k)
          pij = wk(iba+k)
          if (i .ne. j) go to 260
          pij = pij - 1.0d0
          if (dabs(pij) .ge. psmall) go to 260
            iplost = 1
            conmin = dmin1(dabs(con0),conmin)
 260      pij = pij*rcon
          if (i .eq. j) pij = pij + 1.0d0
          wk(iba+k) = pij
 270      continue
        kmin = kmax + 1
 275    continue
c
c do numerical factorization of p matrix. ----------------------------
 290  nlu = nlu + 1
```

```
      con0 = con
      ierpj = 0
      do 295 i = 1,n
295     ftem(i) = 0.0d0
      call cdrv (n,iwk(ipr),iwk(ipc),iwk(ipic),iwk(ipian),iwk(ipjan),
     1   wk(ipa),ftem,ftem,nsp,iwk(ipisp),wk(iprsp),iesp,2,iys)
      if (iys .eq. 0) return
      imul = (iys - 1)/n
      ierpj = -2
      if (imul .eq. 8) ierpj = 1
      if (imul .eq. 10) ierpj = -1
      return
c
c if miter = 3, construct a diagonal approximation to j and p. ---------
 300  continue
      jcur = 1
      nje = nje + 1
      wk(2) = hl0
      ierpj = 0
      r = el0*0.1d0
      do 310 i = 1,n
 310    y(i) = y(i) + r*(h*savf(i) - yh(i,2))
      call f (neq, tn, y, wk(3))
      nfe = nfe + 1
      do 320 i = 1,n
        r0 = h*savf(i) - yh(i,2)
        di = 0.1d0*r0 - h*(wk(i+2) - savf(i))
        wk(i+2) = 1.0d0
        if (dabs(r0) .lt. uround/ewt(i)) go to 320
        if (dabs(di) .eq. 0.0d0) go to 330
        wk(i+2) = 0.1d0*r0/di
 320    continue
      return
 330  ierpj = 2
      return
c----------------------- end of subroutine prjs -----------------------
      end
      subroutine slss (wk, iwk, x, tem)
cl11. optimize
      integer iwk
      integer iownd, iowns,
     1   icf, ierpj, iersl, jcur, jstart, kflag, l, meth, miter,
     2   maxord, maxcor, msbp, mxncf, n, nq, nst, nfe, nje, nqu
      integer iplost, iesp, istatc, iys, iba, ibian, ibjan, ibjgp,
     1   ipian, ipjan, ipjgp, ipigp, ipr, ipc, ipic, ipisp, iprsp, ipa,
     2   lenyh, lenyhm, lenwk, lreq, lrat, lrest, lwmin, moss, msbj,
     3   nslj, ngp, nlu, nnz, nsp, nzl, nzu
      integer i
      double precision wk, x, tem
      double precision rowns,
     1   ccmax, el0, h, hmin, hmxi, hu, rc, tn, uround
      double precision rlss
      double precision di, hl0, phl0, r
      dimension wk(1), iwk(1), x(1), tem(1)
      common /ls0001/ rowns(209),
     2   ccmax, el0, h, hmin, hmxi, hu, rc, tn, uround,
     3   iownd(14), iowns(6),
     4   icf, ierpj, iersl, jcur, jstart, kflag, l, meth, miter,
     5   maxord, maxcor, msbp, mxncf, n, nq, nst, nfe, nje, nqu
```

```
      common /lss001/ rlss(6),
     1   iplost, iesp, istatc, iys, iba, ibian, ibjan, ibjgp,
     2   ipian, ipjan, ipjgp, ipigp, ipr, ipc, ipic, ipisp, iprsp, ipa,
     3   lenyh, lenyhm, lenwk, lreq, lrat, lrest, lwmin, moss, msbj,
     4   nslj, ngp, nlu, nnz, nsp, nzl, nzu
c-----------------------------------------------------------------------
c this routine manages the solution of the linear system arising from
c a chord iteration.  it is called if miter .ne. 0.
c if miter is 1 or 2, it calls cdrv to accomplish this.
c if miter = 3 it updates the coefficient h*el0 in the diagonal
c matrix, and then computes the solution.
c communication with slss uses the following variables..
c wk    = real work space containing the inverse diagonal matrix if
c         miter = 3 and the lu decomposition of the matrix otherwise.
c         storage of matrix elements starts at wk(3).
c         wk also contains the following matrix-related data..
c         wk(1) = sqrt(uround) (not used here),
c         wk(2) = hl0, the previous value of h*el0, used if miter = 3.
c iwk   = integer work space for matrix-related data, assumed to
c         be equivalenced to wk.  in addition, wk(iprsp) and iwk(ipisp)
c         are assumed to have identical locations.
c x     = the right-hand side vector on input, and the solution vector
c         on output, of length n.
c tem   = vector of work space of length n, not used in this version.
c iersl = output flag (in common).
c         iersl = 0  if no trouble occurred.
c         iersl = -1 if cdrv returned an error flag (miter = 1 or 2).
c                    this should never occur and is considered fatal.
c         iersl = 1  if a singular matrix arose with miter = 3.
c this routine also uses other variables in common.
c-----------------------------------------------------------------------
      iersl = 0
      go to (100, 100, 300), miter
 100  call cdrv (n,iwk(ipr),iwk(ipc),iwk(ipic),iwk(ipian),iwk(ipjan),
     1   wk(ipa),x,x,nsp,iwk(ipisp),wk(iprsp),iesp,4,iersl)
      if (iersl .ne. 0) iersl = -1
      return
c
 300  phl0 = wk(2)
      hl0 = h*el0
      wk(2) = hl0
      if (hl0 .eq. phl0) go to 330
      r = hl0/phl0
      do 320 i = 1,n
        di = 1.0d0 - r*(1.0d0 - 1.0d0/wk(i+2))
        if (dabs(di) .eq. 0.0d0) go to 390
 320    wk(i+2) = 1.0d0/di
 330  do 340 i = 1,n
 340    x(i) = wk(i+2)*x(i)
      return
 390  iersl = 1
      return
c
c----------------------- end of subroutine slss -----------------------
      end
      subroutine ewset (n, itol, rtol, atol, ycur, ewt)
cl11. optimize
c-----------------------------------------------------------------------
c this subroutine sets the error weight vector ewt according to
```

```
c     ewt(i) = rtol(i)*abs(ycur(i)) + atol(i),  i = 1,...,n,
c with the subscript on rtol and/or atol possibly replaced by 1 above,
c depending on the value of itol.
c-----------------------------------------------------------------------
      integer n, itol
      integer i
      double precision rtol, atol, ycur, ewt
      dimension rtol(1), atol(1), ycur(n), ewt(n)
c
      go to (10, 20, 30, 40), itol
 10   continue
      do 15 i = 1,n
 15     ewt(i) = rtol(1)*dabs(ycur(i)) + atol(1)
      return
 20   continue
      do 25 i = 1,n
 25     ewt(i) = rtol(1)*dabs(ycur(i)) + atol(i)
      return
 30   continue
      do 35 i = 1,n
 35     ewt(i) = rtol(i)*dabs(ycur(i)) + atol(1)
      return
 40   continue
      do 45 i = 1,n
 45     ewt(i) = rtol(i)*dabs(ycur(i)) + atol(i)
      return
c----------------------- end of subroutine ewset -----------------------
      end
      double precision function vnorm (n, v, w)
cl11. optimize
c-----------------------------------------------------------------------
c this function routine computes the weighted root-mean-square norm
c of the vector of length n contained in the array v, with weights
c contained in the array w of length n..
c   vnorm = sqrt( (1/n) * sum( v(i)*w(i) )**2 )
c-----------------------------------------------------------------------
      integer n,   i
      double precision v, w,   sum
      dimension v(n), w(n)
      sum = 0.0d0
      do 10 i = 1,n
 10     sum = sum + (v(i)*w(i))**2
      vnorm = dsqrt(sum/dfloat(n))
      return
c----------------------- end of function vnorm -------------------------
      end
      subroutine srcms (rsav, isav, job)
c-----------------------------------------------------------------------
c this routine saves or restores (depending on job) the contents of
c the common blocks ls0001, lss001, and eh0001, which are used
c internally by one or more odepack solvers.
c
c rsav = real array of length 224 or more.
c isav = integer array of length 75 or more.
c job  = flag indicating to save or restore the common blocks..
c        job  = 1 if common is to be saved (written to rsav/isav)
c        job  = 2 if common is to be restored (read from rsav/isav)
c        a call with job = 2 presumes a prior call with job = 1.
c-----------------------------------------------------------------------
```

```
      integer isav, job
      integer ieh, ils, ilss
      integer i, lenil, leniss, lenrl, lenrss
      double precision rsav,   rls, rlss
      dimension rsav(1), isav(1)
      common /ls0001/ rls(218), ils(39)
      common /lss001/ rlss(6), ilss(34)
      common /eh0001/ ieh(2)
      data lenrl/218/, lenil/39/, lenrss/6/, leniss/34/
c
      if (job .eq. 2) go to 100
      do 10 i = 1,lenrl
 10     rsav(i) = rls(i)
      do 15 i = 1,lenrss
 15     rsav(lenrl+i) = rlss(i)
c
      do 20 i = 1,lenil
 20     isav(i) = ils(i)
      do 25 i = 1,leniss
 25     isav(lenil+i) = ilss(i)
c
      isav(lenil+leniss+1) = ieh(1)
      isav(lenil+leniss+2) = ieh(2)
      return
c
 100  continue
      do 110 i = 1,lenrl
 110     rls(i) = rsav(i)
      do 115 i = 1,lenrss
 115     rlss(i) = rsav(lenrl+i)
c
      do 120 i = 1,lenil
 120     ils(i) = isav(i)
      do 125 i = 1,leniss
 125     ilss(i) = isav(lenil+i)
c
      ieh(1) = isav(lenil+leniss+1)
      ieh(2) = isav(lenil+leniss+2)
      return
c----------------------- end of subroutine srcms -----------------------
      end
      subroutine odrv
     *     (n, ia,ja,a, p,ip, nsp,isp, path, flag)
cl11. optimize
c                                                                  5/2/83
c***********************************************************************
c  odrv -- driver for sparse matrix reordering routines
c***********************************************************************
c
c  description
c
c    odrv finds a minimum degree ordering of the rows and columns
c    of a matrix m stored in (ia,ja,a) format (see below).  for the
c    reordered matrix, the work and storage required to perform
c    gaussian elimination is (usually) significantly less.
c
c    note.. odrv and its subordinate routines have been modified to
c    compute orderings for general matrices, not necessarily having any
c    symmetry.  the miminum degree ordering is computed for the
```

```
c     structure of the symmetric matrix  m + m-transpose.
c     modifications to the original odrv module have been made in
c     the coding in subroutine mdi, and in the initial comments in
c     subroutines odrv and md.
c
c     if only the nonzero entries in the upper triangle of m are being
c     stored, then odrv symmetrically reorders (ia,ja,a), (optionally)
c     with the diagonal entries placed first in each row.  this is to
c     ensure that if m(i,j) will be in the upper triangle of m with
c     respect to the new ordering, then m(i,j) is stored in row i (and
c     thus m(j,i) is not stored),  whereas if m(i,j) will be in the
c     strict lower triangle of m, then m(j,i) is stored in row j (and
c     thus m(i,j) is not stored).
c
c
c  storage of sparse matrices
c
c     the nonzero entries of the matrix m are stored row-by-row in the
c     array a.  to identify the individual nonzero entries in each row,
c     we need to know in which column each entry lies.  these column
c     indices are stored in the array ja.  i.e., if  a(k) = m(i,j),  then
c     ja(k) = j.  to identify the individual rows, we need to know where
c     each row starts.  these row pointers are stored in the array ia.
c     i.e., if m(i,j) is the first nonzero entry (stored) in the i-th row
c     and  a(k) = m(i,j),  then  ia(i) = k.  moreover, ia(n+1) points to
c     the first location following the last element in the last row.
c     thus, the number of entries in the i-th row is  ia(i+1) - ia(i),
c     the nonzero entries in the i-th row are stored consecutively in
c
c             a(ia(i)),  a(ia(i)+1),  ..., a(ia(i+1)-1),
c
c     and the corresponding column indices are stored consecutively in
c
c             ja(ia(i)), ja(ia(i)+1), ..., ja(ia(i+1)-1).
c
c     since the coefficient matrix is symmetric, only the nonzero entries
c     in the upper triangle need be stored.  for example, the matrix
c
c               ( 1  0  2  3  0 )
c               ( 0  4  0  0  0 )
c           m = ( 2  0  5  6  0 )
c               ( 3  0  6  7  8 )
c               ( 0  0  0  8  9 )
c
c     could be stored as
c
c              - 1  2  3  4  5  6  7  8  9 10 11 12 13
c           ---+-------------------------------------
c           ia - 1  4  5  8 12 14
c           ja - 1  3  4  2  1  3  4  1  3  4  5  4  5
c            a - 1  2  3  4  2  5  6  3  6  7  8  8  9
c
c     or (symmetrically) as
c
c              - 1  2  3  4  5  6  7  8  9
c           ---+--------------------------
c           ia - 1  4  5  7  9 10
c           ja - 1  3  4  2  3  4  4  5  5
c            a - 1  2  3  4  5  6  7  8  9
```

```
c
c
c  parameters
c
c    n     - order of the matrix
c
c    ia    - integer one-dimensional array containing pointers to delimit
c            rows in ja and a.  dimension = n+1
c
c    ja    - integer one-dimensional array containing the column indices
c            corresponding to the elements of a.  dimension = number of
c            nonzero entries in (the upper triangle of) m
c
c    a     - real one-dimensional array containing the nonzero entries in
c            (the upper triangle of) m, stored by rows.  dimension =
c            number of nonzero entries in (the upper triangle of) m
c
c    p     - integer one-dimensional array used to return the permutation
c            of the rows and columns of m corresponding to the minimum
c            degree ordering.  dimension = n
c
c    ip    - integer one-dimensional array used to return the inverse of
c            the permutation returned in p.  dimension = n
c
c    nsp   - declared dimension of the one-dimensional array isp.  nsp
c            must be at least  3n+4k,  where k is the number of nonzeroes
c            in the strict upper triangle of m
c
c    isp   - integer one-dimensional array used for working storage.
c            dimension = nsp
c
c    path - integer path specification.  values and their meanings are -
c              1  find minimum degree ordering only
c              2  find minimum degree ordering and reorder symmetrically
c                   stored matrix (used when only the nonzero entries in
c                   the upper triangle of m are being stored)
c              3  reorder symmetrically stored matrix as specified by
c                   input permutation (used when an ordering has already
c                   been determined and only the nonzero entries in the
c                   upper triangle of m are being stored)
c              4  same as 2 but put diagonal entries at start of each row
c              5  same as 3 but put diagonal entries at start of each row
c
c    flag - integer error flag.  values and their meanings are -
c                 0    no errors detected
c               9n+k  insufficient storage in md
c              10n+1  insufficient storage in odrv
c              11n+1  illegal path specification
c
c
c  conversion from real to double precision
c
c    change the real declarations in odrv and sro to double precision
c    declarations.
c
c-----------------------------------------------------------------------
c
      integer  ia(1), ja(1),  p(1), ip(1),  isp(1),  path,  flag,
     *   v, l, head,  tmp, q
```

```
      double precision  a(1)
      logical  dflag
c
c----initialize error flag and validate path specification
      flag = 0
      if (path.lt.1 .or. 5.lt.path)  go to 111
c
c----allocate storage and find minimum degree ordering
      if ((path-1) * (path-2) * (path-4) .ne. 0)  go to 1
        max = (nsp-n)/2
        v    = 1
        l    = v     +  max
        head = l     +  max
        next = head  +  n
        if (max.lt.n)  go to 110
c
        call  md
     *     (n, ia,ja, max,isp(v),isp(l), isp(head),p,ip, isp(v), flag)
        if (flag.ne.0)  go to 100
c
c----allocate storage and symmetrically reorder matrix
   1  if ((path-2) * (path-3) * (path-4) * (path-5) .ne. 0)  go to 2
        tmp = (nsp+1) -      n
        q   = tmp     - (ia(n+1)-1)
        if (q.lt.1)  go to 110
c
        dflag = path.eq.4 .or. path.eq.5
        call sro
     *     (n,  ip,  ia, ja, a,  isp(tmp),  isp(q),  dflag)
c
   2  return
c
c ** error -- error detected in md
 100  return
c ** error -- insufficient storage
 110  flag = 10*n + 1
      return
c ** error -- illegal path specified
 111  flag = 11*n + 1
      return
      end
      subroutine md
     *     (n, ia,ja, max, v,l, head,last,next, mark, flag)
cl11. optimize
c***********************************************************************
c  md -- minimum degree algorithm (based on element model)
c***********************************************************************
c
c  description
c
c    md finds a minimum degree ordering of the rows and columns of a
c    general sparse matrix m stored in (ia,ja,a) format.
c    when the structure of m is nonsymmetric, the ordering is that
c    obtained for the symmetric matrix  m + m-transpose.
c
c
c  additional parameters
c
c    max  - declared dimension of the one-dimensional arrays v and l.
```

```
c           max must be at least  n+2k,  where k is the number of
c           nonzeroes in the strict upper triangle of m + m-transpose
c
c    v     - integer one-dimensional work array.  dimension = max
c
c    l     - integer one-dimensional work array.  dimension = max
c
c    head - integer one-dimensional work array.  dimension = n
c
c    last - integer one-dimensional array used to return the permutation
c           of the rows and columns of m corresponding to the minimum
c           degree ordering.  dimension = n
c
c    next - integer one-dimensional array used to return the inverse of
c           the permutation returned in last.  dimension = n
c
c    mark - integer one-dimensional work array (may be the same as v).
c           dimension = n
c
c    flag - integer error flag.  values and their meanings are -
c             0     no errors detected
c             9n+k  insufficient storage in md
c
c
c  definitions of internal parameters
c
c    ---------+-------------------------------------------------------
c    v(s)     - value field of list entry
c    ---------+-------------------------------------------------------
c    l(s)     - link field of list entry  (0 =) end of list)
c    ---------+-------------------------------------------------------
c    l(vi)    - pointer to element list of uneliminated vertex vi
c    ---------+-------------------------------------------------------
c    l(ej)    - pointer to boundary list of active element ej
c    ---------+-------------------------------------------------------
c    head(d)  - vj =) vj head of d-list d
c             -  0 =) no vertex in d-list d
c
c
c             -                  vi uneliminated vertex
c             -          vi in ek            -       vi not in ek
c    ---------+----------------------------+-------------------------
c    next(vi) - undefined but nonnegative   - vj =) vj next in d-list
c             -                             -  0 =) vi tail of d-list
c    ---------+----------------------------+-------------------------
c    last(vi) - (not set until mdp)         - -d =) vi head of d-list d
c             --vk =) compute degree        - vj =) vj last in d-list
c             - ej =) vi prototype of ej    -  0 =) vi not in any d-list
c             -  0 =) do not compute degree -
c    ---------+----------------------------+-------------------------
c    mark(vi) - mark(vk)                    - nonneg. tag .lt. mark(vk)
c
c
c             -                  vi eliminated vertex
c             -      ei active element      -           otherwise
c    ---------+----------------------------+-------------------------
c    next(vi) - -j =) vi was j-th vertex    - -j =) vi was j-th vertex
c             -       to be eliminated      -       to be eliminated
c    ---------+----------------------------+-------------------------
```

```
c    last(vi) -  m =) size of ei = m        - undefined
c    ---------+----------------------------+-------------------------
c    mark(vi) - -m =) overlap count of ei   - undefined
c             -       with ek = m           -
c             - otherwise nonnegative tag   -
c             -       .lt. mark(vk)         -
c
c-----------------------------------------------------------------------
c
      integer  ia(1), ja(1),  v(1), l(1),  head(1), last(1), next(1),
     *   mark(1),  flag,  tag, dmin, vk,ek, tail
      equivalence  (vk,ek)
c
c----initialization
      tag = 0
      call  mdi
     *   (n, ia,ja, max,v,l, head,last,next, mark,tag, flag)
      if (flag.ne.0)  return
c
      k = 0
      dmin = 1
c
c----while  k .lt. n  do
   1  if (k.ge.n)  go to 4
c
c------search for vertex of minimum degree
   2    if (head(dmin).gt.0)  go to 3
          dmin = dmin + 1
          go to 2
c
c------remove vertex vk of minimum degree from degree list
   3    vk = head(dmin)
        head(dmin) = next(vk)
        if (head(dmin).gt.0)  last(head(dmin)) = -dmin
c
c------number vertex vk, adjust tag, and tag vk
        k = k+1
        next(vk) = -k
        last(ek) = dmin - 1
        tag = tag + last(ek)
        mark(vk) = tag
c
c------form element ek from uneliminated neighbors of vk
        call  mdm
     *     (vk,tail, v,l, last,next, mark)
c
c------purge inactive elements and do mass elimination
        call  mdp
     *     (k,ek,tail, v,l, head,last,next, mark)
c
c------update degrees of uneliminated vertices in ek
        call  mdu
     *     (ek,dmin, v,l, head,last,next, mark)
c
        go to 1
c
c----generate inverse permutation from permutation
   4  do 5 k=1,n
        next(k) = -next(k)
```

```
   5    last(next(k)) = k
c
      return
      end
      subroutine mdi
     *     (n, ia,ja, max,v,l, head,last,next, mark,tag, flag)
clll. optimize
c***********************************************************************
c  mdi -- initialization
c***********************************************************************
      integer  ia(1), ja(1),  v(1), l(1),  head(1), last(1), next(1),
     *   mark(1), tag,  flag,  sfs, vi,dvi, vj
c
c----initialize degrees, element lists, and degree lists
      do 1 vi=1,n
        mark(vi) = 1
        l(vi) = 0
   1    head(vi) = 0
      sfs = n+1
c
c----create nonzero structure
c----for each nonzero entry a(vi,vj)
      do 6 vi=1,n
        jmin = ia(vi)
        jmax = ia(vi+1) - 1
        if (jmin.gt.jmax)  go to 6
        do 5 j=jmin,jmax
          vj = ja(j)
          if (vj-vi) 2, 5, 4
c
c------if a(vi,vj) is in strict lower triangle
c------check for previous occurrence of a(vj,vi)
   2      lvk = vi
          kmax = mark(vi) - 1
          if (kmax .eq. 0) go to 4
          do 3 k=1,kmax
            lvk = l(lvk)
            if (v(lvk).eq.vj) go to 5
   3        continue
c----for unentered entries a(vi,vj)
   4        if (sfs.ge.max)  go to 101
c
c------enter vj in element list for vi
            mark(vi) = mark(vi) + 1
            v(sfs) = vj
            l(sfs) = l(vi)
            l(vi) = sfs
            sfs = sfs+1
c
c------enter vi in element list for vj
            mark(vj) = mark(vj) + 1
            v(sfs) = vi
            l(sfs) = l(vj)
            l(vj) = sfs
            sfs = sfs+1
   5      continue
   6    continue
c
c----create degree lists and initialize mark vector
```

```
      do 7 vi=1,n
        dvi = mark(vi)
        next(vi) = head(dvi)
        head(dvi) = vi
        last(vi) = -dvi
        nextvi = next(vi)
        if (nextvi.gt.0)  last(nextvi) = vi
   7    mark(vi) = tag
c
      return
c
c ** error-  insufficient storage
 101  flag = 9*n + vi
      return
      end
      subroutine mdm
     *     (vk,tail, v,l, last,next, mark)
clll. optimize
c*********************************************************************
c  mdm -- form element from uneliminated neighbors of vk
c*********************************************************************
      integer  vk, tail,  v(1), l(1),   last(1), next(1),   mark(1),
     *   tag, s,ls,vs,es, b,lb,vb, blp,blpmax
      equivalence  (vs, es)
c
c----initialize tag and list of uneliminated neighbors
      tag = mark(vk)
      tail = vk
c
c----for each vertex/element vs/es in element list of vk
      ls = l(vk)
   1  s = ls
      if (s.eq.0)  go to 5
        ls = l(s)
        vs = v(s)
        if (next(vs).lt.0)  go to 2
c
c------if vs is uneliminated vertex, then tag and append to list of
c------uneliminated neighbors
          mark(vs) = tag
          l(tail) = s
          tail = s
          go to 4
c
c------if es is active element, then ...
c--------for each vertex vb in boundary list of element es
   2      lb = l(es)
          blpmax = last(es)
          do 3 blp=1,blpmax
            b = lb
            lb = l(b)
            vb = v(b)
c
c----------if vb is untagged vertex, then tag and append to list of
c----------uneliminated neighbors
            if (mark(vb).ge.tag)  go to 3
              mark(vb) = tag
              l(tail) = b
              tail = b
```

```
   3        continue
c
c--------mark es inactive
          mark(es) = tag
c
   4    go to 1
c
c----terminate list of uneliminated neighbors
   5  l(tail) = 0
c
      return
      end
      subroutine mdp
     *     (k,ek,tail, v,l, head,last,next, mark)
clll. optimize
c***********************************************************************
c  mdp -- purge inactive elements and do mass elimination
c***********************************************************************
      integer  ek, tail,  v(1), l(1),  head(1), last(1), next(1),
     *   mark(1),  tag, free, li,vi,lvi,evi, s,ls,es, ilp,ilpmax
c
c----initialize tag
      tag = mark(ek)
c
c----for each vertex vi in ek
      li = ek
      ilpmax = last(ek)
      if (ilpmax.le.0)  go to 12
      do 11 ilp=1,ilpmax
        i = li
        li = l(i)
        vi = v(li)
c
c------remove vi from degree list
        if (last(vi).eq.0)  go to 3
          if (last(vi).gt.0)  go to 1
            head(-last(vi)) = next(vi)
            go to 2
   1        next(last(vi)) = next(vi)
   2      if (next(vi).gt.0)  last(next(vi)) = last(vi)
c
c------remove inactive items from element list of vi
   3    ls = vi
   4    s = ls
        ls = l(s)
        if (ls.eq.0)  go to 6
          es = v(ls)
          if (mark(es).lt.tag)  go to 5
            free = ls
            l(s) = l(ls)
            ls = s
   5      go to 4
c
c------if vi is interior vertex, then remove from list and eliminate
   6    lvi = l(vi)
        if (lvi.ne.0)  go to 7
          l(i) = l(li)
          li = i
c
```

```
          k = k+1
          next(vi) = -k
          last(ek) = last(ek) - 1
          go to 11
c
c------else ...
c--------classify vertex vi
   7      if (l(lvi).ne.0)  go to 9
            evi = v(lvi)
            if (next(evi).ge.0)  go to 9
              if (mark(evi).lt.0)  go to 8
c
c----------if vi is prototype vertex, then mark as such, initialize
c----------overlap count for corresponding element, and move vi to end
c----------of boundary list
                last(vi) = evi
                mark(evi) = -1
                l(tail) = li
                tail = li
                l(i) = l(li)
                li = i
                go to 10
c
c----------else if vi is duplicate vertex, then mark as such and adjust
c----------overlap count for corresponding element
   8            last(vi) = 0
                mark(evi) = mark(evi) - 1
                go to 10
c
c----------else mark vi to compute degree
   9            last(vi) = -ek
c
c--------insert ek in element list of vi
  10       v(free) = ek
           l(free) = l(vi)
           l(vi) = free
  11     continue
c
c----terminate boundary list
  12  l(tail) = 0
c
      return
      end
      subroutine mdu
     *     (ek,dmin, v,l, head,last,next, mark)
cl11. optimize
c***********************************************************************
c  mdu -- update degrees of uneliminated vertices in ek
c***********************************************************************
      integer  ek, dmin,  v(1), l(1),  head(1), last(1), next(1),
     *   mark(1),  tag, vi,evi,dvi, s,vs,es, b,vb, ilp,ilpmax,
     *   blp,blpmax
      equivalence  (vs, es)
c
c----initialize tag
      tag = mark(ek) - last(ek)
c
c----for each vertex vi in ek
      i = ek
```

```
      ilpmax = last(ek)
      if (ilpmax.le.0)  go to 11
      do 10 ilp=1,ilpmax
        i = l(i)
        vi = v(i)
        if (last(vi))  1, 10, 8
c
c------if vi neither prototype nor duplicate vertex, then merge elements
c------to compute degree
   1      tag = tag + 1
          dvi = last(ek)
c
c--------for each vertex/element vs/es in element list of vi
          s = l(vi)
   2      s = l(s)
          if (s.eq.0)  go to 9
            vs = v(s)
            if (next(vs).lt.0)  go to 3
c
c----------if vs is uneliminated vertex, then tag and adjust degree
              mark(vs) = tag
              dvi = dvi + 1
              go to 5
c
c----------if es is active element, then expand
c------------check for outmatched vertex
   3          if (mark(es).lt.0)  go to 6
c
c------------for each vertex vb in es
              b = es
              blpmax = last(es)
              do 4 blp=1,blpmax
                b = l(b)
                vb = v(b)
c
c--------------if vb is untagged, then tag and adjust degree
                if (mark(vb).ge.tag)  go to 4
                  mark(vb) = tag
                  dvi = dvi + 1
   4            continue
c
   5        go to 2
c
c------else if vi is outmatched vertex, then adjust overlaps but do not
c------compute degree
   6      last(vi) = 0
          mark(es) = mark(es) - 1
   7      s = l(s)
          if (s.eq.0)  go to 10
            es = v(s)
            if (mark(es).lt.0)  mark(es) = mark(es) - 1
            go to 7
c
c------else if vi is prototype vertex, then calculate degree by
c------inclusion/exclusion and reset overlap count
   8      evi = last(vi)
          dvi = last(ek) + last(evi) + mark(evi)
          mark(evi) = 0
c
```

```
c------insert vi in appropriate degree list
   9     next(vi) = head(dvi)
         head(dvi) = vi
         last(vi) = -dvi
         if (next(vi).gt.0)  last(next(vi)) = vi
         if (dvi.lt.dmin)  dmin = dvi
c
  10     continue
c
  11  return
      end
      subroutine sro
     *     (n, ip, ia,ja,a, q, r, dflag)
clll. optimize
c*********************************************************************
c  sro -- symmetric reordering of sparse symmetric matrix
c*********************************************************************
c
c  description
c
c    the nonzero entries of the matrix m are assumed to be stored
c    symmetrically in (ia,ja,a) format (i.e., not both m(i,j) and m(j,i)
c    are stored if i ne j).
c
c    sro does not rearrange the order of the rows, but does move
c    nonzeroes from one row to another to ensure that if m(i,j) will be
c    in the upper triangle of m with respect to the new ordering, then
c    m(i,j) is stored in row i (and thus m(j,i) is not stored),  whereas
c    if m(i,j) will be in the strict lower triangle of m, then m(j,i) is
c    stored in row j (and thus m(i,j) is not stored).
c
c
c  additional parameters
c
c    q     - integer one-dimensional work array.  dimension = n
c
c    r     - integer one-dimensional work array.  dimension = number of
c            nonzero entries in the upper triangle of m
c
c    dflag - logical variable.  if dflag = .true., then store nonzero
c            diagonal elements at the beginning of the row
c
c---------------------------------------------------------------------
c
      integer  ip(1),  ia(1), ja(1),  q(1), r(1)
      double precision  a(1),  ak
      logical  dflag
c
c
c--phase 1 -- find row in which to store each nonzero
c----initialize count of nonzeroes to be stored in each row
      do 1 i=1,n
  1     q(i) = 0
c
c----for each nonzero element a(j)
      do 3 i=1,n
        jmin = ia(i)
        jmax = ia(i+1) - 1
        if (jmin.gt.jmax)  go to 3
```

```
      do 2 j=jmin,jmax
c
c--------find row (=r(j)) and column (=ja(j)) in which to store a(j) ...
         k = ja(j)
         if (ip(k).lt.ip(i))  ja(j) = i
         if (ip(k).ge.ip(i))  k = i
         r(j) = k
c
c--------... and increment count of nonzeroes (=q(r(j)) in that row
  2      q(k) = q(k) + 1
  3    continue
c
c
c--phase 2 -- find new ia and permutation to apply to (ja,a)
c----determine pointers to delimit rows in permuted (ja,a)
      do 4 i=1,n
        ia(i+1) = ia(i) + q(i)
  4     q(i) = ia(i+1)
c
c----determine where each (ja(j),a(j)) is stored in permuted (ja,a)
c----for each nonzero element (in reverse order)
      ilast = 0
      jmin = ia(1)
      jmax = ia(n+1) - 1
      j = jmax
      do 6 jdummy=jmin,jmax
        i = r(j)
        if (.not.dflag .or. ja(j).ne.i .or. i.eq.ilast)  go to 5
c
c------if dflag, then put diagonal nonzero at beginning of row
          r(j) = ia(i)
          ilast = i
          go to 6
c
c------put (off-diagonal) nonzero in last unused location in row
  5       q(i) = q(i) - 1
          r(j) = q(i)
c
  6     j = j-1
c
c
c--phase 3 -- permute (ja,a) to upper triangular form (wrt new ordering)
      do 8 j=jmin,jmax
  7     if (r(j).eq.j)  go to 8
          k = r(j)
          r(j) = r(k)
          r(k) = k
          jak = ja(k)
          ja(k) = ja(j)
          ja(j) = jak
          ak = a(k)
          a(k) = a(j)
          a(j) = ak
          go to 7
  8     continue
c
      return
      end
      subroutine cdrv
```

```
     *      (n, r,c,ic, ia,ja,a, b, z, nsp,isp,rsp,esp, path, flag)
clll. optimize
c*** subroutine cdrv
c*** driver for subroutines for solving sparse nonsymmetric systems of
c       linear equations (compressed pointer storage)
c
c
c    parameters
c    class abbreviations are--
c       n - integer variable
c       f - real variable
c       v - supplies a value to the driver
c       r - returns a result from the driver
c       i - used internally by the driver
c       a - array
c
c class - parameter
c ------+----------
c       -
c         the nonzero entries of the coefficient matrix m are stored
c    row-by-row in the array a.  to identify the individual nonzero
c    entries in each row, we need to know in which column each entry
c    lies.  the column indices which correspond to the nonzero entries
c    of m are stored in the array ja.  i.e., if  a(k) = m(i,j),  then
c    ja(k) = j.  in addition, we need to know where each row starts and
c    how long it is.  the index positions in ja and a where the rows of
c    m begin are stored in the array ia.  i.e., if m(i,j) is the first
c    nonzero entry (stored) in the i-th row and a(k) = m(i,j),  then
c    ia(i) = k.  moreover, the index in ja and a of the first location
c    following the last element in the last row is stored in ia(n+1).
c    thus, the number of entries in the i-th row is given by
c    ia(i+1) - ia(i),  the nonzero entries of the i-th row are stored
c    consecutively in
c            a(ia(i)),  a(ia(i)+1),  ..., a(ia(i+1)-1),
c    and the corresponding column indices are stored consecutively in
c            ja(ia(i)), ja(ia(i)+1), ..., ja(ia(i+1)-1).
c    for example, the 5 by 5 matrix
c                ( 1. 0. 2. 0. 0.)
c                ( 0. 3. 0. 0. 0.)
c            m = ( 0. 4. 5. 6. 0.)
c                ( 0. 0. 0. 7. 0.)
c                ( 0. 0. 0. 8. 9.)
c    would be stored as
c                - 1  2  3  4  5  6  7  8  9
c            ---+--------------------------
c            ia - 1  3  4  7  8 10
c            ja - 1  3  2  2  3  4  4  4  5
c             a - 1. 2. 3. 4. 5. 6. 7. 8. 9.
c
c nv    - n     - number of variables/equations.
c fva   - a     - nonzero entries of the coefficient matrix m, stored
c       -           by rows.
c       -           size = number of nonzero entries in m.
c nva   - ia    - pointers to delimit the rows in a.
c       -           size = n+1.
c nva   - ja    - column numbers corresponding to the elements of a.
c       -           size = size of a.
c fva   - b     - right-hand side b.  b and z can the same array.
c       -           size = n.
```

```
c fra   - z     - solution x.  b and z can be the same array.
c       -           size = n.
c
c         the rows and columns of the original matrix m can be
c    reordered (e.g., to reduce fillin or ensure numerical stability)
c    before calling the driver.  if no reordering is done, then set
c    r(i) = c(i) = ic(i) = i  for i=1,...,n.  the solution z is returned
c    in the original order.
c         if the columns have been reordered (i.e.,  c(i).ne.i  for some
c    i), then the driver will call a subroutine (nroc) which rearranges
c    each row of ja and a, leaving the rows in the original order, but
c    placing the elements of each row in increasing order with respect
c    to the new ordering.  if  path.ne.1,  then nroc is assumed to have
c    been called already.
c
c nva   - r     - ordering of the rows of m.
c       -           size = n.
c nva   - c     - ordering of the columns of m.
c       -           size = n.
c nva   - ic    - inverse of the ordering of the columns of m.  i.e.,
c       -           ic(c(i)) = i  for i=1,...,n.
c       -           size = n.
c
c         the solution of the system of linear equations is divided into
c    three stages --
c      nsfc -- the matrix m is processed symbolically to determine where
c               fillin will occur during the numeric factorization.
c      nnfc -- the matrix m is factored numerically into the product ldu
c               of a unit lower triangular matrix l, a diagonal matrix
c               d, and a unit upper triangular matrix u, and the system
c               mx = b  is solved.
c      nnsc -- the linear system  mx = b  is solved using the ldu
c  or           factorization from nnfc.
c      nntc -- the transposed linear system  mt x = b  is solved using
c               the ldu factorization from nnf.
c    for several systems whose coefficient matrices have the same
c    nonzero structure, nsfc need be done only once (for the first
c    system).  then nnfc is done once for each additional system.  for
c    several systems with the same coefficient matrix, nsfc and nnfc
c    need be done only once (for the first system).  then nnsc or nntc
c    is done once for each additional right-hand side.
c
c nv    - path  - path specification.  values and their meanings are --
c       -           1  perform nroc, nsfc, and nnfc.
c       -           2  perform nnfc only  (nsfc is assumed to have been
c       -               done in a manner compatible with the storage
c       -               allocation used in the driver).
c       -           3  perform nnsc only  (nsfc and nnfc are assumed to
c       -               have been done in a manner compatible with the
c       -               storage allocation used in the driver).
c       -           4  perform nntc only  (nsfc and nnfc are assumed to
c       -               have been done in a manner compatible with the
c       -               storage allocation used in the driver).
c       -           5  perform nroc and nsfc.
c
c         various errors are detected by the driver and the individual
c    subroutines.
c
c nr    - flag  - error flag.  values and their meanings are --
```

```
c       -            0      no errors detected
c       -            n+k    null row in a  --  row = k
c       -           2n+k    duplicate entry in a  --  row = k
c       -           3n+k    insufficient storage in nsfc  --  row = k
c       -           4n+1    insufficient storage in nnfc
c       -           5n+k    null pivot  --  row = k
c       -           6n+k    insufficient storage in nsfc  --  row = k
c       -           7n+1    insufficient storage in nnfc
c       -           8n+k    zero pivot  --  row = k
c       -          10n+1    insufficient storage in cdrv
c       -          11n+1    illegal path specification
c
c         working storage is needed for the factored form of the matrix
c    m plus various temporary vectors.  the arrays isp and rsp should be
c    equivalenced.  integer storage is allocated from the beginning of
c    isp and real storage from the end of rsp.
c
c nv    - nsp   - declared dimension of rsp.  nsp generally must
c       -           be larger than  8n+2 + 2k  (where  k = (number of
c       -           nonzero entries in m)).
c nvira - isp   - integer working storage divided up into various arrays
c       -           needed by the subroutines.  isp and rsp should be
c       -           equivalenced.
c       -           size = lratio*nsp.
c fvira - rsp   - real working storage divided up into various arrays
c       -           needed by the subroutines.  isp and rsp should be
c       -           equivalenced.
c       -           size = nsp.
c nr    - esp   - if sufficient storage was available to perform the
c       -           symbolic factorization (nsfc), then esp is set to
c       -           the amount of excess storage provided (negative if
c       -           insufficient storage was available to perform the
c       -           numeric factorization (nnfc)).
c
c
c  conversion to double precision
c
c    to convert these routines for double precision arrays..
c    (1) use the double precision declarations in place of the real
c    declarations in each subprogram, as given in comment cards.
c    (2) change the data-loaded value of the integer  lratio
c    in subroutine cdrv, as indicated below.
c    (3) change e0 to d0 in the constants in statement number 10
c    in subroutine nnfc and the line following that.
c
      integer  r(1), c(1), ic(1),  ia(1), ja(1),  isp(1), esp,  path,
     *   flag,  d, u, q, row, tmp, ar,  umax
      double precision  a(1), b(1), z(1), rsp(1)
c
c  set lratio equal to the ratio between the length of floating point
c  and integer array data.  e. g., lratio = 1 for (real, integer),
c  lratio = 2 for (double precision, integer)
c
      data lratio/2/
c
      if (path.lt.1 .or. 5.lt.path)  go to 111
c******initialize and divide up temporary storage  ********************
      il   = 1
      ijl  = il  + (n+1)
```

```
      iu   = ijl +   n
      iju  = iu  + (n+1)
      irl  = iju +   n
      jrl  = irl +   n
      jl   = jrl +   n
c
c  ******  reorder a if necessary, call nsfc if flag is set  ************
      if ((path-1) * (path-5) .ne. 0)  go to 5
        max = (lratio*nsp + 1 - jl) - (n+1) - 5*n
        jlmax = max/2
        q     = jl   + jlmax
        ira   = q    + (n+1)
        jra   = ira  +   n
        irac  = jra  +   n
        iru   = irac +   n
        jru   = iru  +   n
        jutmp = jru  +   n
        jumax = lratio*nsp  + 1 - jutmp
        esp = max/lratio
        if (jlmax.le.0 .or. jumax.le.0)  go to 110
c
        do 1 i=1,n
          if (c(i).ne.i)  go to 2
   1      continue
        go to 3
   2    ar = nsp + 1 - n
        call  nroc
     *     (n, ic, ia,ja,a, isp(il), rsp(ar), isp(iu), flag)
        if (flag.ne.0)  go to 100
c
   3    call  nsfc
     *     (n, r, ic, ia,ja,
     *      jlmax, isp(il), isp(jl), isp(ijl),
     *      jumax, isp(iu), isp(jutmp), isp(iju),
     *      isp(q), isp(ira), isp(jra), isp(irac),
     *      isp(irl), isp(jrl), isp(iru), isp(jru),  flag)
        if(flag .ne. 0)  go to 100
c  ******  move ju next to jl  ****************************************
        jlmax = isp(ijl+n-1)
        ju    = jl + jlmax
        jumax = isp(iju+n-1)
        if (jumax.le.0)  go to 5
        do 4 j=1,jumax
   4      isp(ju+j-1) = isp(jutmp+j-1)
c
c  ******  call remaining subroutines  **********************************
   5  jlmax = isp(ijl+n-1)
      ju    = jl  + jlmax
      jumax = isp(iju+n-1)
      l     = (ju + jumax - 2 + lratio)  /  lratio    +    1
      lmax  = isp(il+n) - 1
      d     = l   + lmax
      u     = d   + n
      row   = nsp + 1 - n
      tmp   = row - n
      umax  = tmp - u
      esp   = umax - (isp(iu+n) - 1)
c
      if ((path-1) * (path-2) .ne. 0)  go to 6
```

```
      if (umax.lt.0)  go to 110
      call nnfc
     *     (n,  r, c, ic,  ia, ja, a, z, b,
     *      lmax, isp(il), isp(jl), isp(ijl), rsp(l),  rsp(d),
     *      umax, isp(iu), isp(ju), isp(iju), rsp(u),
     *      rsp(row), rsp(tmp),  isp(irl), isp(jrl),  flag)
      if(flag .ne. 0)  go to 100
c
   6  if ((path-3) .ne. 0)  go to 7
      call nnsc
     *     (n,  r, c,  isp(il), isp(jl), isp(ijl), rsp(l),
     *      rsp(d),    isp(iu), isp(ju), isp(iju), rsp(u),
     *      z, b,  rsp(tmp))
c
   7  if ((path-4) .ne. 0)  go to 8
      call nntc
     *     (n,  r, c,  isp(il), isp(jl), isp(ijl), rsp(l),
     *      rsp(d),    isp(iu), isp(ju), isp(iju), rsp(u),
     *      z, b,  rsp(tmp))
   8  return
c
c ** error.. error detected in nroc, nsfc, nnfc, or nnsc
 100  return
c ** error.. insufficient storage
 110  flag = 10*n + 1
      return
c ** error.. illegal path specification
 111  flag = 11*n + 1
      return
      end
      subroutine nroc (n, ic, ia, ja, a, jar, ar, p, flag)
c111. optimize
c
c       ----------------------------------------------------------------
c
c                yale sparse matrix package - nonsymmetric codes
c                     solving the system of equations mx = b
c
c    i.   calling sequences
c         the coefficient matrix can be processed by an ordering routine
c    (e.g., to reduce fillin or ensure numerical stability) before using
c    the remaining subroutines.  if no reordering is done, then set
c    r(i) = c(i) = ic(i) = i  for i=1,...,n.  if an ordering subroutine
c    is used, then nroc should be used to reorder the coefficient matrix
c    the calling sequence is --
c        (       (matrix ordering))
c        (nroc   (matrix reordering))
c         nsfc   (symbolic factorization to determine where fillin will
c                  occur during numeric factorization)
c         nnfc   (numeric factorization into product ldu of unit lower
c                  triangular matrix l, diagonal matrix d, and unit
c                  upper triangular matrix u, and solution of linear
c                  system)
c         nnsc   (solution of linear system for additional right-hand
c                  side using ldu factorization from nnfc)
c    (if only one system of equations is to be solved, then the
c    subroutine trk should be used.)
c
c    ii.  storage of sparse matrices
```

```
c          the nonzero entries of the coefficient matrix m are stored
c     row-by-row in the array a.  to identify the individual nonzero
c     entries in each row, we need to know in which column each entry
c     lies.  the column indices which correspond to the nonzero entries
c     of m are stored in the array ja.  i.e., if  a(k) = m(i,j),  then
c     ja(k) = j.  in addition, we need to know where each row starts and
c     how long it is.  the index positions in ja and a where the rows of
c     m begin are stored in the array ia.  i.e., if m(i,j) is the first
c     (leftmost) entry in the i-th row and  a(k) = m(i,j),  then
c     ia(i) = k.  moreover, the index in ja and a of the first location
c     following the last element in the last row is stored in ia(n+1).
c     thus, the number of entries in the i-th row is given by
c     ia(i+1) - ia(i),  the nonzero entries of the i-th row are stored
c     consecutively in
c             a(ia(i)),  a(ia(i)+1),  ..., a(ia(i+1)-1),
c     and the corresponding column indices are stored consecutively in
c             ja(ia(i)), ja(ia(i)+1), ..., ja(ia(i+1)-1).
c     for example, the 5 by 5 matrix
c                 ( 1. 0. 2. 0. 0.)
c                 ( 0. 3. 0. 0. 0.)
c             m = ( 0. 4. 5. 6. 0.)
c                 ( 0. 0. 0. 7. 0.)
c                 ( 0. 0. 0. 8. 9.)
c     would be stored as
c                - 1  2  3  4  5  6  7  8  9
c             ---+---------------------------
c             ia - 1  3  4  7  8 10
c             ja - 1  3  2  2  3  4  4  4  5
c              a - 1. 2. 3. 4. 5. 6. 7. 8. 9.
c
c          the strict upper (lower) triangular portion of the matrix
c     u (l) is stored in a similar fashion using the arrays  iu, ju, u
c     (il, jl, l)  except that an additional array iju (ijl) is used to
c     compress storage of ju (jl) by allowing some sequences of column
c     (row) indices to used for more than one row (column)  (n.b., l is
c     stored by columns).  iju(k) (ijl(k)) points to the starting
c     location in ju (jl) of entries for the kth row (column).
c     compression in ju (jl) occurs in two ways.  first, if a row
c     (column) i was merged into the current row (column) k, and the
c     number of elements merged in from (the tail portion of) row
c     (column) i is the same as the final length of row (column) k, then
c     the kth row (column) and the tail of row (column) i are identical
c     and iju(k) (ijl(k)) points to the start of the tail.  second, if
c     some tail portion of the (k-1)st row (column) is identical to the
c     head of the kth row (column), then iju(k) (ijl(k)) points to the
c     start of that tail portion.  for example, the nonzero structure of
c     the strict upper triangular part of the matrix
c             d 0 x x x
c             0 d 0 x x
c             0 0 d x 0
c             0 0 0 d x
c             0 0 0 0 d
c     would be represented as
c                - 1 2 3 4 5 6
c             ----+------------
c              iu - 1 4 6 7 8 8
c              ju - 3 4 5 4
c             iju - 1 2 4 3
c     the diagonal entries of l and u are assumed to be equal to one and
```

```
c     are not stored.  the array d contains the reciprocals of the
c     diagonal entries of the matrix d.
c
c     iii. additional storage savings
c          in nsfc, r and ic can be the same array in the calling
c     sequence if no reordering of the coefficient matrix has been done.
c          in nnfc, r, c, and ic can all be the same array if no
c     reordering has been done.  if only the rows have been reordered,
c     then c and ic can be the same array.  if the row and column
c     orderings are the same, then r and c can be the same array.  z and
c     row can be the same array.
c          in nnsc or nntc, r and c can be the same array if no
c     reordering has been done or if the row and column orderings are the
c     same.  z and b can be the same array.  however, then b will be
c     destroyed.
c
c     iv.  parameters
c          following is a list of parameters to the programs.  names are
c     uniform among the various subroutines.  class abbreviations are --
c        n - integer variable
c        f - real variable
c        v - supplies a value to a subroutine
c        r - returns a result from a subroutine
c        i - used internally by a subroutine
c        a - array
c
c class - parameter
c ------+----------
c fva   - a     - nonzero entries of the coefficient matrix m, stored
c       -           by rows.
c       -           size = number of nonzero entries in m.
c fva   - b     - right-hand side b.
c       -           size = n.
c nva   - c     - ordering of the columns of m.
c       -           size = n.
c fvra  - d     - reciprocals of the diagonal entries of the matrix d.
c       -           size = n.
c nr    - flag  - error flag.  values and their meanings are --
c       -            0      no errors detected
c       -            n+k    null row in a  --  row = k
c       -           2n+k    duplicate entry in a  --  row = k
c       -           3n+k    insufficient storage for jl  --  row = k
c       -           4n+1    insufficient storage for l
c       -           5n+k    null pivot  --  row = k
c       -           6n+k    insufficient storage for ju  --  row = k
c       -           7n+1    insufficient storage for u
c       -           8n+k    zero pivot  --  row = k
c nva   - ia    - pointers to delimit the rows of a.
c       -           size = n+1.
c nvra  - ijl   - pointers to the first element in each column in jl,
c       -           used to compress storage in jl.
c       -           size = n.
c nvra  - iju   - pointers to the first element in each row in ju, used
c       -           to compress storage in ju.
c       -           size = n.
c nvra  - il    - pointers to delimit the columns of l.
c       -           size = n+1.
c nvra  - iu    - pointers to delimit the rows of u.
c       -           size = n+1.
```

```
c nva    - ja     - column numbers corresponding to the elements of a.
c        -            size = size of a.
c nvra   - jl     - row numbers corresponding to the elements of l.
c        -            size = jlmax.
c nv     - jlmax - declared dimension of jl.  jlmax must be larger than
c        -            the number of nonzeros in the strict lower triangle
c        -            of m plus fillin minus compression.
c nvra   - ju     - column numbers corresponding to the elements of u.
c        -            size = jumax.
c nv     - jumax - declared dimension of ju.  jumax must be larger than
c        -            the number of nonzeros in the strict upper triangle
c        -            of m plus fillin minus compression.
c fvra   - l      - nonzero entries in the strict lower triangular portion
c        -            of the matrix l, stored by columns.
c        -            size = lmax.
c nv     - lmax   - declared dimension of l.  lmax must be larger than
c        -            the number of nonzeros in the strict lower triangle
c        -            of m plus fillin  (il(n+1)-1 after nsfc).
c nv     - n      - number of variables/equations.
c nva    - r      - ordering of the rows of m.
c        -            size = n.
c fvra   - u      - nonzero entries in the strict upper triangular portion
c        -            of the matrix u, stored by rows.
c        -            size = umax.
c nv     - umax   - declared dimension of u.  umax must be larger than
c        -            the number of nonzeros in the strict upper triangle
c        -            of m plus fillin  (iu(n+1)-1 after nsfc).
c fra    - z      - solution x.
c        -            size = n.
c
c       ------------------------------------------------------------
c
c*** subroutine nroc
c*** reorders rows of a, leaving row order unchanged
c
c
c       input parameters.. n, ic, ia, ja, a
c       output parameters.. ja, a, flag
c
c       parameters used internally..
c nia   - p      - at the kth step, p is a linked list of the reordered
c        -            column indices of the kth row of a.  p(n+1) points
c        -            to the first entry in the list.
c        -            size = n+1.
c nia   - jar    - at the kth step,jar contains the elements of the
c        -            reordered column indices of a.
c        -            size = n.
c fia   - ar     - at the kth step, ar contains the elements of the
c        -            reordered row of a.
c        -            size = n.
c
      integer  ic(1), ia(1), ja(1), jar(1), p(1), flag
      double precision  a(1), ar(1)
c
c  ******  for each nonempty row  ********************************
      do 5 k=1,n
        jmin = ia(k)
        jmax = ia(k+1) - 1
        if(jmin .gt. jmax) go to 5
```

```
      p(n+1) = n + 1
c  ******  insert each element in the list  *********************
      do 3 j=jmin,jmax
        newj = ic(ja(j))
        i = n + 1
  1     if(p(i) .ge. newj) go to 2
          i = p(i)
          go to 1
  2     if(p(i) .eq. newj) go to 102
        p(newj) = p(i)
        p(i) = newj
        jar(newj) = ja(j)
        ar(newj) = a(j)
  3     continue
c  ******  replace old row in ja and a  **************************
      i = n + 1
      do 4 j=jmin,jmax
        i = p(i)
        ja(j) = jar(i)
  4     a(j) = ar(i)
  5   continue
     flag = 0
     return
c
c ** error.. duplicate entry in a
 102  flag = n + k
      return
      end
      subroutine nsfc
     *      (n, r, ic, ia,ja, jlmax,il,jl,ijl, jumax,iu,ju,iju,
     *       q, ira,jra, irac, irl,jrl, iru,jru, flag)
cl11. optimize
c*** subroutine nsfc
c*** symbolic ldu-factorization of nonsymmetric sparse matrix
c      (compressed pointer storage)
c
c
c       input variables.. n, r, ic, ia, ja, jlmax, jumax.
c       output variables.. il, jl, ijl, iu, ju, iju, flag.
c
c       parameters used internally..
c nia   - q     - suppose  m*  is the result of reordering  m.  if
c       -           processing of the ith row of  m*  (hence the ith
c       -           row of  u) is being done,  q(j)  is initially
c       -           nonzero if  m*(i,j) is nonzero (j.ge.i).  since
c       -           values need not be stored, each entry points to the
c       -           next nonzero and  q(n+1)  points to the first.  n+1
c       -           indicates the end of the list.  for example, if n=9
c       -           and the 5th row of  m*  is
c       -              0 x x 0 x 0 0 x 0
c       -           then  q  will initially be
c       -              a a a a 8 a a 10 5          (a - arbitrary).
c       -           as the algorithm proceeds, other elements of  q
c       -           are inserted in the list because of fillin.
c       -           q  is used in an analogous manner to compute the
c       -           ith column of  l.
c       -           size = n+1.
c nia   - ira,  - vectors used to find the columns of  m.  at the kth
c nia   - jra,      step of the factorization,  irac(k)  points to the
```

```
c nia    - irac       head of a linked list in  jra  of row indices i
c        -            such that i .ge. k and  m(i,k)  is nonzero.  zero
c        -            indicates the end of the list.  ira(i)  (i.ge.k)
c        -            points to the smallest j such that j .ge. k and
c        -            m(i,j)  is nonzero.
c        -            size of each = n.
c nia    - irl,   - vectors used to find the rows of  l.  at the kth step
c nia    - jrl        of the factorization,  jrl(k)  points to the head
c        -            of a linked list in  jrl  of column indices j
c        -            such j .lt. k and  l(k,j)  is nonzero.  zero
c        -            indicates the end of the list.  irl(j)  (j.lt.k)
c        -            points to the smallest i such that i .ge. k and
c        -            l(i,j)  is nonzero.
c        -            size of each = n.
c nia    - iru,   - vectors used in a manner analogous to  irl and jrl
c nia    - jru        to find the columns of  u.
c        -            size of each = n.
c
c  internal variables..
c    jlptr - points to the last position used in  jl.
c    juptr - points to the last position used in  ju.
c    jmin,jmax - are the indices in  a or u  of the first and last
c                elements to be examined in a given row.
c                for example,  jmin=ia(k), jmax=ia(k+1)-1.
c
      integer cend, qm, rend, rk, vj
      integer ia(1), ja(1), ira(1), jra(1), il(1), jl(1), ijl(1)
      integer iu(1), ju(1), iju(1), irl(1), jrl(1), iru(1), jru(1)
      integer r(1), ic(1), q(1), irac(1), flag
c
c  ******  initialize pointers  ****************************************
      npl = n + 1
      jlmin = 1
      jlptr = 0
      il(1) = 1
      jumin = 1
      juptr = 0
      iu(1) = 1
      do 1 k=1,n
        irac(k) = 0
        jra(k) = 0
        jrl(k) = 0
   1    jru(k) = 0
c  ******  initialize column pointers for a  ***************************
      do 2 k=1,n
        rk = r(k)
        iak = ia(rk)
        if (iak .ge. ia(rk+1))  go to 101
        jaiak = ic(ja(iak))
        if (jaiak .gt. k)  go to 105
        jra(k) = irac(jaiak)
        irac(jaiak) = k
   2    ira(k) = iak
c
c  ******  for each column of l and row of u  ***************************
      do 41 k=1,n
c
c  ******  initialize q for computing kth column of l  *****************
        q(npl) = npl
```

```
      luk = -1
c  ******  by filling in kth column of a  ******************************
      vj = irac(k)
      if (vj .eq. 0)  go to 5
  3     qm = np1
  4     m = qm
        qm =  q(m)
        if (qm .lt. vj)  go to 4
        if (qm .eq. vj)  go to 102
          luk = luk + 1
          q(m) = vj
          q(vj) = qm
          vj = jra(vj)
          if (vj .ne. 0)  go to 3
c  ******  link through jru  ********************************************
  5   lastid = 0
      lasti = 0
      ijl(k) = jlptr
      i = k
  6     i = jru(i)
        if (i .eq. 0)  go to 10
        qm = np1
        jmin = irl(i)
        jmax = ijl(i) + il(i+1) - il(i) - 1
        long = jmax - jmin
        if (long .lt. 0)  go to 6
        jtmp = jl(jmin)
        if (jtmp .ne. k)  long = long + 1
        if (jtmp .eq. k)  r(i) = -r(i)
        if (lastid .ge. long)  go to 7
          lasti = i
          lastid = long
c  ******  and merge the corresponding columns into the kth column  ****
  7     do 9 j=jmin,jmax
          vj = jl(j)
  8       m = qm
          qm = q(m)
          if (qm .lt. vj)  go to 8
          if (qm .eq. vj)  go to 9
            luk = luk + 1
            q(m) = vj
            q(vj) = qm
            qm = vj
  9       continue
          go to 6
c  ******  lasti is the longest column merged into the kth  ************
c  ******  see if it equals the entire kth column  *********************
 10   qm = q(np1)
      if (qm .ne. k)  go to 105
      if (luk .eq. 0)  go to 17
      if (lastid .ne. luk)  go to 11
c  ******  if so, jl can be compressed  *********************************
      irll = irl(lasti)
      ijl(k) = irll + 1
      if (jl(irll) .ne. k)  ijl(k) = ijl(k) - 1
      go to 17
c  ******  if not, see if kth column can overlap the previous one  *****
 11   if (jlmin .gt. jlptr)  go to 15
      qm = q(qm)
```

```
      do 12 j=jlmin,jlptr
        if (jl(j) - qm)  12, 13, 15
 12     continue
      go to 15
 13   ijl(k) = j
      do 14 i=j,jlptr
        if (jl(i) .ne. qm)  go to 15
        qm = q(qm)
        if (qm .gt. n)  go to 17
 14     continue
      jlptr = j - 1
c  ******  move column indices from q to jl, update vectors  ***********
 15   jlmin = jlptr + 1
      ijl(k) = jlmin
      if (luk .eq. 0)  go to 17
      jlptr = jlptr + luk
      if (jlptr .gt. jlmax)  go to 103
        qm = q(np1)
        do 16 j=jlmin,jlptr
          qm = q(qm)
 16       jl(j) = qm
 17   irl(k) = ijl(k)
      il(k+1) = il(k) + luk
c
c  ******  initialize q for computing kth row of u  ********************
      q(np1) = np1
      luk = -1
c  ******  by filling in kth row of reordered a  ************************
      rk = r(k)
      jmin = ira(k)
      jmax = ia(rk+1) - 1
      if (jmin .gt. jmax)  go to 20
      do 19 j=jmin,jmax
        vj = ic(ja(j))
        qm = np1
 18     m = qm
        qm = q(m)
        if (qm .lt. vj)  go to 18
        if (qm .eq. vj)  go to 102
          luk = luk + 1
          q(m) = vj
          q(vj) = qm
 19     continue
c  ******  link through jrl,  ******************************************
 20   lastid = 0
      lasti = 0
      iju(k) = juptr
      i = k
      i1 = jrl(k)
 21     i = i1
        if (i .eq. 0)  go to 26
        i1 = jrl(i)
        qm = np1
        jmin = iru(i)
        jmax = iju(i) + iu(i+1) - iu(i) - 1
        long = jmax - jmin
        if (long .lt. 0)  go to 21
        jtmp = ju(jmin)
        if (jtmp .eq. k)  go to 22
```

```
c  ******  update irl and jrl, ****************************************
            long = long + 1
            cend = ijl(i) + il(i+1) - il(i)
            irl(i) = irl(i) + 1
            if (irl(i) .ge. cend)  go to 22
              j = jl(irl(i))
              jrl(i) = jrl(j)
              jrl(j) = i
  22      if (lastid .ge. long)  go to 23
            lasti = i
            lastid = long
c  ******  and merge the corresponding rows into the kth row  **********
  23      do 25 j=jmin,jmax
            vj = ju(j)
  24        m = qm
            qm = q(m)
            if (qm .lt. vj)  go to 24
            if (qm .eq. vj)  go to 25
              luk = luk + 1
              q(m) = vj
              q(vj) = qm
              qm = vj
  25        continue
          go to 21
c  ******  update jrl(k) and irl(k)  ***********************************
  26    if (il(k+1) .le. il(k))  go to 27
          j = jl(irl(k))
          jrl(k) = jrl(j)
          jrl(j) = k
c  ******  lasti is the longest row merged into the kth  ***************
c  ******  see if it equals the entire kth row  ************************
  27    qm = q(npl)
        if (qm .ne. k)  go to 105
        if (luk .eq. 0)  go to 34
        if (lastid .ne. luk)  go to 28
c  ******  if so, ju can be compressed  *********************************
        irul = iru(lasti)
        iju(k) = irul + 1
        if (ju(irul) .ne. k)  iju(k) = iju(k) - 1
        go to 34
c  ******  if not, see if kth row can overlap the previous one  ********
  28    if (jumin .gt. juptr)  go to 32
        qm = q(qm)
        do 29 j=jumin,juptr
          if (ju(j) - qm)  29, 30, 32
  29      continue
        go to 32
  30    iju(k) = j
        do 31 i=j,juptr
          if (ju(i) .ne. qm)  go to 32
          qm = q(qm)
          if (qm .gt. n)  go to 34
  31      continue
        juptr = j - 1
c  ******  move row indices from q to ju, update vectors  **************
  32    jumin = juptr + 1
        iju(k) = jumin
        if (luk .eq. 0)  go to 34
        juptr = juptr + luk
```

```
      if (juptr .gt. jumax)  go to 106
        qm = q(np1)
        do 33 j=jumin,juptr
          qm = q(qm)
 33       ju(j) = qm
 34   iru(k) = iju(k)
      iu(k+1) = iu(k) + luk
c
c  ******  update iru, jru  ********************************************
      i = k
 35     i1 = jru(i)
        if (r(i) .lt. 0)  go to 36
        rend = iju(i) + iu(i+1) - iu(i)
        if (iru(i) .ge. rend)  go to 37
          j = ju(iru(i))
          jru(i) = jru(j)
          jru(j) = i
          go to 37
 36     r(i) = -r(i)
 37     i = i1
        if (i .eq. 0)  go to 38
        iru(i) = iru(i) + 1
        go to 35
c
c  ******  update ira, jra, irac  ************************************
 38   i = irac(k)
      if (i .eq. 0)  go to 41
 39     i1 = jra(i)
        ira(i) = ira(i) + 1
        if (ira(i) .ge. ia(r(i)+1))  go to 40
        irai = ira(i)
        jairai = ic(ja(irai))
        if (jairai .gt. i)  go to 40
        jra(i) = irac(jairai)
        irac(jairai) = i
 40     i = i1
        if (i .ne. 0)  go to 39
 41   continue
c
      ijl(n) = jlptr
      iju(n) = juptr
      flag = 0
      return
c
c ** error.. null row in a
 101  flag = n + rk
      return
c ** error.. duplicate entry in a
 102  flag = 2*n + rk
      return
c ** error.. insufficient storage for jl
 103  flag = 3*n + k
      return
c ** error.. null pivot
 105  flag = 5*n + k
      return
c ** error.. insufficient storage for ju
 106  flag = 6*n + k
      return
```

```
      end
      subroutine nnfc
     *     (n, r,c,ic, ia,ja,a, z, b,
     *      lmax,il,jl,ijl,l, d, umax,iu,ju,iju,u,
     *      row, tmp, irl,jrl, flag)
clll. optimize
c*** subroutine nnfc
c*** numerical ldu-factorization of sparse nonsymmetric matrix and
c      solution of system of linear equations (compressed pointer
c      storage)
c
c
c       input variables..  n, r, c, ic, ia, ja, a, b,
c                          il, jl, ijl, lmax, iu, ju, iju, umax
c       output variables.. z, l, d, u, flag
c
c       parameters used internally..
c nia   - irl,  - vectors used to find the rows of  l.  at the kth step
c nia   - jrl       of the factorization,  jrl(k)  points to the head
c       -           of a linked list in  jrl  of column indices j
c       -           such j .lt. k and  l(k,j)  is nonzero.  zero
c       -           indicates the end of the list.  irl(j)  (j.lt.k)
c       -           points to the smallest i such that i .ge. k and
c       -           l(i,j)  is nonzero.
c       -           size of each = n.
c fia   - row   - holds intermediate values in calculation of  u and l.
c       -           size = n.
c fia   - tmp   - holds new right-hand side  b*  for solution of the
c       -           equation ux = b*.
c       -           size = n.
c
c  internal variables..
c    jmin, jmax - indices of the first and last positions in a row to
c      be examined.
c    sum - used in calculating  tmp.
c
      integer rk,umax
      integer  r(1), c(1), ic(1), ia(1), ja(1), il(1), jl(1), ijl(1)
      integer  iu(1), ju(1), iju(1), irl(1), jrl(1), flag
      double precision  a(1), l(1), d(1), u(1), z(1), b(1), row(1)
      double precision  tmp(1), lki, sum, dk
c
c  ******  initialize pointers and test storage  ************************
      if(il(n+1)-1 .gt. lmax) go to 104
      if(iu(n+1)-1 .gt. umax) go to 107
      do 1 k=1,n
        irl(k) = il(k)
        jrl(k) = 0
   1    continue
c
c  ******  for each row  ***********************************************
      do 19 k=1,n
c  ******  reverse jrl and zero row where kth row of l will fill in  ***
        row(k) = 0
        i1 = 0
        if (jrl(k) .eq. 0) go to 3
        i = jrl(k)
   2    i2 = jrl(i)
        jrl(i) = i1
```

```
      i1 = i
      row(i) = 0
      i = i2
      if (i .ne. 0) go to 2
c ******  set row to zero where u will fill in  ************************
  3   jmin = iju(k)
      jmax = jmin + iu(k+1) - iu(k) - 1
      if (jmin .gt. jmax) go to 5
      do 4 j=jmin,jmax
  4     row(ju(j)) = 0
c ******  place kth row of a in row  **********************************
  5   rk = r(k)
      jmin = ia(rk)
      jmax = ia(rk+1) - 1
      do 6 j=jmin,jmax
        row(ic(ja(j))) = a(j)
  6     continue
c ****** initialize sum, and link through jrl  ************************
      sum = b(rk)
      i = i1
      if (i .eq. 0) go to 10
c ******  assign the kth row of l and adjust row, sum  ****************
  7     lki = -row(i)
c ******  if l is not required, then comment out the following line  **
        l(irl(i)) = -lki
        sum = sum + lki * tmp(i)
        jmin = iu(i)
        jmax = iu(i+1) - 1
        if (jmin .gt. jmax) go to 9
        mu = iju(i) - jmin
        do 8 j=jmin,jmax
  8       row(ju(mu+j)) = row(ju(mu+j)) + lki * u(j)
  9     i = jrl(i)
        if (i .ne. 0) go to 7
c
c ******  assign kth row of u and diagonal d, set tmp(k)  *************
 10   if (row(k) .eq. 0.0d0) go to 108
      dk = 1.0d0 / row(k)
      d(k) = dk
      tmp(k) = sum * dk
      if (k .eq. n) go to 19
      jmin = iu(k)
      jmax = iu(k+1) - 1
      if (jmin .gt. jmax)  go to 12
      mu = iju(k) - jmin
      do 11 j=jmin,jmax
 11     u(j) = row(ju(mu+j)) * dk
 12   continue
c
c ******  update irl and jrl, keeping jrl in decreasing order  ********
      i = i1
      if (i .eq. 0) go to 18
 14   irl(i) = irl(i) + 1
      i1 = jrl(i)
      if (irl(i) .ge. il(i+1)) go to 17
      ijlb = irl(i) - il(i) + ijl(i)
      j = jl(ijlb)
 15   if (i .gt. jrl(j)) go to 16
        j = jrl(j)
```

```
         go to 15
 16    jrl(i) = jrl(j)
       jrl(j) = i
 17    i = i1
       if (i .ne. 0) go to 14
 18    if (irl(k) .ge. il(k+1)) go to 19
       j = jl(ijl(k))
       jrl(k) = jrl(j)
       jrl(j) = k
 19    continue
c
c  ******  solve  ux = tmp  by back substitution  **********************
      k = n
      do 22 i=1,n
        sum =  tmp(k)
        jmin = iu(k)
        jmax = iu(k+1) - 1
        if (jmin .gt. jmax)  go to 21
        mu = iju(k) - jmin
        do 20 j=jmin,jmax
 20       sum = sum - u(j) * tmp(ju(mu+j))
 21     tmp(k) =  sum
        z(c(k)) =  sum
 22     k = k-1
      flag = 0
      return
c
c ** error.. insufficient storage for l
 104  flag = 4*n + 1
      return
c ** error.. insufficient storage for u
 107  flag = 7*n + 1
      return
c ** error.. zero pivot
 108  flag = 8*n + k
      return
      end
      subroutine nnsc
     *     (n, r, c, il, jl, ijl, l, d, iu, ju, iju, u, z, b, tmp)
cl11. optimize
c*** subroutine nnsc
c*** numerical solution of sparse nonsymmetric system of linear
c      equations given ldu-factorization (compressed pointer storage)
c
c
c       input variables..  n, r, c, il, jl, ijl, l, d, iu, ju, iju, u, b
c       output variables.. z
c
c       parameters used internally..
c fia   - tmp   - temporary vector which gets result of solving  ly = b.
c       -           size = n.
c
c  internal variables..
c    jmin, jmax - indices of the first and last positions in a row of
c      u or l  to be used.
c
      integer r(1), c(1), il(1), jl(1), ijl(1), iu(1), ju(1), iju(1)
      double precision  l(1), d(1), u(1), b(1), z(1), tmp(1), tmpk,sum
c
```

```
c  ******  set tmp to reordered b  ************************************
      do 1 k=1,n
   1    tmp(k) = b(r(k))
c  ******  solve  ly = b  by forward substitution  *********************
      do 3 k=1,n
        jmin = il(k)
        jmax = il(k+1) - 1
        tmpk = -d(k) * tmp(k)
        tmp(k) = -tmpk
        if (jmin .gt. jmax) go to 3
        ml = ijl(k) - jmin
        do 2 j=jmin,jmax
   2      tmp(jl(ml+j)) = tmp(jl(ml+j)) + tmpk * l(j)
   3    continue
c  ******  solve  ux = y  by back substitution  ************************
      k = n
      do 6 i=1,n
        sum = -tmp(k)
        jmin = iu(k)
        jmax = iu(k+1) - 1
        if (jmin .gt. jmax) go to 5
        mu = iju(k) - jmin
        do 4 j=jmin,jmax
   4      sum = sum + u(j) * tmp(ju(mu+j))
   5    tmp(k) = -sum
        z(c(k)) = -sum
        k = k - 1
   6    continue
      return
      end
      subroutine nntc
     *     (n, r, c, il, jl, ijl, l, d, iu, ju, iju, u, z, b, tmp)
cl11. optimize
c*** subroutine nntc
c*** numeric solution of the transpose of a sparse nonsymmetric system
c      of linear equations given lu-factorization (compressed pointer
c      storage)
c
c
c       input variables..  n, r, c, il, jl, ijl, l, d, iu, ju, iju, u, b
c       output variables.. z
c
c       parameters used internally..
c fia   - tmp   - temporary vector which gets result of solving ut y = b
c       -           size = n.
c
c  internal variables..
c    jmin, jmax - indices of the first and last positions in a row of
c      u or l  to be used.
c
      integer r(1), c(1), il(1), jl(1), ijl(1), iu(1), ju(1), iju(1)
      double precision l(1), d(1), u(1), b(1), z(1), tmp(1), tmpk,sum
c
c  ******  set tmp to reordered b  ************************************
      do 1 k=1,n
   1    tmp(k) = b(c(k))
c  ******  solve  ut y = b  by forward substitution  ******************
      do 3 k=1,n
        jmin = iu(k)
```

```
        jmax = iu(k+1) - 1
        tmpk = -tmp(k)
        if (jmin .gt. jmax) go to 3
        mu =  u(k) - jmin
        do 2   jmin,jmax
   2      tmp(ju(mu+j)) = tmp(ju(mu+j)) + tmpk * u(j)
   3    continue
c  ******  solve  lt x = y  by back substitution  **********************
      k = n
      do 6 i=1,n
        sum = -tmp(k)
        jmin = il(k)
        jmax = il(k+1) - 1
        if (jmin .gt. jmax) go to 5
        ml = ijl(k) - jmin
        do 4 j=jmin,jmax
   4      sum = sum + l(j) * tmp(jl(ml+j))
   5    tmp(k) = -sum * d(k)
        z(r(k)) = tmp(k)
        k = k - 1
   6    continue
      return
      end
      double precision function dlmach (idum)
      integer idum
c-----------------------------------------------------------------------
c this routine computes the unit roundoff of the machine in double
c precision.  this is defined as the smallest positive machine number
c u such that  1.0d0 + u .ne. 1.0d0 (in double precision).
c-----------------------------------------------------------------------
      double precision u, comp
      u = 1.0d0
 10   u = u*0.5d0
      comp = 1.0d0 + u
      if (comp .ne. 1.0d0) go to 10
      dlmach = u*2.0d0
      return
c----------------------- end of function dlmach -----------------------
      end


c      subroutine xerrwv (msg, nmes, nerr, level, ni, i1, i2, nr, r1, r2)
c     integer msg, nmes, nerr, level, ni, i1, i2, nr,
c      integer      nmes, nerr, level, ni, i1, i2, nr,
c     1   i, lun, lunit, mesflg, ncpw, nch, nwds
c      double precision r1, r2
c     dimension msg(nmes)
c      character*(*) msg
c-----------------------------------------------------------------------
c subroutines xerrwv, xsetf, and xsetun, as given here, constitute
c a simplified version of the slatec error handling package.
c written by a. c. hindmarsh at llnl.  version of march 30, 1987.
c this version is in double precision.
c
c all arguments are input arguments.
c
c msg    = the message (hollerith literal or integer array).
c nmes   = the length of msg (number of characters).
c nerr   = the error number (not used).
```

```
c level  = the error level..
c          0 or 1 means recoverable (control returns to caller).
c          2 means fatal (run is aborted--see note below).
c ni     = number of integers (0, 1, or 2) to be printed with message.
c i1,i2  = integers to be printed, depending on ni.
c nr     = number of reals (0, 1, or 2) to be printed with message.
c r1,r2  = reals to be printed, depending on nr.
c
c note..  this routine is machine-dependent and specialized for use
c in limited context, in the following ways..
c 1. the number of hollerith characters stored per word, denoted
c    by ncpw below, is a data-loaded constant.
c 2. the value of nmes is assumed to be at most 60.
c    (multi-line messages are generated by repeated calls.)
c 3. if level = 2, control passes to the statement   stop
c    to abort the run.  this statement may be machine-dependent.
c 4. r1 and r2 are assumed to be in double precision and are printed
c    in d21.13 format.
c 5. the common block /eh0001/ below is data-loaded (a machine-
c    dependent feature) with default values.
c    this block is needed for proper retention of parameters used by
c    this routine which the user can reset by calling xsetf or xsetun.
c    the variables in this block are as follows..
c       mesflg = print control flag..
c                1 means print all messages (the default).
c                0 means no printing.
c       lunit  = logical unit number for messages.
c                the default is 6 (machine-dependent).
c-----------------------------------------------------------------------
c the following are instructions for installing this routine
c in different machine environments.
c
c to change the default output unit, change the data statement
c in the block data subprogram below.
c
c for a different number of characters per word, change the
c data statement setting ncpw below, and format 10.  alternatives for
c various computers are shown in comment cards.
c
c for a different run-abort command, change the statement following
c statement 100 at the end.
c-----------------------------------------------------------------------
c      common /eh0001/ mesflg, lunit
c-----------------------------------------------------------------------
c the following data-loaded value of ncpw is valid for the cdc-6600
c and cdc-7600 computers.
c     data ncpw/10/
c the following is valid for the cray-1 computer.
c     data ncpw/8/
c the following is valid for the burroughs 6700 and 7800 computers.
c     data ncpw/6/
c the following is valid for the pdp-10 computer.
c     data ncpw/5/
c the following is valid for the vax computer with 4 bytes per integer,
c and for the ibm-360, ibm-370, ibm-303x, and ibm-43xx computers.
c     data ncpw/4/
c the following is valid for the pdp-11, or vax with 2-byte integers.
c     data ncpw/2/
c-----------------------------------------------------------------------
```

```
c       if (mesflg .eq. 0) go to 100
c get logical unit number. ----------------------------------------
c       lun = lunit
c get number of words in message. ----------------------------------
c      nch = min0(nmes,60)
c      nwds = nch/ncpw
c      if (nch .ne. nwds*ncpw) nwds = nwds + 1
c write the message. ------------------------------------------------
c      write (lun, 10) (msg(i),i=1,nwds)

CCC got rid of the messages to standard out - Anne 8/2/95
CCC       write (lun, 10) msg

c-------------------------------------------------------------------
c the following format statement is to have the form
c 10  format(1x,mmann)
c where nn = ncpw and mm is the smallest integer .ge. 60/ncpw.
c the following is valid for ncpw = 10.
c 10  format(1x,6a10)
c the following is valid for ncpw = 8.
c 10  format(1x,8a8)
c the following is valid for ncpw = 6.
c 10  format(1x,10a6)
c the following is valid for ncpw = 5.
c 10  format(1x,12a5)
c the following is valid for ncpw = 4.
c 10  format(1x,15a4)
c the following is valid for ncpw = 2.
c 10  format(1x,30a2)

CCC got rid of the messages to standard out - Anne 8/2/95
CCC  10  format(1x,a)
c-------------------------------------------------------------------
CCC       if (ni .eq. 1) write (lun, 20) i1
CCC 20    format(6x,23hin above message,  i1 =,i10)
CCC       if (ni .eq. 2) write (lun, 30) i1,i2
CCC 30    format(6x,23hin above message,  i1 =,i10,3x,4hi2 =,i10)
CCC       if (nr .eq. 1) write (lun, 40) r1
CCC 40    format(6x,23hin above message,  r1 =,d21.13)
CCC       if (nr .eq. 2) write (lun, 50) r1,r2
CCC 50    format(6x,15hin above,  r1 =,d21.13,3x,4hr2 =,d21.13)

c abort the run if level = 2. ---------------------------------------
c 100  if (level .ne. 2) return
c      stop
c---------------------- end of subroutine xerrwv ----------------------
c      end


      subroutine xsetf (mflag)
c
c this routine resets the print control flag mflag.
c
      integer mflag, mesflg, lunit
      common /eh0001/ mesflg, lunit
c
      if (mflag .eq. 0 .or. mflag .eq. 1) mesflg = mflag
      return
c---------------------- end of subroutine xsetf ----------------------
```

```
      end
      subroutine xsetun (lun)
c
c this routine resets the logical unit number for messages.
c
      integer lun, mesflg, lunit
      common /eh0001/ mesflg, lunit
c
      if (lun .gt. 0) lunit = lun
      return
c----------------------- end of subroutine xsetun ----------------------
      end
      block data
c-----------------------------------------------------------------------
c this data subprogram loads variables into the internal common
c blocks used by the odepack solvers.  the variables are
c defined as follows..
c   illin  = counter for the number of consecutive times the package
c            was called with illegal input.  the run is stopped when
c            illin reaches 5.
c   ntrep  = counter for the number of consecutive times the package
c            was called with istate = 1 and tout = t.  the run is
c            stopped when ntrep reaches 5.
c   mesflg = flag to control printing of error messages.  1 means print,
c            0 means no printing.
c   lunit  = default value of logical unit number for printing of error
c            messages.
c-----------------------------------------------------------------------
      integer illin, iduma, ntrep, idumb, iowns, icomm, mesflg, lunit
      double precision rowns, rcomm
      common /ls0001/ rowns(209), rcomm(9),
     1   illin, iduma(10), ntrep, idumb(2), iowns(6), icomm(19)
      common /eh0001/ mesflg, lunit
      data illin/0/, ntrep/0/
      data mesflg/1/, lunit/6/
c
c------------------------ end of block data ----------------------------
      end
```

F.27 DOSMAX.FOR

```
c       This subroutine checks for the maximum listed radionuclide
c       doses per pathway and saves them and their time of occurence

      subroutine dosmax(mxpath,mxlist,npath,nlist,tcalc,period,
     $    dosup,tdos,tpdos,dosmx)

      double precision dosmx,dosup
      double precision tcalc,tdos
      integer ilist,ipath
      integer nlist,npath
      integer mxlist,mxpath
      integer period, tpdos
      dimension dosmx(mxlist,mxpath),dosup(mxlist,mxpath)
      dimension tdos(mxlist,mxpath)
      dimension tpdos(mxlist,mxpath)

      do 20, ilist=1,nlist
         do 10, ipath=1,npath
            if (dosup(ilist,ipath) .gt. dosmx(ilist,ipath)) then
               dosmx(ilist,ipath)=dosup(ilist,ipath)
               tdos(ilist,ipath)=tcalc
               tpdos(ilist,ipath)=period
            endif
   10    continue
   20 continue
      end
```

F.28 DRIDEF.FOR

```
c
c...This subroutine reads the drinking water scenario default
c   values.
c
      subroutine dridef(in,ad,f1,f2,h1,h2,nlay,i,n1,n2,
     $   rho1,rho2,td,ttd,uw,vdd,tstart,tend,dt,skip,ioerr)

      double precision ad,dt
      double precision f1,f2
      double precision h1,h2
      double precision i
      double precision n1,n2
      double precision rho1,rho2
      double precision td,ttd,tstart,tend
      double precision uw
      double precision vdd,tskip,tnlay
      integer in,ioerr,nlay,skip

CCC...added a parameter for the number of layers

c...read constant parameter data
c     rho1 = surface soil density
c     rho2 = unsaturated zone soil density
c     ttd = exposure duration time
c     td = drinking water ingestion period
c     uw = drinking water ingestion rate
c     h1 = surface soil layer thickness
c     h2 = unsaturated zone soil thickness
c     nlay = number of layers in unsaturated zone
c     n1 = surface soil layer porosity
c     n2 = unsaturated zone porosity
c     f1 = surface soil layer saturation ratio
c     f2 = unsaturated zone saturation ratio
c     vdd = domestic water use ratio
c     i = infiltration rate
c     ad = contaminated site area

      read(in,*,iostat=ioerr) rho1,rho2,ttd,
     $ td,uw,h1,h2,tnlay,n1,n2,
     $ f1,f2,vdd,i,ad,
     $ tstart,tend,dt,tskip

          if (ioerr .ne. 0) then
              call ioreturn(ioerr)
              return
          endif

      skip = tskip
      nlay = tnlay

      end
```

F.29 DRIDOS.FOR

```
c
c...this subroutine calculates the doses for each radionuclide
c   in a chain for each pathway for the drinking water scenario
c

      subroutine dridos(out,rc,mxnr,mxnip,mxorg,mxvar,con,ci,
     $   concen,infil0,atol,ad1old,dd,ddd,dfg,dfo,ia,ja,lambdr,nbox,neq,
     $   nip,nlay,nnz,norg,nr,prfrac,rtol,t,td,ttd,uw,vtd,
     $   cwt,cswt,ad1new,dgd,hocde,ioerr)

      double precision ad1new,ad1old,atol
      double precision cc,cdel,ci,cint,cwt,cswt
      double precision dd,ddd,dfg,dfo,dgd
      double precision hocde
      double precision lambdr
      double precision prfrac
      double precision rsp,rtol
      double precision sd3
      double precision t,td,tnew,told,ttd
      double precision uw
      double precision vtd
      double precision zero
      integer c,con
      integer esp
      integer flag
      integer ia,ic,ieq,iip,ioerr,iorg,ir,isp,istate
      integer ja
      integer lratio
      integer maxbox,maxneq,maxnnz,maxnsp,maxnr,maxvar
      integer mxnip,mxnr,mxorg,mxvar
      integer nbox,neq,nip,nlay,nnz,norg,nr,nsp
      integer out
      integer r,rc
      logical concen
      logical infil0
      parameter (lratio=2)
      parameter (maxbox=12,maxnr=20,maxvar=3*maxnr)
      parameter (maxneq=maxbox*maxnr)
      parameter (maxnnz=(maxbox*maxnr*(maxnr+3))/2)
      parameter (maxnsp=8*maxneq+2*maxnnz+2)
      parameter (zero=0.0d0)
      dimension ad1new(maxneq),ad1old(maxneq)
      dimension c(maxneq)
      dimension cc(maxvar)
      dimension cdel(maxneq)
      dimension ci(mxvar)
      dimension cint(maxneq)
      dimension cswt(mxnr,0:mxnip),cwt(mxnr,0:mxnip)
      dimension dd(mxvar,mxvar)
      dimension ddd(maxnnz)
      dimension dfg(mxnr,0:mxnip)
      dimension dfo(mxorg,mxnr,0:mxnip)
      dimension dgd(mxnr,0:mxnip)
      dimension hocde(mxnr,0:mxnip,mxorg)
      dimension ia(neq+1)
      dimension ic(maxneq)
      dimension isp(lratio*maxnsp)
```

```
      dimension ja(nnz)
      dimension lambdr(mxvar)
      dimension nip(mxnr)
      dimension prfrac(mxnr,0:mxnip)
      dimension r(maxneq)
      dimension rsp(maxnsp)
      dimension sd3(maxvar)
      equivalence (isp(1),rsp(1))
c
      if (concen) then
c......read activity data from file
         do 10, ir=1,nr
            read(con,*,iostat=ioerr) cwt(ir,0),cswt(ir,0)
              if (ioerr .ne. 0) then
                  call ioreturn(ioerr)
                  return
              endif
   10    continue
      else
c...set ground-water concentrations to 0 for zero infiltration
c      case, because radionuclides cannot reach ground-water
         if (infil0) then
            do 20, ir=1,nr
               cwt(ir,0)=zero
               cswt(ir,0)=zero
   20       continue
c
         elseif (nlay .eq. 1) then
c...perform analytical solution  from eq. (4.13)
c......calculate the activity of each radionuclide at the beginning
c      of the time of interest, t
            call adoper(ci,cc,lambdr,dd,t,3*nr)
c
c......integrate the activity over the time period of the drinking water
c      scenario, ttd
            call sdoper(cc,sd3,lambdr,dd,ttd,3*nr)
c
c......calculate the activity for each radionuclide and pathway
c      in the chain
            do 30, ir=1,nr
               cwt(ir,0)=cc(3*ir)/vtd
               cswt(ir,0)=sd3(3*ir)/(vtd*ttd)
   30       continue
         else
c...perform numerical solution
c......calculate the average ground-water activity over
c      the period of interest from t to t+ttd
c......first, calculate integrated activity at end of current dose time
c      period, t+ttd
            do 40, ieq=1,neq
               adlnew(ieq)=adlold(ieq)
   40       continue
c
            istate=1
            told=t
            tnew=t+ttd
c
            call solvde(atol,adlnew,ia,istate,ja,neq,nnz,rtol,
     $         told,tnew)
```

```
c
            if (istate .lt. 0) then
               rc=0
               return
            endif
c
c......solve for average ground-water concentrations
c.........calculate difference in ground-water activities from current
c         time, t, to end of dose time period, t+ttd
            do 50, ieq=1,neq
               cdel(ieq)=adlnew(ieq)-adlold(ieq)
   50       continue
c
c......solve for integrated activities with sparse matrix solver
            do 60, ieq=1,neq
               r(ieq)=ieq
               c(ieq)=ieq
               ic(ieq)=ieq
   60       continue
c
            nsp=8*neq+2+2*nnz
c
            call cdrv(neq,r,c,ic,ia,ja,ddd,cdel,cint,
     $         nsp,isp,rsp,esp,5,flag)
c
            if (esp .lt. 0) then
               nsp=nsp+abs(esp)
               call cdrv(neq,r,c,ic,ia,ja,ddd,cdel,cint,
     $            nsp,isp,rsp,esp,5,flag)
            endif
c
            call cdrv(neq,r,c,ic,ia,ja,ddd,cdel,cint,
     $         nsp,isp,rsp,esp,2,flag)
            call cdrv(neq,r,c,ic,ia,ja,ddd,cdel,cint,
     $         nsp,isp,rsp,esp,4,flag)
c
c......calculate ground-water activity and average ground-water
c      activity from integrated actvitiy
            do 70, ir=1,nr
               cwt(ir,0)=adlold((ir-1)*nbox+nbox)/vtd
               cswt(ir,0)=cint((ir-1)*nbox+nbox)/(vtd*ttd)
   70       continue
         endif
      endif
c
c......calculate concentrations for implicit progeny
      do 90, ir=1,nr
         do 80, iip=0,nip(ir)
            cwt(ir,iip)=cwt(ir,0)*prfrac(ir,iip)
            cswt(ir,iip)=cswt(ir,0)*prfrac(ir,iip)
   80    continue
   90 continue
c
c...drinking water ingestion dose from eq. (4.21)
      do 120, ir=1,nr
         do 110, iip=0,nip(ir)
            dgd(ir,iip)=uw*td*dfg(ir,iip)*cswt(ir,iip)
            do 100, iorg=1,norg
               hocde(ir,iip,iorg)=uw*td*cswt(ir,iip)*dfo(iorg,ir,iip)
```

```
100        continue
110     continue
120 continue
    end
```

F.30 DRIMAX.FOR

```
c  This subroutine performs the maximum dose calculations
c  and updates the history file for selected time steps.


      include "cdecl.inc"

      subroutine drimax(out,rc,mxchn,mxlist,mxorg,mxpar,mxpth,mxpro,
     $   norg,npath,his,histry,conunt,import,distrb,
     $   ad,f1,f2,h1,h2,i,n1,n2,nlay,rho1,rho2,td,ttd,uw,vdd,
     $   dfer,dfes,dfg,dfh,dfo,kd,
     $   tstart,tend,dt,skip,
     $   nchn,chain,echain,conc,nuclid,nr,nip,nipdis,hlif,lambda,
     $   ipos,ipar,frac,prfrac,
     $   nlist,ttede,period,printed,persv,
     $   list,cwtsv,cswtsv,dossv,nucsv,pathsv,hocdsv,orgnsv,tedemx,
     $   dosmx,tdos,tpdos,pathmx,tpath,tppath,nucmx,tnuc,tpnuc,ioerr)

      character*6 list
      character*6 nuclid
      character*8 chain
      character*80 status_msg
      double precision ad,ad1new,ad1old,atol
      double precision ci,cswt,cswtm1,cswtsv,cswtup
      double precision cwt,cwtm1,cwtsv,cwtup
      double precision con,conc,cond
      double precision d,dd,ddd,dfer,dfes,dfg,dfh,dfo
      double precision dos,dosmx,dossv,dosup
      double precision dt
      double precision f1,f2,fd,frac
      double precision h1,h2,hlif,hocde,hocdup,hocdsv
      double precision i
      double precision kd
      double precision l,l12,l23,lambda,lambdr
      double precision n1,n2,nuc,nucmx,nucsv,nucup
      double precision orgnsm,orgnsv,orgnup
      double precision path,pathmx,pathup,pathsv
      double precision prfrac
      double precision rho1,rho2,rtol
      double precision smalli
      double precision tcalc,td,tdos,tede,tedemx,tedeup,tend
      double precision tnew,tnuc,told,tpath,tsav,tstart
      double precision tstop,ttd,ttede
      double precision uw
      double precision vdd,vtd
      double precision wd
      double precision zero
      integer achk
      integer chnnum,conunt
      integer his
      integer ia,ibox,ichn,ieq,indx,ioerr,ipar,ipos,ir
      integer istate
      integer ja
      integer maxbox,maxchn,maxlis,maxneq,maxnnz,maxorg
      integer maxpar,maxpth,maxpro,maxvar
      integer mxchn,mxlist,mxorg,mxpar,mxpth,mxpro
      integer nbox,nchn,neq,nip,nipdis,nlay,nlist,nnz,norg,npath,nr
```

```
integer out
integer period,printed,persv
integer rc
integer skip
integer tpdos,tpnuc,tppath,tsteps
logical distrb,done
logical dcreas
logical echain
logical histry
logical import,iszero,infil0
logical report
parameter (maxbox=12,maxchn=22,maxorg=21,maxpar=20,maxpro=6)
parameter (maxpth=7)
parameter (maxlis=maxchn*maxpar,maxvar=3*maxpar)
parameter (maxneq=maxbox*maxpar)
parameter (maxnnz=(maxbox*maxpar*(maxpar+3))/2)
parameter (smalli=1.0d-10)
parameter (zero=0.0d+0)
dimension chain(mxchn)
dimension adlnew(maxneq,maxchn),adlold(maxneq,maxchn)
dimension atol(maxchn)
dimension ci(3*maxpar,maxchn)
dimension con(maxpar,maxchn)
dimension conc(mxchn)
dimension cond(maxbox,maxpar,maxchn)
dimension cswt(maxpar,0:maxpro),cswtml(maxlis)
dimension cswtsv(mxlist),cswtup(maxlis)
dimension cwt(maxpar,0:maxpro),cwtml(maxlis)
dimension cwtsv(mxlist),cwtup(maxlis)
dimension d(maxpar,maxpar,maxchn)
dimension dd(maxvar,maxvar,maxchn)
dimension ddd(maxnnz,maxchn)
dimension dfer(mxpar,0:mxpro,mxchn),dfes(mxpar,0:mxpro,mxchn)
dimension dfg(mxpar,0:mxpro,mxchn),dfh(mxpar,0:mxpro,mxchn)
dimension dfo(mxorg,mxpar,0:mxpro,mxchn)
dimension dos(maxpar,0:maxpro,maxpth)
dimension dosmx(mxlist,mxpth)
dimension dossv(mxlist,mxpth),dosup(maxlis,maxpth)
dimension echain(mxchn)
dimension frac(2,mxpar,mxchn)
dimension hocde(maxpar,0:maxpro,maxorg)
dimension hocdsv(mxlist,mxorg),hocdup(maxlis,maxorg)
dimension hlif(mxpar,mxchn)
dimension ia(maxneq+1,maxchn)
dimension indx(maxpar,0:maxpro,maxchn)
dimension ipar(2,mxpar,mxchn)
dimension ipos(mxpar,mxchn)
dimension ja(maxnnz,maxchn)
dimension kd(mxpar,mxchn)
dimension l(maxbox,maxpar,maxchn)
dimension l12(maxpar,maxchn),l23(maxpar,maxchn)
dimension lambda(mxpar,mxchn)
dimension lambdr(maxvar,maxchn)
dimension list(mxlist)
dimension neq(maxchn)
dimension nip(mxpar,mxchn),nipdis(mxpar,mxchn)
dimension nr(mxchn)
dimension nnz(maxchn)
dimension nuc(maxpar,0:maxpro)
```

```
      dimension nuclid(mxpar,0:mxpro,mxchn)
      dimension nucmx(mxlist),nucsv(mxlist),nucup(maxlis)
      dimension orgnsm(maxorg),orgnsv(mxorg),orgnup(maxorg)
      dimension path(maxpth),pathmx(mxpth)
      dimension pathsv(mxpth),pathup(maxpth)
      dimension prfrac(mxpar,0:mxpro,mxchn)
      dimension rtol(maxchn)
      dimension tdos(mxlist,mxpth)
      dimension tnuc(mxlist)
      dimension tpath(mxpth)
      dimension tpdos(mxlist,mxpth)
      dimension tpnuc(mxlist)
      dimension tppath(mxpth)
      common /ddd/ ddd
      common /ia/ ia
      common /ja/ ja
      common /chnnum/ chnnum
c
      tsteps=(tend-tstart)/dt
c
      if ((dt .eq. zero) .or. ((tend-tstart) .le. 100.0d+0)) then
         achk=10
      else
         achk=tsteps/20
      endif
      if (achk .eq. 0) achk=10
c
c...determine if infilration rate is small enough to be 0
      if (i .le. smalli) then
         infil0=.true.
      else
         infil0=.false.
      endif
c
c...perform preliminary calculations
      nbox=nlay+2

c...if import concentration file is used
      if (import) then
         do 10, ichn=1,nchn
c......set up the initial activities for each radionuclide in the chain
            call initzd(mxpar,con(1,ichn))
c
c......write chain, cede and activity data to output file
            call wrchn(out,mxpar,mxpro,nr(ichn),nip(1,ichn),ichn,
     $         chain(ichn),echain(ichn),nuclid(1,0,ichn),ipos(1,ichn),
     $         ipar(1,1,ichn),frac(1,1,ichn),hlif(1,ichn),
     $         prfrac(1,0,ichn),con(1,ichn),dfg(1,0,ichn),
     $         dfh(1,0,ichn),dfes(1,0,ichn),dfer(1,0,ichn),'DRINK',
     $         ioerr)
c
            if (ioerr .ne. 0) then
               return
            endif
c
c......convert dfer, dfes, dfg, and dfh from Bq-Sv system to
c      mrem-H-pCi system
            call convrt(mxpar,mxpro,nr(ichn),nip(1,ichn),
     $         dfg(1,0,ichn),dfh(1,0,ichn),
```

```
     $          dfes(1,0,ichn),dfer(1,0,ichn))
c
c......convert organ dose from Sv/Bq system to mrem/pCi
            call dfocon(mxpro,mxpar,mxorg,nip(1,ichn),norg,nr(ichn),
     $          dfo(1,1,0,ichn))
   10     continue
c
      else
         do 20, ichn=1,nchn
c......set up the initial activities for each radionuclide in the chain
            call concen(mxpar,mxpro,nr(ichn),echain(ichn),distrb,
     $          lambda(1,ichn),ipar(1,1,ichn),frac(1,1,ichn),
     $          prfrac(1,0,ichn),nipdis(1,ichn),conc(ichn),con(1,ichn))
c
c......write chain, cede and activity data to output file
            call wrchn(out,mxpar,mxpro,nr(ichn),nip(1,ichn),ichn,
     $          chain(ichn),echain(ichn),nuclid(1,0,ichn),ipos(1,ichn),
     $          ipar(1,1,ichn),frac(1,1,ichn),hlif(1,ichn),
     $          prfrac(1,0,ichn),con(1,ichn),dfg(1,0,ichn),
     $          dfh(1,0,ichn),dfes(1,0,ichn),dfer(1,0,ichn),'DRINK',
     $          ioerr)
c
              if (ioerr .ne. 0) then
                 return
              endif
c
c......convert dfer, dfes, dfg, and dfh from Bq-Sv system to
c      mrem-H-pCi system
            call convrt(mxpar,mxpro,nr(ichn),nip(1,ichn),
     $          dfg(1,0,ichn),dfh(1,0,ichn),
     $          dfes(1,0,ichn),dfer(1,0,ichn))
c
c......set up the fractional yield array
            call frctnl(mxpar,nr(ichn),d(1,1,ichn),
     $          frac(1,1,ichn),ipar(1,1,ichn))
c
c......calculate transfer coefficients between boxes
            call trnsfr(maxbox,mxpar,nbox,nlay,nr(ichn),h1,h2,n1,n2,
     $          f1,f2,rho1,rho2,zero,i,kd(1,ichn),ad,vdd,vtd,fd,wd,
     $          l(1,1,ichn))
c
c......set up initial radionuclide activities into format
c      useable by n-box model
            call conres(maxbox,mxpar,nbox,nr(ichn),con(1,ichn),
     $          cond(1,1,ichn))
            if (nlay .eq. 1) then
c
c......set up "decay" constants for drinking water scenario
               do 200, ir=1,nr(ichn)
                  l12(ir,ichn)=l(1,ir,ichn)
                  l23(ir,ichn)=l(2,ir,ichn)
  200          continue
c
               call newlam(maxpar,maxvar,l12(1,ichn),l23(1,ichn),
     $            lambda(1,ichn),nr(ichn),wd,lambdr(1,ichn))
c
c......set up dd array in matrix form
c      for use by drinking water scenario 3-box model
               call ddaray(maxpar,maxvar,d(1,1,ichn),l12(1,ichn),
```

```
     $            123(1,ichn),lambda(1,ichn),nr(ichn),dd(1,1,ichn))
c
            else
               call sparse(d(1,1,ichn),fd,l(1,1,ichn),
     $            lambda(1,ichn),nbox,nr(ichn),ddd(1,ichn),
     $            ia(1,ichn),ja(1,ichn),neq(ichn),nnz(ichn))
c
               call toler(cond(1,1,ichn),maxneq,
     $            atol(ichn),rtol(ichn))
            endif
c
c......convert organ dose from Sv/Bq system to mrem/pCi
            call dfocon(mxpro,mxpar,mxorg,nip(1,ichn),norg,nr(ichn),
     $         dfo(1,1,0,ichn))
   20    continue
      endif
c
c...combine radionnuclides from all chains into a unique list of
c   radionuclides and tie each radionuclide from each chain into
c   an index number in the list
      call nuclis(mxlist,mxchn,mxpar,mxpro,
     $   nlist,nchn,nr,nip,nuclid,list,indx)
c
      printed=zero
c
      status_msg="Calculating up to                "
      write(status_msg(19:23),'(i4)',iostat=ioerr) tsteps
c
      if (ioerr .ne. 0) then
          call ioretint(ioerr)
          return
      endif
c
      status_msg=status_msg(1:24)//' timesteps...'C
      call CMessage(status_msg,rc)
c
c...calculate doses
c......initialize some time parameters
      call init0(out,rc,conunt,import,ttd,tstart,tend,
     $   period,tcalc,tstop,ioerr)
c
      if (ioerr .ne. 0) then
          return
      endif
c
          if (rc .eq. 0) then
              return
          endif
c
c......initialize maximum values of comparison variables
      call initzd(mxlist*mxpth,dosmx)
      call initzd(mxlist,nucmx)
      call initzd(mxpth,pathmx)
      call initzd(1,tedemx)
      call initzd(mxlist,cwtml)
      call initzd(mxlist,cswtml)
      call initlg(1,.false.,dcreas)
c
c...find concentrations at start time
```

```
c......if numerical model is used
      if (nlay .gt. 1) then
         do 50, ichn=1,nchn
            chnnum=ichn
            do 40, ir=1,nr(ichn)
               do 30, ibox=1,nbox
                  ad1old((ir-1)*nbox+ibox,ichn)=cond(ibox,ir,ichn)
   30          continue
   40       continue
c
            told=zero
            tnew=tstart
            istate=1
c
            call solvde(atol(ichn),ad1old(1,ichn),ia(1,ichn),istate,
     $         ja(1,ichn),neq(ichn),nnz(ichn),rtol(ichn),told,
     $         tnew)
c
            if (istate .lt. 0) then
                  rc=0
                  return
            endif
   50    continue
c
c......if analytical model is used
      else
         do 80, ichn=1,nchn
            do 70, ir=1,nr(ichn)
               do 60,ibox=1,nbox
                  ci((ir-1)*3+ibox,ichn)=cond(ibox,ir,ichn)
   60         continue
   70       continue
   80    continue
      endif
c
   85 continue
c
c......determine if current calculation time is a report time
      call rpttim(tcalc,tstop,period,skip,histry,report)
c
c......calculate dose for each radionuclide in each chain
c      and for each pathway
c......initialize update variables for the current time step
      call initzd(mxlist*mxpth,dosup)
      call initzd(mxlist*mxorg,hocdup)
      call initzd(mxlist,nucup)
      call initzd(mxpth,pathup)
      call initzd(mxorg,orgnup)
      call initzd(mxlist,cwtup)
      call initzd(mxlist,cswtup)
      call initzd(1,tedeup)
c
      do 90, ichn=1,nchn
c.........calculate dose in chain for each radionuclide
c         for each pathway
         chnnum=ichn
c
         call dridos(out,rc,mxpar,mxpro,mxorg,maxvar,conunt,ci(1,ichn),
     $      import,infil0,atol(ichn),ad1old(1,ichn),
```

```
     $       dd(1,1,ichn),ddd(1,ichn),dfg(1,0,ichn),dfo(1,1,0,ichn),
     $       ia(1,ichn),ja(1,ichn),
     $       lambdr(1,ichn),nbox,neq(ichn),nip(1,ichn),nlay,nnz(ichn),
     $       norg,nr(ichn),prfrac(1,0,ichn),rtol(ichn),
     $       tcalc,td,ttd,uw,vtd,
     $       cwt,cswt,adlnew(1,ichn),dos(1,0,1),hocde,ioerr)
c
          if (ioerr .ne. 0) then
              return
          endif
c
         if (rc .eq. 0) then
              return
         endif
c
c.........calculate pathway dose for chain
         call pthsum(mxpar,mxpro,mxpth,nr(ichn),nip(1,ichn),npath,
     $       dos,path)
c
c.........calculate total organ dose from each radionuclide in chain
         call pthsum(mxpar,mxpro,mxorg,nr(ichn),nip(1,ichn),norg,
     $       hocde,orgnsm)
c
c.........calculate radionuclide dose for chain
         call nucsum(mxpar,mxpro,mxpth,nr(ichn),nip(1,ichn),npath,
     $       dos,nuc)
c
c.........calculate tede for chain
         call tedsum(mxpar,mxpro,nr(ichn),nip(1,ichn),nuc,tede)
c
c.........update listed radionuclide soil activity
         call upsbar(mxpar,mxpro,mxlist,nr(ichn),nip(1,ichn),
     $       indx(1,0,ichn),cwt,cwtup)
c
c.........update listed radionuclide average groundwater activity
         call upsbar(mxpar,mxpro,mxlist,nr(ichn),nip(1,ichn),
     $       indx(1,0,ichn),cswt,cswtup)
c
c.........update listed radionuclide dose for each pathway for all chains
         call updose(mxpar,mxpro,mxpth,mxlist,
     $       nr(ichn),nip(1,ichn),npath,
     $       indx(1,0,ichn),dos,dosup)
c
c.........update listed radionuclide dose for each organ for all chains
         call updose(mxpar,mxpro,mxorg,mxlist,
     $       nr(ichn),nip(1,ichn),norg,
     $       indx(1,0,ichn),hocde,hocdup)
c
c.........update pathway dose for all chains
         call uppath(mxpth,npath,path,pathup)
c
c.........update organ dose for all chains
         call uppath(mxorg,norg,orgnsm,orgnup)
c
c.........update listed radionuclide dose for all chains
         call upnuc(mxpar,mxpro,mxlist,nr(ichn),nip(1,ichn),
     $       indx(1,0,ichn),nuc,nucup)
c
c.........update total dose for all chains
```

```
         call uptede(tede,tedeup)
   90 continue
c
c...if this is a report time write results to history file
      if (report) then
         call wrhist(his,mxlist,mxpth,nlist,npath,
     $               tcalc,tcalc+ttd,list,cwtup,cswtup,dosup,pathup,
     $               nucup,tedeup,ioerr)
c
          if (ioerr .ne. 0) then
              return
          endif
c
         printed=printed+1
      endif
c
c
c......if integrated activities are increasing, then test for
c      maximum values of total dose (tede) and find its components,
c      pathway dose, listed radionuclide dose, and listed
c      radionuclide dose by pathway
      if (.not. dcreas) then
c
c......check if total tede is maximum so far; if so, save it,
c      its components, and its time of occurrence
         call tedmxd(mxlist,mxorg,mxpth,nlist,norg,npath,tcalc,
     $      period,persv,
     $      ttede,tedeup,tedemx,pathsv,pathup,nucsv,nucup,dossv,dosup,
     $      cwtsv,cwtup,cswtsv,cswtup,hocdsv,hocdup,orgnsv,orgnup)
c
c......check for maximum pathway doses; if so, save them and their
c      time of occurrence
         call pthmax(mxpth,npath,tcalc,period,tpath,tppath,
     $      pathup,pathmx)
c
c......check for maximum listed radionuclide doses; if so, save
c      them and their time of occurrence
         call nucmax(mxlist,nlist,tcalc,period,tnuc,tpnuc,
     $      nucup,nucmx)
c
c......check for maximum listed radionuclide doses per pathway; if so,
c      save them and their time of occurence
         call dosmax(mxpth,mxlist,npath,nlist,tcalc,period,dosup,
     $      tdos,tpdos,dosmx)
      endif
c
c......determine if the soil activities, soil integrated activities,
c      are and water integrated activities are decreasing. if they
c      are all decreasing, set a test value to true and check history
c      value to determine if maximum results should be written out at
c      this time
      iszero=.true.
c
      call initlg(1,.true.,dcreas)
      call valtst(nlist,cwtup,cwtml,dcreas,iszero)
      call valtst(nlist,cswtup,cswtml,dcreas,iszero)
c
      if (dcreas .and. .not. histry) then
         status_msg="All doses are decreasing..."C
```

```
         call CMessage(status_msg,rc)
         goto 125
      endif
c
      if (dcreas .and. iszero) then
         status_msg="All doses are decreasing and under minimum..."C
         call CMessage(status_msg,rc)
         goto 125
      endif
c
      if (tcalc .ge. tstop) goto 125
      if (infil0) goto 125
c
c......update time
      tsav=tcalc
      call uptime(conunt,import,done,period,tcalc,dt,tstop,ioerr)
c
      if (ioerr .ne. 0) then
          return
      endif
c
      if (mod(period,achk) .eq. 1) then
         write(status_msg(1:10),'(i10)',iostat=ioerr) period
          if (ioerr .ne. 0) then
              call ioretint(ioerr)
              return
          endif
c
         status_msg=status_msg(1:10)//' timestep(s) out of 'C
         write(status_msg(32:36),'(i4)',iostat=ioerr) tsteps
c
          if (ioerr .ne. 0) then
              call ioretint(ioerr)
              return
          endif
c
         status_msg=status_msg(1:36)//' calculated...'C
         call CMessage(status_msg,rc)
c
         if (rc .eq. 0) then
            return
         endif
c
      else
         call CProcess(rc)
         if (rc .eq. 0) then
            return
         endif
      endif
c
      if (done) goto 125
c
c......for numerical solution, update activities if another
c      time step will occur
      if (.not. import) then
         if (nlay .gt. 1) then
            if (dt .ne. ttd .or. tcalc .lt. told+dt) then
               do 100,ichn=1,nchn
                  told=tsav
```

```
                  tnew=tcalc
                  chnnum=ichn
                  istate=1
c
                  call solvde(atol(ichn),ad1old(1,ichn),ia(1,ichn),
     $               istate,
     $               ja(1,ichn),neq(ichn),nnz(ichn),rtol(ichn),
     $               told,tnew)
c
                  if (istate .lt. 0) then
                     rc=0
                     return
                  endif
  100          continue
c
            else
               do 120,ichn=1,nchn
                  do 110, ieq=1,neq(ichn)
                     ad1old(ieq,ichn)=ad1new(ieq,ichn)
  110             continue
  120          continue
            endif
         endif
      endif
c
      goto 85
  125 continue
c
      write(status_msg,'(i10)',iostat=ioerr) period
c
      if (ioerr .ne. 0) then
          call ioretint(ioerr)
          return
      endif
c
      status_msg=status_msg(1:10)//
     $   ' timesteps were actually calculated...'C
      call CMessage(status_msg,rc)
c
      end
```

F.31 DRINK.FOR

```
c
c...this subroutine calculates the dose due ingestion of drinking
c   water using the first part of Eqn. 5.74 and Eqn. 5.75
c
      subroutine drink(nrmax,npmax,awt,dfg,nip,nr,prfrac,td,uw,dwr1)

      double precision awt
      double precision dfg,dwr1
      double precision factor
      double precision prfrac
      double precision td
      double precision uw
      integer j
      integer k
      integer nip,npmax,nr,nrmax
      dimension awt(nrmax)
      dimension dwr1(nrmax,0:npmax)
      dimension dfg(nrmax,0:npmax)
      dimension nip(nrmax)
      dimension prfrac(nrmax,0:npmax)
c
c...calculate drinking water dose
      factor=uw*td
      do 20, j=1,nr
         do 10, k=0,nip(j)
            dwr1(j,k)=awt(j)*dfg(j,k)*prfrac(j,k)*factor
   10    continue
   20 continue
      end
```

F.32 DRIOUT.FOR

```
c
c...this subroutine writes out the user specified input data
c   for the drinking water scenario

c
      subroutine driout(out,mxchn,adate,atime,short,long,title,
     $   histry,combin,import,distrb,confil,
     $   ad,f1,f2,h1,h2,i,n1,n2,nlay,rho1,rho2,td,ttd,uw,vdd,
     $   tstart,tend,dt,skip,
     $   nchn,chain,conc,ioerr,nchnp2)

      character*6 short
      character*10 adate
      character*8 chain
      character*11 atime
      character*27 long
      character*80 confil
      character*80 title
      integer ichn,ioerr
      integer loc,loc1,loc2
      integer mxchn
      integer nchn,nchnp2,nlay
      integer out
      integer skip
      logical combin
      logical distrb
      logical equil
      logical histry
      logical import
      double precision ad
      double precision conc
      double precision dt
      double precision f1,f2
      double precision h1,h2
      double precision i
      double precision n1,n2
      double precision rho1,rho2
      double precision td,tend,tstart,ttd
      double precision uw
      double precision vdd
      dimension chain(mxchn)
      dimension conc(mxchn)
c
      write(out,'(3a)',iostat=ioerr) 'Output From Program ''',short,''''

      if (ioerr .ne. 0) then
        call ioreturn(ioerr)
        return
      endif

      write(out,'(a)',iostat=ioerr) long

      if (ioerr .ne. 0) then
        call ioreturn(ioerr)
        return
      endif
```

```
      write(out,'(2a)',iostat=ioerr) 'Run Date: ',adate

      if (ioerr .ne. 0) then
        call ioreturn(ioerr)
        return
      endif

      write(out,'(2a)',iostat=ioerr) 'Run Time: ',atime

      if (ioerr .ne. 0) then
        call ioreturn(ioerr)
        return
      endif

      write(out,'(a)',iostat=ioerr) 'INPUT DATA:'

      if (ioerr .ne. 0) then
        call ioreturn(ioerr)
        return
      endif

      write(out,'(2a)',iostat=ioerr) 'Title: ',title

      if (ioerr .ne. 0) then
        call ioreturn(ioerr)
        return
      endif

      write(out,'(a)',iostat=ioerr) 'Notes: '

      if (ioerr .ne. 0) then
        call ioreturn(ioerr)
        return
      endif

      if (histry) then
         write(out,'(a)',iostat=ioerr) 'History file will be generated.'

          if (ioerr .ne. 0) then
              call ioreturn(ioerr)
              return
          endif

      else
         write(out,'(a)',iostat=ioerr)
     $      'History file will not be generated.'

          if (ioerr .ne. 0) then
              call ioreturn(ioerr)
              return
          endif

      endif


      if (combin) then
         write(out,'(2a)',iostat=ioerr) 'Implicit progeny doses will',
     $      ' be included with explicit parent.'
```

```
      if (ioerr .ne. 0) then
          call ioreturn(ioerr)
          return
      endif

   else
      write(out,'(2a)',iostat=ioerr)
 $       'Implicit progeny doses will not',
 $       ' be included with explicit parent.'

      if (ioerr .ne. 0) then
          call ioreturn(ioerr)
          return
      endif

   endif

   if (import) then
      write(out,'(4a)',iostat=ioerr)
 $       'Concentration data will be read ',
 $       'from file: ',confil,'.'

      if (ioerr .ne. 0) then
          call ioreturn(ioerr)
          return
      endif

   else
      write(out,'(a)',iostat=ioerr)
 $       'Concentration data will be calculated.'

      if (ioerr .ne. 0) then
          call ioreturn(ioerr)
          return
      endif

      equil=.false.
      do 10, ichn=1,nchn

         if (chain(ichn) .eq. 'U_Nat') then
            loc=0
         elseif(chain(ichn) .eq. 'Th_Nat') then
            loc=6
         else
            loc1=max(index(chain(ichn),'+C'),index(chain(ichn),'+c'))
            loc2=max(index(chain(ichn),'_C'),index(chain(ichn),'_c'))
            loc=max(loc1,loc2)
         endif

         if (loc .gt. 0) equil=.true.

10    continue
      if (equil) then
         write(out,'(2a)',iostat=ioerr)
 $          'Concentrations of radionuclides in ',
 $          'equilibrium chains will be'

           if (ioerr .ne. 0) then
               call ioreturn(ioerr)
```

```
            return
          endif

        if (distrb) then
           write(out,'(2a)',iostat=ioerr)
$             '   distributed based on initial ',
$             'concentration given for parent.'

              if (ioerr .ne. 0) then
                  call ioreturn(ioerr)
                  return
              endif

        else
           write(out,'(2a)',iostat=ioerr)
$             '   equilibrated based on initial ',
$             'concentration of parent.'

              if (ioerr .ne. 0) then
                  call ioreturn(ioerr)
                  return
              endif

        endif
     endif
  endif

  write(out,'(a)',iostat=ioerr) ' '

  if (ioerr .ne. 0) then
    call ioreturn(ioerr)
    return
  endif

  write(out,'(a)',iostat=ioerr) 'PARAMETER DATA:'

  if (ioerr .ne. 0) then
    call ioreturn(ioerr)
    return
  endif

  write(out,'(/a,t60,a,1p,e10.4,a)',iostat=ioerr)
$   'Drinking Water Ingestion Rate',': ',uw,' L/d'

  if (ioerr .ne. 0) then
    call ioreturn(ioerr)
    return
  endif

  write(out,'(a,t60,a,1p,e10.4,a)',iostat=ioerr)
$   'Drinking Water Consumption Period',': ',td,' d/y'

  if (ioerr .ne. 0) then
    call ioreturn(ioerr)
    return
  endif

  write(out,'(a,t60,a,1p,e10.4,a)',iostat=ioerr)
$   'Drinking Water Scenario Exposure Period',': ',ttd,' d'
```

```
 if (ioerr .ne. 0) then
   call ioreturn(ioerr)
   return
 endif

 write(out,'(a,t60,a,1p,e10.4,a)',iostat=ioerr)
$   'Surface Soil Layer Thickness',': ',h1,' m'

 if (ioerr .ne. 0) then
   call ioreturn(ioerr)
   return
 endif

 write(out,'(a,t60,a,1p,e10.4,a)',iostat=ioerr)
$   'Unsaturated Zone Layer Thickness',': ',h2,' m'

 if (ioerr .ne. 0) then
   call ioreturn(ioerr)
   return
 endif

 write(out,'(a,t60,a,1p,i2)',iostat=ioerr)
$   'Number of Unsaturated Zone Layers',': ',nlay

 if (ioerr .ne. 0) then
   call ioreturn(ioerr)
   return
 endif

 write(out,'(a,t60,a,1p,e10.4)',iostat=ioerr)
$   'Surface Soil Porosity',': ',n1

 if (ioerr .ne. 0) then
   call ioreturn(ioerr)
   return
 endif

 write(out,'(a,t60,a,1p,e10.4)',iostat=ioerr)
$   'Unsaturated Zone Porosity',': ',n2

 if (ioerr .ne. 0) then
   call ioreturn(ioerr)
   return
 endif

 write(out,'(a,t60,a,1p,e10.4)',iostat=ioerr)
$   'Surface Soil Saturation',': ',f1

 if (ioerr .ne. 0) then
   call ioreturn(ioerr)
   return
 endif

 write(out,'(a,t60,a,1p,e10.4)',iostat=ioerr)
$   'Unsaturated Zone Saturation',': ',f2

 if (ioerr .ne. 0) then
   call ioreturn(ioerr)
```

```
      return
    endif

    write(out,'(a,t60,a,1p,e10.4,a)',iostat=ioerr)
   $   'Surface Soil Bulk Soil Density',': ',rhol,' g/cm^3'

    if (ioerr .ne. 0) then
      call ioreturn(ioerr)
      return
    endif

    write(out,'(a,t60,a,1p,e10.4,a)',iostat=ioerr)
   $   'Unsaturated Zone Bulk Soil Density',': ',rho2,' g/cm^3'

    if (ioerr .ne. 0) then
      call ioreturn(ioerr)
      return
    endif

    write(out,'(a,t60,a,1p,e10.4,a)',iostat=ioerr)
   $   'Water Volume Removed From Aquifer Per Year',': ',vdd,' L'

    if (ioerr .ne. 0) then
      call ioreturn(ioerr)
      return
    endif

    write(out,'(a,t60,a,1p,e10.4,a)',iostat=ioerr)
   $   'Infiltration Rate',': ',i,' m/y'

    if (ioerr .ne. 0) then
      call ioreturn(ioerr)
      return
    endif

    write(out,'(a,t60,a,1p,e10.4,a)',iostat=ioerr)
   $   'Area of Contaminated Site',': ',ad,' m^2'

    if (ioerr .ne. 0) then
      call ioreturn(ioerr)
      return
    endif

    write(out,'(a)',iostat=ioerr) 'TIME DATA:'

    if (ioerr .ne. 0) then
      call ioreturn(ioerr)
      return
    endif

    write(out,'(a,t60,a,1p,e10.4,a)',iostat=ioerr)
   $   'Start Time',': ',tstart,' d'

    if (ioerr .ne. 0) then
      call ioreturn(ioerr)
      return
    endif

    write(out,'(a,t60,a,1p,e10.4,a)',iostat=ioerr)
```

```
$   'End Time',': ',tend,' d'

 if (ioerr .ne. 0) then
   call ioreturn(ioerr)
   return
 endif

 write(out,'(a,t60,a,1p,e10.4,a)',iostat=ioerr)
$   'DT Size',': ',dt,' d'

 if (ioerr .ne. 0) then
   call ioreturn(ioerr)
   return
 endif

 write(out,'(a,t60,a,1p,e10.4,a)',iostat=ioerr)
$   'Time Step Size',': ',ttd,' d'

 if (ioerr .ne. 0) then
   call ioreturn(ioerr)
   return
 endif

 write(out,'(a,t60,a,i5,a)',iostat=ioerr)
$   'Write results every',': ',skip,' calculation times'

 if (ioerr .ne. 0) then
   call ioreturn(ioerr)
   return
 endif

 write(out,'(a)',iostat=ioerr) ' '

 if (ioerr .ne. 0) then
   call ioreturn(ioerr)
   return
 endif

 write(out,'(a)',iostat=ioerr) 'INITIAL ACTIVITIES:'

 if (ioerr .ne. 0) then
   call ioreturn(ioerr)
   return
 endif

 write(out,'(/a,i3)',iostat=ioerr) 'Number of chains: ',nchnp2

 if (ioerr .ne. 0) then
   call ioreturn(ioerr)
   return
 endif

 write(out,'(/a)',iostat=ioerr) ' Chain       Chain      Initial'

 if (ioerr .ne. 0) then
   call ioreturn(ioerr)
   return
 endif
```

```
      write(out,'(a)',iostat=ioerr) 'Number        Name     Activity'

      if (ioerr .ne. 0) then
        call ioreturn(ioerr)
        return
      endif

      write(out,'(a)',iostat=ioerr) '                                (pCi)'

      if (ioerr .ne. 0) then
        call ioreturn(ioerr)
        return
      endif

      write(out,'(i6,2x,a,2x,1p,e10.4)',iostat=ioerr)
     $   (ichn,chain(ichn),conc(ichn),ichn=1,nchn)

      if (ioerr .ne. 0) then
        call ioreturn(ioerr)
        return
      endif

      end
```

F.33 DRMAIN.FOR

```
c*********************************************************************
c
c  PROGRAM DRWATER; 11/94
c  C. D. Updegraff, GRAM, Inc. Albuquerque, NM 87112.
c  Solves the drinking water scenario equations from Kennedy, Jr., W.E.
c  and D.L. Strenge, 1992, "Residual Radioactive Contamination From
c  Decommissioning: Technical Basis for Translating Contamination
c  Levels to Annual Total Effective Dose Equivalent," NUREG/CR-5512,
c  PNL-7994, Vol. 1, Pacific Northwest Laboratory, Richland, WA 99352.
c  Benchmarked against RADDECAY Version 3.01 11/92 for the U-238 chain
c
c*********************************************************************
c


      include "cdecl.inc"

      subroutine drwater(for_rc)

      character*2 elemnt
      character*6 list
      character*6 nuclid,nucnam,nucrd
      character*6 short
      character*10 adate
      character*8 chain
      character*11 atime
      character*27 long
      character*80 filfil,drfil
      character*80 chainfil,cedefil,org1,org2,org3,kdfil
      character*80 confil
      character*80 hisfil
      character*80 infil
      character*80 outfil
      character*80 title
      character*80 status_msg
      double precision ad
      double precision conc
      double precision cwtsv,cswtsv
      double precision dfer,dfes,dfg,dfh,dfo,dosmx,dossv
      double precision dt
      double precision f1,f2,frac
      double precision h1,h2,hlif,hocdsv
      double precision i
      double precision kd
      double precision lambda
      double precision n1,n2,nucmx,nucsv
      double precision orgnsv
      double precision pathmx,pathsv,prfrac
      double precision rho1,rho2
      double precision td,tdos,tedemx,tend,ttd
      double precision tnuc,tpath,tstart,ttede
      double precision uw
      double precision vdd
      double precision zero
      integer filefn,infn,outfn,confn,hisfn
      integer chainfn,parfn,cedefn,kdfn,orgfn,org1fn,org2fn,org3fn
      integer ichn,ierr,ioerr,iorg,ipar,ipos
```

```
integer maxchn,maxlis,maxnam,maxorg,maxpar,maxpth,maxpro
integer nchn,nchnp2,nip,nipdis,nlay,nlist,nnam,norg,npath,nr,nrrd
integer period,printed,persv,tppath,tpdos,tpnuc
integer skip
integer*4 for_rc
integer*4 c_rc
integer*2 month,day,year
integer*2 hour,minute,second,hsecond
logical combin
logical distrb
logical echain,exists
logical histry
logical import
parameter (maxchn=22,maxorg=21,maxnam=1000,maxpar=20)
parameter (maxpth=7,maxpro=6)
parameter (npath=1,norg=21)
parameter (maxlis=maxchn*maxpar)
parameter (zero=0.0d+0)
dimension chain(maxchn)
dimension conc(maxchn)
dimension cwtsv(maxlis),cswtsv(maxlis)
dimension dfer(maxpar,0:maxpro,maxchn)
dimension dfes(maxpar,0:maxpro,maxchn)
dimension dfg(maxpar,0:maxpro,maxchn),dfh(maxpar,0:maxpro,maxchn)
dimension dfo(maxorg,maxpar,0:maxpro,maxchn)
dimension dosmx(maxlis,maxpth)
dimension dossv(maxlis,maxpth)
dimension elemnt(maxpar)
dimension echain(maxchn)
dimension frac(2,maxpar,maxchn)
dimension hlif(maxpar,maxchn)
dimension hocdsv(maxlis,maxorg)
dimension ipar(2,maxpar,maxchn)
dimension ipos(maxpar,maxchn)
dimension kd(maxpar,maxchn)
dimension lambda(maxpar,maxchn)
dimension list(maxlis)
dimension nip(maxpar,maxchn),nipdis(maxpar,maxchn)
dimension nr(maxchn),nrrd(maxchn)
dimension nuclid(maxpar,0:maxpro,maxchn)
dimension nucmx(maxlis),nucsv(maxlis)
dimension nucrd(maxpar,maxchn)
dimension orgnsv(maxorg)
dimension pathmx(maxpth)
dimension pathsv(maxpth)
dimension prfrac(maxpar,0:maxpro,maxchn)
dimension tdos(maxlis,maxpth)
dimension tnuc(maxlis)
dimension tpath(maxpth)
dimension tppath(maxpth)
dimension tpdos(maxlis,maxpth)
dimension tpnuc(maxlis)
data long /'Drinking Water Scenario'/
data short /'DRINK'/

for_rc = 1
c_rc = 1
filefn  = 50
```

```
      status_msg(1:80) = "Starting Calculational Modules..."C
      call CMessage(status_msg,c_rc)
      if (c_rc .eq. 0) then
            return
      endif

CCC...converted to Microsoft Powerstation FORTRAN Date and Time functions
      call GetDat(year,month,day)
      write(adate,'(I2.2,A,I2.2,A,I4.4)',iostat=ioerr)
     $   month, '/', day, '/', year

      if (ioerr .ne. 0) then
            for_rc = 0
            call ioretint(ioerr)
            call closefiles(filefn,20)
            return
      endif

      call GetTim(hour,minute,second,hsecond)
      write(atime,'(I2.2,A,I2.2,A,I2.2)',iostat=ioerr)
     $   hour, ':', minute, ':',
     $         second

      if (ioerr .ne. 0) then
            for_rc = 0
            call ioretint(ioerr)
            call closefiles(filefn,20)
            return
      endif

      status_msg(1:80) = "Reading Input Filenames..."C
      call CMessage(status_msg,c_rc)

      if (c_rc .eq. 0) then
            call closefiles(filefn,20)
            return
      endif

      filfil   = "data\dwfnam"
      chainfil = "data\chain.dat"
      cedefil  = "data\cede.dat"
      drfil    = "data\dw.dat"
      org1     = "data\organ1.dat"
      org2     = "data\organ2.dat"
      org3     = "data\organ3.dat"
      kdfil    = "data\kd.dat"

c     assign the logical file units
      filefn  = 50
      infn    = filefn + 1
      outfn   = filefn + 2
      hisfn   = filefn + 3
      confn   = filefn + 4
      chainfn = filefn + 5
      parfn   = filefn + 6
      cedefn  = filefn + 7
      kdfn    = filefn + 8
      orgfn   = filefn + 9
      org1fn  = orgfn + 1
```

```
      org2fn  = orgfn + 2
      org3fn  = orgfn + 3

c
c...read the input file number and name
      OPEN(filefn,FILE=filfil,STATUS='OLD',MODE='READ',
     $     SHARE='DENYNONE',iostat=ioerr)

      if (ioerr .ne. 0) then
            for_rc = 0
            call ioreturn(ioerr)
            call closefiles(filefn,20)
            return
      endif

      read(filefn,*,iostat=ioerr) infil,outfil,hisfil,confil

      if (ioerr .ne. 0) then
            for_rc = 0
            call ioreturn(ioerr)
            call closefiles(filefn,20)
            return
      endif

      CLOSE(filefn,iostat=ioerr)

      if (ioerr .ne. 0) then
            for_rc = 0
            call ioreturn(ioerr)
            call closefiles(filefn,20)
            return
      endif


c
c...open output file
      open(outfn,file=outfil,status='unknown',mode='write',
     $     SHARE='DENYNONE',iostat=ioerr)

      if (ioerr .ne. 0) then
            for_rc = 0
            call ioreturn(ioerr)
            call closefiles(filefn,20)
            return
      endif

c

      status_msg(1:80) = "Reading User Input..."C
      call CMessage(status_msg,c_rc)

      if (c_rc .eq. 0) then
          call closefiles(filefn,20)
          return
      endif
c
c...read the user specified input data
      open(infn,file=infil,status='old',mode='read',
     $     SHARE='DENYNONE',iostat=ioerr)
```

```
      if (ioerr .ne. 0) then
            for_rc = 0
            call ioreturn(ioerr)
            call closefiles(filefn,20)
            return
      endif

      ioerr=0

      call input(infn,maxchn,title,histry,combin,import,distrb,
     $   nchn,chain,conc,ioerr,for_rc,nchnp2)

      if (ioerr .gt. 0) then
            for_rc = 0
            call closefiles(filefn,20)
            return
      endif

      if (for_rc .eq. 0) then
         status_msg(1:80)= "Error or Abort In Calculational Modules..."C
         call CMessage(status_msg,c_rc)
         call closefiles(filefn,10)
         return
      endif


      close (infn,iostat=ioerr)

      if (ioerr .ne. 0) then
            for_rc = 0
            call ioreturn(ioerr)
            call closefiles(filefn,20)
            return
      endif


      ierr = 0
c
c...if necessary, read chain and radionuclide data
c   from concentration file
      if (import) then

          status_msg(1:80) = "Reading Import Concentrations..."C
          call CMessage(status_msg,c_rc)

          if (c_rc .eq. 0) then
              call closefiles(filefn,20)
              return
          endif

         open(chainfn,file=chainfil,status='old',mode='read',
     $        SHARE='DENYNONE',iostat=ioerr)

          if (ioerr .ne. 0) then
              for_rc = 0
              call ioreturn(ioerr)
              call closefiles(filefn,20)
              return
          endif
```

```
      call rdlist(chainfn,outfn,maxnam,ierr,nnam,nucnam,ioerr)

      if (ierr .ne. 0) then
           for_rc = 0
           call closefiles(filefn,20)
           return
      endif

       if (ioerr .gt. 0) then
           for_rc = 0
           call ioreturn(ioerr)
           call closefiles(filefn,20)
           return
       endif

      close (chainfn,iostat=ioerr)

       if (ioerr .ne. 0) then
           for_rc = 0
           call ioreturn(ioerr)
           call closefiles(filefn,20)
           return
       endif

      inquire(file=confil,exist=exists,iostat=ioerr)



      if (.not. exists) then

         write(outfn,'(3a)',iostat=ioerr)
$           'Import concentration file ''',
$           confil,''' does not exist'

         ierr=1

           if (ioerr .ne. 0) then
               for_rc = 0
               call ioreturn(ioerr)
               call closefiles(filefn,20)
               return
           endif


       elseif (exists .and. ioerr .ne. 0) then

           for_rc = 0
           call ioreturn(ioerr)
           call closefiles(filefn,20)
           return

       endif

       if (ierr .ne. 0) then
           for_rc = 0
           call closefiles(filefn,20)
           return
       endif
```

```
          open(confn,file=confil,status='old',mode='read',
     $     SHARE='DENYNONE',iostat=ioerr)

           if (ioerr .ne. 0) then
               for_rc = 0
               call ioreturn(ioerr)
               call closefiles(filefn,20)
               return
           endif

          call concrd(confn,outfn,maxchn,maxnam,maxpar,
     $      nnam,nucnam,chain,ierr,nchn,nrrd,nucrd,ioerr)

           if (ioerr .ne. 0) then
               for_rc = 0
               call closefiles(filefn,20)
               return
           endif

          if (ierr .ne. 0) then
               for_rc = 0
               call closefiles(filefn,20)
               return
          endif
      endif

      status_msg(1:80) = "Reading Variable Parameters..."C
      call CMessage(status_msg,c_rc)

      if (c_rc .eq. 0) then
          call closefiles(filefn,20)
          return
      endif
c
c...read the drinking water scenario default parameters
      open(parfn,file=drfil,status='old',mode='read',
     $     SHARE='DENYNONE',iostat=ioerr)

           if (ioerr .ne. 0) then
               for_rc = 0
               call ioreturn(ioerr)
               call closefiles(filefn,20)
               return
           endif

      call dridef(parfn,ad,f1,f2,h1,h2,nlay,i,n1,n2,rho1,rho2,td,ttd,
     $    uw,vdd,tstart,tend,dt,skip,ioerr)

           if (ioerr .ne. 0) then
               for_rc = 0
               call closefiles(filefn,20)
               return
           endif


      close (parfn,iostat=ioerr)

           if (ioerr .ne. 0) then
               for_rc = 0
```

```
            call ioreturn(ioerr)
            call closefiles(filefn,20)
            return
        endif

      status_msg(1:80) = "Writing Calculational Modules Output..."C
      call CMessage(status_msg,c_rc)

      if (c_rc .eq. 0) then
          call closefiles(filefn,20)
          return
      endif

      status_msg(1:80) = "Checking for Natural Chains..."C
      call CMessage(status_msg,c_rc)

      if (c_rc .eq. 0) then
          call closefiles(filefn,20)
          return
      endif


c
c...write the user specified input data
      call driout(outfn,maxchn,adate,atime,short,long,title,
     $   histry,combin,import,distrb,confil,
     $   ad,f1,f2,h1,h2,i,n1,n2,nlay,rho1,rho2,td,ttd,uw,vdd,
     $   tstart,tend,dt,skip,
     $   nchn,chain,conc,ioerr,nchnp2)

          if (ioerr .ne. 0) then
              for_rc = 0
              call closefiles(filefn,20)
              return
          endif

c
c...check to see if any chains are the 'natural' chains
c   U_Nat or Th_Nat
      call naturl(maxchn,nchn,chain,conc)

c       status_msg(1:80) = "Checking for Equilibrium..."C
c       call CMessage(status_msg,c_rc)

c       if (c_rc .eq. 0) then
c           call closefiles(filefn,20)
c           return
c       endif

c
c...if +C (equilibrium) designation exists on chain, then strip
c   +C designation from chain and set equilibrium flag
      call equil(maxchn,nchn,chain,echain)

c
c...read chain data, cede data, kd data, and organ commited dose
c   equivalent factors

      open(chainfn,file=chainfil,status='old',blocksize=39424,
```

```
$     SHARE='DENYNONE',mode='read',iostat=ioerr)

 if (ioerr .ne. 0) then
       for_rc = 0
       call ioreturn(ioerr)
       call closefiles(filefn,20)
       return
 endif


 open(cedefn,file=cedefil,status='old',blocksize=15360,
$     SHARE='DENYNONE',mode='read',iostat=ioerr)

 if (ioerr .ne. 0) then
       for_rc = 0
       call ioreturn(ioerr)
       call closefiles(filefn,20)
       return
 endif



 open(kdfn,file=kdfil,status='old',mode='read',
$     SHARE='DENYNONE',iostat=ioerr)

 if (ioerr .ne. 0) then
       for_rc = 0
       call ioreturn(ioerr)
       call closefiles(filefn,20)
       return
 endif


 open(org1fn,file=org1,status='old',mode='read',
$     SHARE='DENYNONE',iostat=ioerr)

 if (ioerr .ne. 0) then
       for_rc = 0
       call ioreturn(ioerr)
       call closefiles(filefn,20)
       return
 endif


 open(org2fn,file=org2,status='old',mode='read',
$     SHARE='DENYNONE',iostat=ioerr)

 if (ioerr .ne. 0) then
       for_rc = 0
       call ioreturn(ioerr)
       call closefiles(filefn,20)
       return
 endif


 open(org3fn,file=org3,status='old',mode='read',
$     SHARE='DENYNONE',iostat=ioerr)
```

```
      if (ioerr .ne. 0) then
            for_rc = 0
            call ioreturn(ioerr)
            call closefiles(filefn,20)
            return
      endif



      status_msg(1:80) = "Reading Data for Chain(s)"C
      call CMessage(status_msg,c_rc)

      if (c_rc .eq. 0) then
          call closefiles(filefn,20)
          return
      endif


      do 20, ichn=1,nchn


c......read the chain data

          call rdchn(chainfn,outfn,for_rc,maxpar,maxpro,chain(ichn),
     $       nr(ichn),nuclid(1,0,ichn),hlif(1,ichn),lambda(1,ichn),
     $       ipos(1,ichn),ipar(1,1,ichn),frac(1,1,ichn),ioerr)

          if (for_rc .eq. 0) then
             exit
          endif

           if (ioerr .gt. 0) then
               for_rc = 0
               call closefiles(filefn,20)
               return
           endif

c......if import concentration file is used, check for chain
c       compatibility; if not, stop
          if (import)
     $       call contst(outfn,maxpar,nr(ichn),nrrd(ichn),
     $          nuclid(1,0,ichn),nucrd(1,ichn),ioerr)

           if (ioerr .ne. 0) then
               for_rc = 0
               call closefiles(filefn,20)
               return
           endif


c......read the cede data
          call rdcede(cedefn,outfn,for_rc,maxpar,maxpro,combin,nr(ichn),
     $       nip(1,ichn),nipdis(1,ichn),nuclid(1,0,ichn),
     $       prfrac(1,0,ichn),dfer(1,0,ichn),dfes(1,0,ichn),
     $       dfg(1,0,ichn),dfh(1,0,ichn),ioerr)

          if (for_rc .eq. 0) then
               call closefiles(filefn,20)
               exit
```

```
        endif

         if (ioerr .ne. 0) then
             for_rc = 0
             call closefiles(filefn,20)
             return
         endif


c......find the elemental names of radionuclides in chain
        call nucit(maxpar,nr(ichn),nuclid(1,0,ichn),elemnt)

c......read the partition coefficient (kd) data
        call rdkd(kdfn,outfn,for_rc,maxpar,nr(ichn),elemnt,
     $   nuclid(1,0,ichn),kd(1,ichn),ioerr)

         if (ioerr .gt. 0) then
             for_rc = 0
             call closefiles(filefn,20)
             return
         endif

        if (for_rc .eq. 0) then
             call closefiles(filefn,20)
             return
        endif

c......read the committed dose eqivalent to organ factors
        do 10, iorg=1,3
           call rddfo(orgfn+iorg,outfn,for_rc,maxpar,maxpro,maxorg,
     $       combin,nr(ichn),nuclid(1,0,ichn),
     $       dfo(7*(iorg-1)+1,1,0,ichn),ioerr)

             if (ioerr .gt. 0) then
                 for_rc = 0
                 call closefiles(filefn,20)
                 return
             endif


           if (for_rc .eq. 0) then
             call closefiles(filefn,20)
             return
           endif

   10    continue


   20 continue

      close (chainfn,iostat=ioerr)
      if (ioerr .ne. 0) then
            for_rc = 0
            call ioreturn(ioerr)
            call closefiles(filefn,20)
            return
      endif
```

```
      close (cedefn,iostat=ioerr)
      if (ioerr .ne. 0) then
            for_rc = 0
            call ioreturn(ioerr)
            call closefiles(filefn,20)
            return
      endif

      close (kdfn,iostat=ioerr)
      if (ioerr .ne. 0) then
            for_rc = 0
            call ioreturn(ioerr)
            call closefiles(filefn,20)
            return
      endif

      close (org1fn,iostat=ioerr)
      if (ioerr .ne. 0) then
            for_rc = 0
            call ioreturn(ioerr)
            call closefiles(filefn,20)
            return
      endif

      close (org2fn,iostat=ioerr)
      if (ioerr .ne. 0) then
            for_rc = 0
            call ioreturn(ioerr)
            call closefiles(filefn,20)
            return
      endif

      close (org3fn,iostat=ioerr)
      if (ioerr .ne. 0) then
            for_rc = 0
            call ioreturn(ioerr)
            call closefiles(filefn,20)
            return
      endif

      if (for_rc .eq. 0) then
          status_msg(1:80) = "Error In Calculational Modules..."C
          call CMessage(status_msg,c_rc)
          call closefiles(filefn,20)
          return
      endif


      status_msg(1:80) = "Starting Calculations..."C
      call CMessage(status_msg,c_rc)

      if (c_rc .eq. 0) then
          call closefiles(filefn,20)
          return
      endif


c
c...perform the maximum dose and history calculations
```

```
 if (histry) then
     open(hisfn,file=hisfil,status='unknown',mode='write',
$          SHARE='DENYNONE',iostat=ioerr)

     if (ioerr .ne. 0) then
         for_rc = 0
         call ioreturn(ioerr)
         call closefiles(filefn,20)
         return
     endif

 endif

 call drimax(outfn,for_rc,maxchn,maxlis,maxorg,maxpar,maxpth,
$   maxpro,norg,npath,hisfn,histry,confn,import,distrb,
$   ad,f1,f2,h1,h2,i,n1,n2,nlay,rho1,rho2,td,ttd,uw,vdd,
$   dfer,dfes,dfg,dfh,dfo,kd,
$   tstart,tend,dt,skip,
$   nchn,chain,echain,conc,nuclid,nr,nip,nipdis,hlif,lambda,
$   ipos,ipar,frac,prfrac,
$   nlist,ttede,period,printed,persv,
$   list,cwtsv,cswtsv,dossv,nucsv,pathsv,hocdsv,orgnsv,tedemx,
$   dosmx,tdos,tpdos,pathmx,tpath,tppath,nucmx,tnuc,tpnuc,ioerr)

 if (ioerr .ne. 0) then
         for_rc = 0
         call closefiles(filefn,20)
         return
 endif

 if (histry) then
    close (hisfn,iostat=ioerr)

     if (ioerr .ne. 0) then
         for_rc = 0
         call ioreturn(ioerr)
         call closefiles(filefn,20)
         return
     endif
 endif


 if (import) then
    close (confn,iostat=ioerr)

     if (ioerr .ne. 0) then
         for_rc = 0
         call ioreturn(ioerr)
         call closefiles(filefn,20)
         return
     endif
 endif


 if (for_rc .eq. 0) then
     status_msg(1:80)="Error or Abort In Calculational Modules..."C
     call CMessage(status_msg,c_rc)
     call closefiles(filefn,20)
     return
```

```
      endif

      status_msg(1:80) = "Writing Calculational Modules Results..."C
      call CMessage(status_msg,c_rc)
      if (c_rc .eq. 0) then
          call closefiles(filefn,20)
          return
      endif

c
c...write out maximum results
      call wrdri(outfn,maxorg,maxpth,maxlis,npath,nlist,
     $   ttede,ttd,period,printed,persv,
     $   list,cwtsv,cswtsv,dossv,nucsv,pathsv,tedemx,
     $   dosmx,tdos,tpdos,pathmx,tpath,tppath,nucmx,tnuc,tpnuc,
     $   hocdsv,orgnsv,ioerr)


          if (ioerr .ne. 0) then
              for_rc = 0
              call closefiles(filefn,20)
              return
          endif


       close (outfn,iostat=ioerr)

           if (ioerr .ne. 0) then
               for_rc = 0
               call ioreturn(ioerr)
               call closefiles(filefn,20)
               return
           endif

        for_rc = 1

        status_msg(1:80) = "Completed Calculational Modules..."C
        call CMessage(status_msg,c_rc)
        call closefiles(filefn,20)

        end
```

F.34 DTMEXP.FOR

```
c
c     Calculates the exponential term for the G and Ge Operators.
c     Called by goper and geoper.
c
      double precision function dtmexp(x)

      double precision count
      double precision epslon
      double precision one
      double precision two
      double precision small
      double precision x
      double precision y
      parameter (epslon=1.0e-17,one=1.0d+0,two=2.0d+0,small=4.48d-1)
      if (x .gt. small) then
         dtmexp=x-one+exp(-x)
      else
         count=two
         dtmexp=x*x/count
         y=x*x/count
    5    count=count+one
         y=-y*x/count
         dtmexp=dtmexp+y
         if (abs(y) .gt. epslon*abs(dtmexp)) go to 5
      endif
      end
```

F.35 EQUIL.FOR

```
c
c...this subroutine strips the +C designator from an equilibrium
c   chain name and sets the equilibrium flag
c
      subroutine equil(mxchn,nchn,chain,echain)

      character*8 chain
      integer ichn
      integer loc,loc1,loc2
      integer mxchn
      integer nchn
      logical echain
      dimension chain(mxchn)
      dimension echain(mxchn)
      do 10, ichn=1,nchn
         echain(ichn)=.false.
         loc1=max(index(chain(ichn),'+C'),index(chain(ichn),'+c'))
         loc2=max(index(chain(ichn),'_C'),index(chain(ichn),'_c'))
         loc=max(loc1,loc2)
         if (loc .gt. 0) then
            chain(ichn)(loc:loc+1)='  '
            echain(ichn)=.true.
         endif
   10 continue
      end
```

F.36 EXTERN.FOR

```
c
c...this subroutine calculates the external dose due to a person
c   spending time indoors, outdoors, and gardening at home.
c   see Section 5.6.1
c
      subroutine extern(nrmax,npmax,dfer,nip,nr,prfrac,
     $    sasttg,sasttr,sfi,sfo,tg,ti,ttr,ttg,tx,dexr)

      double precision dexr,dfer
      double precision factor
      double precision prfrac
      double precision sasttg,sasttr
      double precision sfi,sfo
      double precision tg,ti,ttg,ttr,twnty4,tx
      integer j
      integer k
      integer nip,nr,npmax,nrmax
      parameter (twnty4=24.0d+0)
      dimension dexr(nrmax,0:npmax),dfer(nrmax,0:npmax)
      dimension nip(nrmax)
      dimension prfrac(nrmax,0:npmax)
      dimension sasttg(nrmax),sasttr(nrmax)
c
      factor=twnty4*(tg/ttg)*sfo
c
c...calculate external dose
c......gardening dose component
      do 20, j=1,nr
         do 10, k=0,nip(j)
            dexr(j,k)=factor*sasttg(j)*prfrac(j,k)*dfer(j,k)
   10    continue
   20 continue
c......outdoor dose component
      factor=twnty4*(tx/ttr)*sfo
      do 40, j=1,nr
         do 30, k=0,nip(j)
            dexr(j,k)=dexr(j,k)+factor*sasttr(j)*prfrac(j,k)*dfer(j,k)
   30    continue
   40 continue
c......indoor dose component
      factor=twnty4*(ti/ttr)*sfi
      do 60, j=1,nr
         do 50, k=0,nip(j)
            dexr(j,k)=dexr(j,k)+factor*sasttr(j)*prfrac(j,k)*dfer(j,k)
   50    continue
   60 continue
      end
```

F.37 FRACT1.FOR

```
c
c...this subroutine sets up a fractional yield array in a vector
c   format useable by the boxset routine
c
      subroutine fract1(maxnr,nr,ipar,frac,dd)

      double precision dd
      double precision frac
      double precision zero
      integer i,indexx,ipar
      integer j
      integer maxnr
      integer nr
      parameter (zero=0.0d+0)
      dimension dd(maxnr*(maxnr-1)/2)
      dimension frac(2,maxnr)
      dimension ipar(2,maxnr)
c
      do 10, i=1,nr*(nr-1)/2
         dd(i)=zero
   10 continue
      do 30, i=2,nr
         do 20, j=1,2
            if (ipar(j,i) .ge. 1) then
               indexx=(i-1)*(i-2)/2+ipar(j,i)
               dd(indexx)=frac(j,i)
            endif
   20    continue
   30 continue
      end
```

F.38 FRCTNL.FOR

```
c
c...this subroutine sets up the fractional yield array from the
c    fractional yield data given in the chain data file
c
      subroutine frctnl(mxnr,nr,f,frac,ipar)

      double precision f,frac
      double precision zero
      integer ir,ipar
      integer jr
      integer mxnr
      integer nr
      dimension f(mxnr,mxnr)
      dimension frac(2,mxnr)
      dimension ipar(2,mxnr)
      parameter (zero=0.0d+0)
C
C...initialize array
      do 20, jr=1,mxnr
         do 10, ir=1,mxnr
            f(ir,jr)=zero
   10    continue
   20 continue
C
C...set up fractional yield array
      do 30, ir=1,nr
         if (ipar(1,ir) .gt. 0) f(ipar(1,ir),ir)=frac(1,ir)
         if (ipar(2,ir) .gt. 0) f(ipar(2,ir),ir)=frac(2,ir)
   30 continue
      end
```

F.39 GEOPER.FOR

```
c*********************************************************************
c
c  subroutine geoper; 5/93
c  C. David Updegraff, GRAM, Inc., Albuquerque, NM 87112.
c  Solves the 'ge' operator for deposition/accumulation/time-integration
c  with removal time period using equation 2.5, equation B.15, and
c  Table 2.4.
c*********************************************************************
c
      subroutine geoper(c,ge,lambdr,lambdw,d,t,nr)

      double precision c
      double precision d,dtmexp
      double precision ge
      double precision k
      double precision lambde,lambdr,lambdw
      double precision one
      double precision t
      double precision zero
      integer j
      integer maxr
      integer n,nr
      parameter (maxr=20)
      parameter (one=1.0d+0,zero=0.0d+0)
      dimension c(maxr)
      dimension d(maxr,maxr)
      dimension ge(maxr)
      dimension k(maxr,maxr)
      dimension lambdr(maxr)
c
      call kcoef(c,lambdr,d,k,nr,maxr)
c
      do 20, j=1,nr
         ge(j)=zero
            do 10, n=1,j
               lambde=lambdr(n)+lambdw
               ge(j)=ge(j)+k(j,n)*dtmexp(lambde*t)/(lambde*lambde)
   10       continue
CCC...added a check for small numbers that turn negative;
CCC...zero those numbers out
CCC            ge(j)=lambdr(j)*ge(j)
            ge(j)=max(zero,lambdr(j)*ge(j))
   20 continue
      end
```

F.40 GOPER.FOR

```
c*********************************************************************
c
c  subroutine goper;  5/93
c  C. David Updegraff, GRAM, Inc., Albuquerque, NM 87112.
c  Solves the 'g' operator for deposition/accumulation/time-integration
c  over a time period using equation 2.5, equation B.2, and Table 2.4.
c*********************************************************************
c
      subroutine goper(c,g,lambdr,d,t,nr)

      double precision c
      double precision d,dtmexp
      double precision g
      double precision k
      double precision lambdr
      double precision one
      double precision t
      double precision zero
      integer j
      integer maxr
      integer n,nr
      parameter (maxr=20)
      parameter (one=1.0d+0,zero=0.0d+0)
      dimension c(maxr)
      dimension d(maxr,maxr)
      dimension g(maxr)
      dimension k(maxr,maxr)
      dimension lambdr(maxr)
c
      call kcoef(c,lambdr,d,k,nr,maxr)
c
      do 20, j=1,nr
         g(j)=zero
            do 10, n=1,j
               g(j)=g(j)+
     $            k(j,n)*dtmexp(lambdr(n)*t)/(lambdr(n)*lambdr(n))
   10       continue
CCC...added a check for small numbers that turn negative;
CCC...zero those numbers out
CCC            g(j)=lambdr(j)*g(j)
            g(j)=max(zero,lambdr(j)*g(j))
   20 continue
      end
```

F.41 INHALE.FOR

```
c
c...this subroutine calculates the inhalation dose from gardening,
c   outdoors, and indoors at home according to Eqn. 5.70
c

      subroutine inhale(nrmax,npmax,cdg,cdi,cdo,dfh,nip,nr,pd,prfrac,
     $    rfr,sasttg,sasttr,tg,ti,ttr,ttg,tx,vg,vr,vx,dhr)

      double precision cdg,cdi,cdo
      double precision dfh,dhr
      double precision factor
      double precision pd,prfrac
      double precision rfr
      double precision sasttg,sasttr
      double precision tg,ti,ttg,ttr,twnty4,tx
      double precision vg,vr,vx
      integer j
      integer k
      integer nip,nr,npmax,nrmax
      parameter (twnty4=24.0d+0)
      dimension dfh(nrmax,0:npmax)
      dimension dhr(nrmax,0:npmax)
      dimension nip(nrmax)
      dimension prfrac(nrmax,0:npmax)
      dimension sasttg(nrmax),sasttr(nrmax)
c...calculate inhalation dose
c......gardening dose component
      factor=twnty4*vg*(tg/ttg)*cdg
      do 20, j=1,nr
         do 10, k=0,nip(j)
            dhr(j,k)=factor*sasttg(j)*prfrac(j,k)*dfh(j,k)
   10    continue
   20 continue
c......outdoor dose component
      factor=twnty4*vx*(tx/ttr)*cdo
      do 40, j=1,nr
         do 30, k=0,nip(j)
            dhr(j,k)=dhr(j,k)+factor*sasttr(j)*prfrac(j,k)*dfh(j,k)
   30    continue
   40 continue
c...indoor dose component
      factor=twnty4*vr*(ti/ttr)*(cdi+pd*rfr)
      do 60, j=1,nr
         do 50, k=0,nip(j)
            dhr(j,k)=dhr(j,k)+factor*sasttr(j)*prfrac(j,k)*dfh(j,k)
   50    continue
   60 continue
      end
```

F.42 INIT0.FOR

```
      subroutine init0(out,rc,con,concen,tt,tstart,tend,
     $        period,tcalc,tstop,ioerr)

      double precision tcalc,tend,tstart,tstop,tt
      integer ioerr
      integer out
      integer rc
      integer con
      integer endfil
      integer period
      logical concen

      period=1

      if (concen) then
          read(con,*,iostat=endfil) tstart
          if (endfil .lt. 0) then
              write(out,'(a)',iostat=ioerr)
     $         'No concentration data in import file.'

              if (ioerr .ne. 0) then
                  call ioreturn(ioerr)
                  return
              endif

              rc = 0
              return

          endif

      else

         tcalc=tstart

      endif
      tstop=tend-tt

      end
```

F.43 INITLG.FOR

```
c
c...this subroutine initializes logical variables to true

      subroutine initlg(maxdim,value,array)

      integer idim
      integer maxdim
      logical array
      logical value
      dimension array(maxdim)
c
      do 10, idim=1,maxdim
         array(idim)=value
   10 continue
      end
```

F.44 INITZC.FOR

```
c
c...this subroutine initializes double precision complex variables
c   to zero
c
      subroutine initzc(maxdim,array)

      double complex array
      double complex zero
      integer idim
      integer maxdim
      dimension array(maxdim)
      parameter (zero=(0.0d+0,0.0d+0))
c
      do 10, idim=1,maxdim
         array(idim)=zero
   10 continue
      end
```

F.45 INITZD.FOR

```
c
c...this subroutine initializes double precision variables to zero

      subroutine initzd(maxdim,array)

      double precision array
      double precision zero
      integer idim
      integer maxdim
      dimension array(maxdim)
      parameter (zero=0.0d+0)
c
      do 10, idim=1,maxdim
         array(idim)=zero
   10 continue
      end
```

F.46 INPUT.FOR

```
c
c...this subroutine reads the user specified input data
c   for the scenario
c

      subroutine input(in,mxchn,title,histry,combin,
     $   import,distrb,
     $   nchn,chain,conc,ioerr,for_rc,nchnp2)

      character*8 chain
      character*80 title
      integer in,ioerr,for_rc
      integer mxchn
      integer nchn,nchnp2
      double precision conc
      double precision zero
      logical combin
      logical import
      logical distrb
      logical histry
      dimension chain(mxchn)
      dimension conc(mxchn)
      parameter (zero=0.0d+0)

c
c...read the title of the problem
      read(in,'(a)',iostat=ioerr) title

      if (ioerr .ne. 0) then
          for_rc = 0
          call ioreturn(ioerr)
          return
      endif

c...read logical variables to determine 1. if a history plot will be made
c   2. if the implicit progeny effects are to be included with their
c   respective parents 3. an import concentrations will be used and
c   4. if initial activity will be distributed among progeny.

      read(in,*,iostat=ioerr) histry,combin,import,distrb

      if (ioerr .ne. 0) then
          for_rc = 0
          call ioreturn(ioerr)
          return
      endif

c...read chains to be solved for
      if (import) then

         do 10, nchn=1,mxchn
            conc(nchn)=zero
   10    continue

      else

         nchn=0
```

```
      nchnp2=0

15    nchn=nchn+1
      nchnp2=nchnp2+1

      read(in,*,end=25,iostat=ioerr) chain(nchn),conc(nchn)

      if (ioerr .gt. 0) then
          call ioreturn(ioerr)
          return
      endif

      if (chain(nchn) .eq. 'U_Nat') then
         nchnp2=nchnp2+2
      endif

      goto 15

25    nchn=nchn-1
      nchnp2=nchnp2-1

   endif

   end
```

F.47 IORETINT.FOR

```
c     This subroutine displays an i/o error message in the GUI along
c     with the iostat error number for INTERNAL file errors

      include "cdecl.inc"

      subroutine ioretint(iovalue)

      character*80 status_msg
      integer*4 c_rc,iovalue

      status_msg(1:80) =
     $  "An Internal File Error has occurred in the Modeling Code"C

      call CMessage(status_msg,c_rc)


      status_msg(1:80) =
     $  "                                                          "C

      write(status_msg(1:5),'(i5)') iovalue
      status_msg = status_msg(1:5)//" is the error number"C

      call CMessage(status_msg,c_rc)

      end
```

F.48 IORETURN.FOR

```
c     This subroutine displays an i/o error message in the GUI along
c     with the iostat error number for regular (non-internal file)
c     i/o errors.

      include "cdecl.inc"

      subroutine ioreturn(iovalue)

      character*80 status_msg
      integer*4 c_rc,iovalue


      status_msg(1:80) =
     $  "An Input or Output Error has occurred in the Modeling Code"C

      call CMessage(status_msg,c_rc)

CCC     Commented out the following because we do not care if the
CCC     C++ code has issued an abort. We are going to abort from the
CCC     FORTRAN side of the code.  3-25-96 MAW.
ccc      if (c_rc .eq. 0) then
ccc            return
ccc      endif

      status_msg(1:80) =
     $  "                                                            "C

      write(status_msg(1:5),'(i5)') iovalue
      status_msg = status_msg(1:5)//" is the error number"C

      call CMessage(status_msg,c_rc)

CCC     Commented out the following because we do not care if the
CCC     C++ code has issued an abort. We are going to abort from the
CCC     FORTRAN side of the code.  3-25-96 MAW.
ccc      if (c_rc .eq. 0) then
ccc            return
ccc      endif

      end
```

F.49 IRIGAT.FOR

```
c
c...this subroutine calculates dose due to ingestion of foods grown
c   with irrigated water using the second part of Eqn. 5.74 and
c   Eqn 5.76
c
      subroutine irigat(nrmax,npmax,awt,dfg,diet,nip,nr,pfw,prfrac,dwr2)

      double precision awt
      double precision dfg
      double precision diet
      double precision dwr2
      double precision pfw,prfrac
      double precision sum
      double precision zero
      integer i
      integer j
      integer k
      integer nip,npmax,nr,nrmax
      parameter (zero=0.0d+0)
      dimension awt(nrmax)
      dimension dwr2(nrmax,0:npmax)
      dimension pfw(nrmax,nrmax)
      dimension dfg(nrmax,0:npmax)
      dimension nip(nrmax)
      dimension prfrac(nrmax,0:npmax)
c
c...calculate dose due to ingestion of foods grown with
c   irrigated water
      do 30, j=1,nr
         sum=zero
         do 10, i=1,nr
            sum=sum+awt(i)*pfw(j,i)
   10    continue
         do 20, k=0,nip(j)
            dwr2(j,k)=diet*prfrac(j,k)*dfg(j,k)*sum
   20    continue
   30 continue
      end
```

F.50 ISANUC.FOR

```
c...this function checks to make sure a radionuclide
c   appears in a list
c
      logical function isanuc(nnam,nuclid,nucnam)

      character*6 nuclid,nucnam
      integer inam
      integer nnam
      dimension nucnam(nnam)
c
      isanuc=.false.
      do 10, inam=1,nnam
         if (nuclid .eq. nucnam(inam)) then
            isanuc=.true.
            goto 15
         endif
   10 continue
   15 return
      end
```

F.51 JACOBN.FOR

```
      subroutine jacobn(neq,t,y,jcol,ia,ja,pdj)
      double precision ddd
      double precision pdj
      double precision t
      double precision y
      integer i,ia,ian,ichn
      integer ja,jan,jcol
      integer maxbox,maxchn,maxneq,maxnnz,maxnuc
      integer neq
      parameter (maxbox=12,maxchn=22,maxnuc=20)
      parameter (maxneq=maxbox*maxnuc)
      parameter (maxnnz=maxbox*maxnuc*(maxnuc+3)/2)
      dimension ddd(maxnnz,maxchn)
      dimension ia(neq+1)
      dimension ian(maxneq+1,maxchn)
      dimension ja(maxnnz)
      dimension jan(maxnnz,maxchn)
      dimension pdj(neq)
      dimension y(neq)
      common /ddd/ ddd
      common /chnnum/ ichn
      common /ia/ ian
      common /ja/ jan
c
      do 10, i=ian(jcol,ichn),ian(jcol+1,ichn)-1
         pdj(jan(i,ichn))=ddd(i,ichn)
   10 continue
c
      end
```

F.52 KCOEF.FOR

```
c***********************************************************************
c
c  subroutine kcoef; 5/93
c  C. David Updegraff, GRAM, Inc., Albuquerque, NM 87112
c  Solves the coefficient equations, B.3 to B.5 for decay and ingrowth
c***********************************************************************
c
      subroutine kcoef(c,lambda,d,k,nr,maxr)

      double precision c
      double precision d
      double precision lambda
      double precision k
      double precision zero
      parameter (zero=0.0d+0)
      integer j
      integer n,nr
      integer maxr
      integer p
      dimension c(maxr)
      dimension d(maxr,maxr)
      dimension lambda(maxr)
      dimension k(maxr,maxr)
c
      k(1,1)=c(1)/lambda(1)
      do 40, j=2,nr
         do 20, n=1,j-1
            k(j,n)=zero
            do 10, p=n,j-1
               k(j,n)=k(j,n)+d(p,j)*lambda(p)*k(p,n)
   10       continue
            k(j,n)=k(j,n)/(lambda(j)-lambda(n))
   20    continue
         k(j,j)=c(j)/lambda(j)
         do 30, n=1,j-1
            k(j,j)=k(j,j)-k(j,n)
   30    continue
   40 continue
      end
```

F.53 KDCOEF.FOR

```
c***********************************************************************
c
c  subroutine kdcoef; 5/93
c  C. David Updegraff, GRAM, Inc., Albuquerque, NM 87112
c  Solves the coefficient equations, B.3 to B.5 for decay and ingrowth
c  for the drinking water scenario
c***********************************************************************
c

      subroutine kdcoef(c,lambda,d,k,nr,maxr)

      double precision c
      double precision d
      double precision lambda
      double precision k
      double precision zero
      parameter (zero=0.0d+0)
      integer j
      integer n,nr
      integer maxr
      integer p
      dimension c(maxr)
      dimension d(maxr,maxr)
      dimension lambda(maxr)
      dimension k(maxr,maxr)

      k(1,1)=c(1)
      do 40, j=2,nr
         do 20, n=1,j-1
            k(j,n)=zero
            do 10, p=n,j-1
               k(j,n)=k(j,n)+d(p,j)*k(p,n)
   10       continue
            k(j,n)=k(j,n)/(lambda(j)-lambda(n))
   20    continue
         k(j,j)=c(j)
         do 30, n=1,j-1
            k(j,j)=k(j,j)-k(j,n)
   30    continue
   40 continue

      end
```

F.54 MVMLTC.FOR

```
c
c...this subroutine multiplies an nrow by ncol complex matrix by an
c   ncol length complex vector to form a nrow length complex vector
c
c     c(nrow)=a(nrow,ncol)*b(ncol)
c
      subroutine mvmltc(nrow,ncol,a,b,c)

      double complex a
      double complex b
      double complex c
      double complex zero
      integer i
      integer j
      integer ncol,nrow
      dimension a(nrow,ncol)
      dimension b(ncol)
      dimension c(nrow)
      parameter (zero=(0.0d+0,0.0d+0))

      do 20, i=1,nrow
         c(i)=zero
         do 10, j=1,ncol
            c(i)=c(i)+a(i,j)*b(j)
   10    continue
   20 continue
      end
```

F.55 NATURL.FOR

```
c...this subroutine checks if the any chains are U_Nat or Th_Nat.
c   the U_Nat chain is broken into three chains, 234U, 235U, and 238U
c   and the U_Nat chain is eliminated. the initial activity of the
c   U_Nat chain is adjusted so that 48.9% is attributed U-234,
c   2.25% to 235U, and 48.9% to 238U.
c   the Th_Nat chain name is changed to '232Th+C', its initial activity
c   is not changed
c
      subroutine naturl(mxchn,nchn,chain,conc)

      character*8 chain
      double precision conc
      double precision factr1,factr2
      integer i,ichn
      integer mxchn
      integer nchn,nchnp2
      dimension chain(mxchn)
      dimension conc(mxchn)
      parameter (factr1=4.89D-1,factr2=2.25D-2)
c
c...if the chain is natural thorium, change the name to
c   equilibrium 232Th
      do 10, ichn=1,nchn
         if (chain(ichn) .eq. 'Th_Nat') then
            chain(ichn)='232Th+C'
            goto 15
         endif
   10 continue
   15 continue
c
c...if the chain is natural uranium
      nchnp2=nchn
      do 30, ichn=1,nchn
         if (chain(ichn) .eq. 'U_Nat') then
c......update number of chains
            nchnp2=nchnp2+2
c......move all chains and activities below U_Nat in chain list
c      down 2 locations in chain list
            do 20, i=nchn,ichn+1,-1
               chain(i+2)=chain(i)
               conc(i+2)=conc(i)
   20       continue
c......fill in the 3 locations starting with U_Nat with the
c      components of U_Nat and the respective activities
            chain(ichn)='234U    '
            chain(ichn+1)='235U    '
            chain(ichn+2)='238U    '
            conc(ichn+2)=factr1*conc(ichn)
            conc(ichn+1)=factr2*conc(ichn)
            conc(ichn)=factr1*conc(ichn)
            goto 35
         endif
   30 continue
   35 continue
c...update number of chains
      nchn=nchnp2
      end
```

F.56 NEWLAM.FOR

```
      subroutine newlam(maxpar,maxvar,112,123,lambda,nr,wd,lambdr)

      double precision 112,123,lambda,lambdr
      double precision wd
      integer ir
      integer maxpar,maxvar
      integer n3,nr
      dimension 112(maxpar),123(maxpar)
      dimension lambda(maxpar)
      dimension lambdr(maxvar)

      do 10, ir=1,nr
         n3=3*(ir-1)
         lambdr(n3+1)=112(ir)+lambda(ir)
         lambdr(n3+2)=123(ir)+lambda(ir)
         lambdr(n3+3)=wd+lambda(ir)
   10 continue
      end
```

F.57 NUCIT.FOR

```
c
c...This subroutine finds the elemental name of the isotope
c
      subroutine nucit(maxnr,nr,parnam,nuc)

      character*2 nuc
      character*6 parnam
      integer i
      integer j
      integer maxnr
      integer nr
      dimension nuc(maxnr)
      dimension parnam(maxnr)

      do 10, i=1,nr
         nuc(i)='  '
         j=0
    5    j=j+1
         if (lge(parnam(i)(j:j),'A') .and.
     $      lle(parnam(i)(j:j),'Z')) then
            nuc(i)(1:2)=parnam(i)(j:j+1)
         else
            go to 5
         endif
   10 continue
      end
```

F.58 NUCLIS.FOR

```
c
c...this subroutine takes the radionuclides in each chain and places
c   them into a unique list of radionuclides so that no radionuclide
c   repeats
c
      subroutine nuclis(mxlist,mxchn,mxnr,mxnip,
     $   nlist,nchn,nr,nip,name,list,index)

      character*6 list
      character*6 name
      integer ichn,index,iip,ir
      integer l
      integer mxchn,mxlist,mxnr,mxnip
      integer nchn,nip,nlist,nr
      dimension index(mxnr,0:mxnip,mxchn)
      dimension list(mxlist)
      dimension nip(mxnr,mxchn),nr(mxchn)
      dimension name(mxnr,0:mxnip,mxchn)
c
c...initialize a list counter
      nlist=0
c...pick a radionuclide from the chain
      do 40, ichn=1,nchn
         do 30, ir=1,nr(ichn)
            do 20, iip=0,nip(ir,ichn)
c......check the radionuclide against the list;
c      if there is a match, go to next radionuclide in the chain
               do 10, l=1,nlist
                  if (name(ir,iip,ichn) .eq. list(l)) then
                     index(ir,iip,ichn)=l
                     goto 20
                  endif
   10          continue
c......if there is no match, increment the number of radionuclides
c      in the list, add radionuclide to the list, and then store the
c      list index with the radionuclide
               nlist=nlist+1
               list(nlist)=name(ir,iip,ichn)
               index(ir,iip,ichn)=nlist
   20       continue
   30    continue
   40 continue
      end
```

F.59 NUCMAX.FOR

```
c       This subroutine checks for the maximum listed radionuclide doses;
c       saves them and their time of occurrence

      subroutine nucmax(mxlist,nlist,tcalc,period,tnuc,tpnuc,
     $    nucup,nucmx)

      double precision nucmx,nucup
      double precision tcalc,tnuc
      integer ilist
      integer mxlist
      integer nlist
      integer period,tpnuc
      dimension nucmx(mxlist),nucup(mxlist)
      dimension tnuc(mxlist)
      dimension tpnuc(mxlist)

      do 10, ilist=1,nlist
         if (nucup(ilist) .gt. nucmx(ilist)) then
            nucmx(ilist)=nucup(ilist)
            tnuc(ilist)=tcalc
            tpnuc(ilist)=period
         endif
   10 continue
      end
```

F.60 NUCSUM.FOR

```
c
c...this subroutine calculates dose by radionuclide for a chain
c
      subroutine nucsum(mxnr,mxnip,mxpath,nr,nip,npath,dose,nuc)

      double precision dose
      double precision nuc
      double precision zero
      integer iip,ipath,ir
      integer mxnip,mxnr,mxpath
      integer nip,npath,nr
      dimension dose(mxnr,0:mxnip,mxpath)
      dimension nip(mxnr)
      dimension nuc(mxnr,0:mxnip)
      parameter (zero=0.0d+0)
c
c...initialize radionuclide dose
      do 20, ir=1,nr
         do 10, iip=0,nip(ir)
            nuc(ir,iip)=zero
   10    continue
   20 continue
c
c...calculate radionuclide dose
      do 50, ipath=1,npath
         do 40, ir=1,nr
            do 30, iip=0,nip(ir)
               nuc(ir,iip)=nuc(ir,iip)+dose(ir,iip,ipath)
   30       continue
   40    continue
   50 continue
      end
```

F.61 OCCDOS.FOR

```
c
c...this subroutine calculates the doses for each radionuclide
c   in a chain for each pathway for the building renovation scenario
c
      subroutine occdos(mxnr,mxnip,nr,nip,
     $   t,dfes,dfh,dfg,to,tto,vo,rfo,go,
     $   ci,lambda,f,prfrac,
     $   c,sbar,dexo,dho,dgo)

      double precision c,ci
      double precision dexo,dfes,dfg,dfh,dgo,dho
      double precision f,frt505
      double precision go
      double precision lambda
      double precision prfrac
      double precision rfo
      double precision sbar
      double precision t,to,tto,twnty4
      double precision vo
      integer iip,ir
      integer mxnip,mxnr
      integer nip,nr
      dimension c(mxnr,0:mxnip),ci(mxnr)
      dimension dexo(mxnr,0:mxnip),dfes(mxnr,0:mxnip)
      dimension dfg(mxnr,0:mxnip),dfh(mxnr,0:mxnip)
      dimension dgo(mxnr,0:mxnip),dho(mxnr,0:mxnip)
      dimension nip(mxnr)
      dimension f(mxnr,mxnr)
      dimension lambda(nr)
      dimension prfrac(mxnr,0:mxnip)
      dimension sbar(mxnr,0:mxnip)
      parameter (frt505=4.505d+1)
      parameter (twnty4=2.4d+1)
c
c...calculate the activity of each radionuclide at the beginning
c   of the time of interest
      call aoper(ci,c,lambda,f,t,nr)
c
c...integrate the activity over the time in building
      call soper(c,sbar,lambda,f,tto,nr)

c
c...calculate average activity for each explicit radionuclide
c
      do 10, ir=1,nr
         sbar(ir,0)=sbar(ir,0)/tto
   10 continue

c
c...calculate the dose for each radionuclide and pathway in the chain
      do 30, ir=1,nr
         do 20, iip=0,nip(ir)
            c(ir,iip)=c(ir,0)*prfrac(ir,iip)
            sbar(ir,iip)=sbar(ir,0)*prfrac(ir,iip)

c......external dose from eq. (3.16)
            dexo(ir,iip)=twnty4*to*dfes(ir,iip)*sbar(ir,iip)
```

```
c......inhalation dose from eq. (3.18)
            dho(ir,iip)=
     $         frt505*twnty4*to*rfo*vo*dfh(ir,iip)*sbar(ir,iip)

c......ingestion dose from eq. (3.20)
            dgo(ir,iip)=
     $         frt505*twnty4*to*go*dfg(ir,iip)*sbar(ir,iip)

   20    continue
   30 continue
      end
```

F.62 OCCMAIN.FOR

```
c*********************************************************************
c
c  PROGRAM OCCUPA; v2.0; 9/94
c  C. D. Updegraff, GRAM, Inc. Albuquerque, NM 87112.
c  Solves the building occupancy equations from Kennedy, Jr., W.E.
c  and D.L. Strenge, 1992, "Residual Radioactive Contamination From
c  Decommissioning: Technical Basis for Translating Contamination
c  Levels to Annual Total Effective Dose Equivalent," NUREG/CR-5512,
c  PNL-7994, Vol. 1, Pacific Northwest Laboratory, Richland, WA 99352.
c  Benchmarked against RADDECAY Version 3.01 11/92 for the U-238 chain
c
c*********************************************************************
c

      include "cdecl.inc"

      subroutine occupa(for_rc)

      character*6 list
      character*6 nuclid
      character*6 short
      character*10 adate
      character*8 chain
      character*11 atime
      character*27 long
      character*80 filfil
      character*80 chainfil,cedefil,occfil
      character*80 hisfil,confil
      character*80 infil
      character*80 outfil
      character*80 title
      character*80 status_msg
      double precision c,con,conc,cup
      double precision dfer,dfes,dfg,dfh,dos,dosmx,dossv,dosup
      double precision d,dt
      double precision frac
      double precision go
      double precision hlif
      double precision lambda
      double precision nuc,nucmx,nucsv,nucup
      double precision path,pathmx,pathsv,pathup
      double precision prfrac
      double precision rfo
      double precision sbar,sbarml,sbarsv,sbarup
      double precision to,tto,tdos,tedemx,tend
      double precision tnuc,tpath,tstart,ttede
      double precision vo
      double precision zero
      integer conunt
      integer hisfn,infn,outfn,filefn,chainfn,cedefn
      integer ichn,indx,ioerr,ipar,ipos
      integer mxchn,mxlist,mxpar,mxpath,mxpro
      integer nchn,nchnp2,nip,nipdis,nlist,npath,nr
      integer period,printed,persv,tppath,tpdos,tpnuc
      integer skip
      integer*4 for_rc
      integer*4 c_rc
```

```
CCC...added to set the date and time of the run
      integer*2 month,day,year
      integer*2 hour,minute,second,hsecond
      logical distrb
      logical combin
      logical echain
      logical histry
      logical import
      parameter (mxchn=22,mxpar=20,mxpath=3,mxpro=6,npath=3)
      parameter (mxlist=mxchn*mxpar)
      parameter (zero=0.0d+0)
      dimension c(mxpar,0:mxpro)
      dimension chain(mxchn)
      dimension con(mxpar,mxchn)
      dimension conc(mxchn)
      dimension cup(mxlist)
      dimension d(mxpar,mxpar,mxchn)
      dimension dfer(mxpar,0:mxpro,mxchn),dfes(mxpar,0:mxpro,mxchn)
      dimension dfg(mxpar,0:mxpro,mxchn),dfh(mxpar,0:mxpro,mxchn)
      dimension dos(mxpar,0:mxpro,mxpath)
      dimension dosmx(mxlist,mxpath)
      dimension dossv(mxlist,mxpath)
      dimension dosup(mxlist,mxpath)
      dimension echain(mxchn)
      dimension frac(2,mxpar,mxchn)
      dimension hlif(mxpar,mxchn)
      dimension indx(mxpar,0:mxpro,mxchn)
      dimension ipar(2,mxpar,mxchn)
      dimension ipos(mxpar,mxchn)
      dimension lambda(mxpar,mxchn)
      dimension list(mxlist)
      dimension nip(mxpar,mxchn),nipdis(mxpar,mxchn)
      dimension nr(mxchn)
      dimension nuc(mxpar,0:mxpro)
      dimension nuclid(mxpar,0:mxpro,mxchn)
      dimension nucmx(mxlist),nucsv(mxlist),nucup(mxlist)
      dimension path(mxpath),pathmx(mxpath)
      dimension pathsv(mxpath),pathup(mxpath)
      dimension prfrac(mxpar,0:mxpro,mxchn)
      dimension sbar(mxpar,0:mxpro)
      dimension sbarml(mxlist),sbarsv(mxlist),sbarup(mxlist)
      dimension tdos(mxlist,mxpath)
      dimension tpdos(mxlist,mxpath)
      dimension tnuc(mxlist)
      dimension tpnuc(mxlist)
      dimension tpath(mxpath)
      dimension tppath(mxpath)

      data long /'Building Occupancy Scenario'/
      data short /'OCCUPA'/

      filefn  = 50
      for_rc = 1
      c_rc = 1

      status_msg(1:80) = "Starting Calculational Modules..."C

      call CMessage(status_msg,c_rc)
```

```
      if (c_rc .eq. 0) then
            call closefiles(filefn,15)
            return
      endif


c
CCC       call date(adate)
CCC       call time(atime)
CCC       call timer(tick1)
CCC...converted to Microsoft Powerstation FORTRAN functions
      call GetDat(year,month,day)
      write(adate,'(I2.2,A,I2.2,A,I4.4)',iostat=ioerr)
     $   month, '/', day, '/', year

      if (ioerr .ne. 0) then
            for_rc = 0
            call ioretint(ioerr)
            call closefiles(filefn,15)
            return
      endif

      call GetTim(hour,minute,second,hsecond)
      write(atime,'(I2.2,A,I2.2,A,I2.2)',iostat=ioerr)
     $   hour, ':', minute, ':', second

      if (ioerr .ne. 0) then
            for_rc = 0
            call ioretint(ioerr)
            call closefiles(filefn,15)
            return
      endif

      conunt=0
      confil=' '

      status_msg(1:80) = "Reading Input Filenames..."C
      call CMessage(status_msg,c_rc)

      if (c_rc .eq. 0) then
            call closefiles(filefn,15)
            return
      endif

      filfil = "data\occfnam"
      chainfil = "data\chain.dat"
      cedefil = "data\cede.dat"
      occfil = "data\occ.dat"

c     assign the logical file units
      filefn  = 50
      infn    = filefn + 1
      outfn   = filefn + 2
      hisfn   = filefn + 3
      chainfn = filefn + 4
      cedefn  = filefn + 5
c
c...read the input file number and name
      OPEN(filefn,FILE=filfil,STATUS='OLD',mode='read',
```

```
     $      SHARE='DENYNONE',iostat=ioerr)

      if (ioerr .ne. 0) then
            for_rc = 0
            call ioreturn(ioerr)
            call closefiles(filefn,15)
            return
      endif

      read(filefn,*) infil,outfil,hisfil


      CLOSE(filefn,iostat=ioerr)

      if (ioerr .ne. 0) then
            for_rc = 0
            call ioreturn(ioerr)
            call closefiles(filefn,15)
            return
      endif

c...open output file
      open(outfn,file=outfil,status='unknown',mode='write',
     $      SHARE='DENYNONE',iostat=ioerr)

      if (ioerr .ne. 0)then
            call ioreturn(ioerr)
            call closefiles(filefn,15)
            return
      endif

      status_msg(1:80) = "Reading User Input..."C
      call CMessage(status_msg,c_rc)

      if (c_rc .eq. 0) then
          call closefiles(filefn,15)
          return
      endif

c
c...read the user specified input data
      open(filefn,file=infil,status='old',mode='read',
     $      SHARE='DENYNONE',iostat=ioerr)

      if (ioerr .ne. 0) then
            for_rc = 0
            call ioreturn(ioerr)
            call closefiles(filefn,15)
            return
      endif

      ioerr = 0
      call input(filefn,mxchn,title,histry,combin,import,distrb,
     $   nchn,chain,conc,ioerr,for_rc,nchnp2)

      if (ioerr .gt. 0) then
            for_rc = 0
            call closefiles(filefn,15)
            return
```

```
      endif

      if (for_rc .eq. 0) then
         status_msg(1:80)= "Error or Abort In Calculational Modules..."C
         call CMessage(status_msg,c_rc)
         call closefiles(filefn,15)
         return
      endif

      close (filefn,iostat=ioerr)

      if (ioerr .ne. 0) then
            for_rc = 0
            call ioreturn(ioerr)
            call closefiles(filefn,15)
            return
      endif
c
c...read the occupancy scenario default parameters
      open(filefn,file=occfil,status='old',mode='read',
     $     SHARE='DENYNONE',iostat=ioerr)

      if (ioerr .ne. 0)then
            call ioreturn(ioerr)
            call closefiles(filefn,15)
            return
      endif

      ioerr = 0
      call blddef(filefn,to,tto,vo,rfo,go,tstart,tend,dt,skip,ioerr)

      if (ioerr .ne. 0) then
            for_rc = 0
            call closefiles(filefn,15)
            return
      endif

      close (filefn,iostat=ioerr)

      if (ioerr .ne. 0) then
            for_rc = 0
            call ioreturn(ioerr)
            call closefiles(filefn,15)
            return
      endif

c...write the user specified input data
      ioerr = 0

      call occout(outfn,mxchn,adate,atime,short,long,title,
     $   histry,combin,import,distrb,confil,
     $   to,tto,vo,rfo,go,
     $   tstart,tend,dt,skip,
     $   nchn,chain,conc,ioerr,nchnp2)

      if (ioerr .ne. 0) then
            for_rc = 0
            call closefiles(filefn,15)
            return
```

```
      endif

      status_msg(1:80) = "Reading Data for Chain(s)..."C
      call CMessage(status_msg,c_rc)

      if (c_rc .eq. 0) then
          call closefiles(filefn,15)
          return
      endif
c
c...check to see if any chains are the 'natural' chains
c   U_Nat or Th_Nat
      call naturl(mxchn,nchn,chain,conc)
c
c...if +C (equilibrium) designation exists on chain, then strip
c   +C designation from chain and set equilibrium flag
      call equil(mxchn,nchn,chain,echain)


c
c...read chain data and cede data
c     Blocksize was changed because the chain.asc file became
c     larger. actual file size is 38,903 bytes and the next
c     increment in 512k byte blocks is 39424. 03-25-96 MAW

      open(chainfn,file=chainfil,status='old',blocksize=39424,
     $    SHARE='DENYNONE',mode='read',iostat=ioerr)

      if (ioerr .ne. 0) then
            for_rc = 0
            call ioreturn(ioerr)
            call closefiles(filefn,15)
            return
      endif

      open(cedefn,file=cedefil,status='old',blocksize=15360,
     $    SHARE='DENYNONE',mode='read',iostat=ioerr)

      if (ioerr .ne. 0) then
            for_rc = 0
            call ioreturn(ioerr)
            call closefiles(filefn,15)
            return
      endif

      for_rc = 1

      do 10, ichn=1,nchn
c......read the chain data
         call rdchn(chainfn,outfn,for_rc,mxpar,mxpro,chain(ichn),
     $      nr(ichn), nuclid(1,0,ichn),hlif(1,ichn),lambda(1,ichn),
     $      ipos(1,ichn),ipar(1,1,ichn),frac(1,1,ichn),ioerr)

          if (ioerr .gt. 0) then
              for_rc = 0
              call closefiles(filefn,15)
              return
          endif
```

```
         if (for_rc .eq. 0) then
              exit
         endif

c......read the cede data


         call rdcede(cedefn,outfn,for_rc,mxpar,mxpro,combin,nr(ichn),
     $      nip(1,ichn),nipdis(1,ichn),nuclid(1,0,ichn),
     $      prfrac(1,0,ichn),dfer(1,0,ichn),dfes(1,0,ichn),
     $      dfg(1,0,ichn),dfh(1,0,ichn),ioerr)

          if (ioerr .gt. 0) then
                for_rc = 0
                call closefiles(filefn,15)
                return
          endif

         if (for_rc .eq. 0) then
            exit
         endif

   10 continue

      close(chainfn,iostat=ioerr)

      if (ioerr .ne. 0) then
            for_rc = 0
            call ioreturn(ioerr)
            call closefiles(filefn,15)
            return
      endif

      close(cedefn,iostat=ioerr)

      if (ioerr .ne. 0) then
            for_rc = 0
            call ioreturn(ioerr)
            call closefiles(filefn,15)
            return
      endif

      if (for_rc .eq. 0) then
          status_msg(1:80) = "Error In Calculational Modules..."C
          call CMessage(status_msg,c_rc)
          call closefiles(filefn,15)
          return
      endif

      status_msg(1:80) = "Starting Calculations..."C
      call CMessage(status_msg,c_rc)

      if (c_rc .eq. 0) then
          call closefiles(filefn,15)
          return
      endif

c
c...perform the maximum dose and history calculations
```

```
 if (histry) then

 open(hisfn,file=hisfil,status='unknown',mode='write',
$     SHARE='DENYNONE',iostat=ioerr)

     if (ioerr .ne. 0) then
         for_rc = 0
         call ioreturn(ioerr)
         call closefiles(filefn,15)
         return
     endif

 endif

 call occmax(outfn,for_rc,mxchn,mxlist,mxpar,mxpath,mxpro,npath,
$   hisfn,histry,conunt,import,distrb,
$   to,tto,vo,rfo,go,
$   tstart,tend,dt,skip,
$   nchn,chain,conc,
$   nuclid,nr,nip,nipdis,hlif,lambda,
$   ipos,ipar,frac,prfrac,
$   dfer,dfes,dfg,dfh,
$   nlist,
$   ttede,period,printed,persv,
$   list,sbarsv,dossv,nucsv,pathsv,tedemx,
$   dosmx,tdos,tpdos,pathmx,tpath,tppath,nucmx,tnuc,tpnuc,
$   c,cup,con,dos,dosup,echain,d,indx,nuc,nucup,
$   path,pathup,sbar,sbarml,sbarup,ioerr)

 if (ioerr .ne. 0) then
       for_rc = 0
       call closefiles(filefn,15)
       return
 endif

 close (hisfn,iostat=ioerr)

 if (ioerr .ne. 0) then
       for_rc = 0
       call ioreturn(ioerr)
       call closefiles(filefn,15)
       return
 endif

 if (for_rc .eq. 0) then
     status_msg(1:80)="Error or Abort In Calculational Modules..."C
     call CMessage(status_msg,c_rc)
     call closefiles(filefn,15)
     return
 endif


 status_msg(1:80) = "Writing Calculational Modules Results..."C
 call CMessage(status_msg,c_rc)

 if (c_rc .eq. 0) then
     call closefiles(filefn,15)
     return
```

```
      endif

c
c...write outfn maximum results
      call wrout(outfn,mxpath,mxlist,npath,nlist,short,
     $   ttede,tto,period,printed,persv,
     $   list,sbarsv,dossv,nucsv,pathsv,tedemx,
     $   dosmx,tdos,tpdos,pathmx,tpath,tppath,nucmx,tnuc,tpnuc,ioerr)

      if (ioerr .ne. 0) then
            for_rc = 0
            call closefiles(filefn,15)
            return
      endif



      close (outfn,iostat=ioerr)

          if (ioerr .ne. 0) then
              for_rc = 0
              call ioreturn(ioerr)
              call closefiles(filefn,20)
              return
          endif

      for_rc = 1

      status_msg(1:80) = "Completed Calculational Modules..."C
      call CMessage(status_msg,c_rc)
      call closefiles(filefn,15)

      end
```

F.63 OCCMAX.FOR

```
c
c  This subroutine performs the maximum dose calculations
c  and updates the history file for selected time steps
c


      include "cdecl.inc"

      subroutine occmax(out,rc,mxchn,mxlist,mxpar,mxpath,mxpro,npath,
     $   his,histry,conunt,import,distrb,
     $   to,tto,vo,rfo,go,
     $   tstart,tend,dt,skip,
     $   nchn,chain,conc,
     $   nuclid,nr,nip,nipdis,hlif,lambda,
     $   ipos,ipar,frac,prfrac,
     $   dfer,dfes,dfg,dfh,
     $   nlist,
     $   ttede,period,printed,persv,
     $   list,sbarsv,dossv,nucsv,pathsv,tedemx,
     $   dosmx,tdos,tpdos,pathmx,tpath,tppath,nucmx,tnuc,tpnuc,
     $   c,cup,con,dos,dosup,echain,f,indx,nuc,nucup,
     $   path,pathup,sbar,sbarml,sbarup,ioerr)
      character*6 list
      character*6 nuclid
      character*8 chain
      character*80 status_msg
      double precision c,con,conc,cup
      double precision dfer,dfes,dfg,dfh
      double precision dos,dosmx,dossv,dosup
      double precision dt
      double precision f,frac
      double precision go
      double precision hlif
      double precision lambda
      double precision nuc,nucmx,nucsv,nucup
      double precision path,pathmx,pathup,pathsv,prfrac
      double precision rfo
      double precision sbar,sbarml,sbarsv,sbarup
      double precision to,tto,tcalc,tdos,tede,tedemx,tedeup,tend
      double precision tnuc,tpath,tstart,tstop,ttede
      double precision vo
      integer conunt
      integer his
      integer ichn,indx,ioerr,ipar,ipos
      integer mxchn,mxlist,mxpar,mxpath,mxpro
      integer nchn,nip,nipdis,nlist,npath,nr
      integer out,rc
      integer period,printed,tppath,persv,tpdos,tpnuc
      integer skip
      integer tsteps,achk
      logical distrb
      logical echain
      logical histry
      logical import
      logical report
      logical dcreas,iszero
```

```
      logical done
      dimension c(mxpar,0:mxpro)
      dimension chain(mxchn)
      dimension con(mxpar,mxchn)
      dimension conc(mxchn)
      dimension cup(mxlist)
      dimension dfer(mxpar,0:mxpro,mxchn),dfes(mxpar,0:mxpro,mxchn)
      dimension dfg(mxpar,0:mxpro,mxchn),dfh(mxpar,0:mxpro,mxchn)
      dimension dos(mxpar,0:mxpro,mxpath)
      dimension dosmx(mxlist,mxpath)
      dimension dossv(mxlist,mxpath),dosup(mxlist,mxpath)
      dimension echain(mxchn)
      dimension f(mxpar,mxpar,mxchn)
      dimension frac(2,mxpar,mxchn)
      dimension hlif(mxpar,mxchn)
      dimension indx(mxpar,0:mxpro,mxchn)
      dimension ipar(2,mxpar,mxchn)
      dimension ipos(mxpar,mxchn)
      dimension lambda(mxpar,mxchn)
      dimension list(mxlist)
      dimension nip(mxpar,mxchn),nipdis(mxpar,mxchn)
      dimension nr(mxchn)
      dimension nuc(mxpar,0:mxpro)
      dimension nuclid(mxpar,0:mxpro,mxchn)
      dimension nucmx(mxlist),nucsv(mxlist),nucup(mxlist)
      dimension path(mxpath),pathmx(mxpath)
      dimension pathsv(mxpath),pathup(mxpath)
      dimension prfrac(mxpar,0:mxpro,mxchn)
      dimension sbar(mxpar,0:mxpro)
      dimension sbarm1(mxlist),sbarsv(mxlist),sbarup(mxlist)
      dimension tdos(mxlist,mxpath)
      dimension tnuc(mxlist)
      dimension tpath(mxpath)
      dimension tppath(mxpath)
      dimension tpdos(mxlist,mxpath)
      dimension tpnuc(mxlist)

      printed = 0

      tsteps = ((tend - tstart)/dt)

      if ((dt .eq. 0) .OR. ((tend - tstart) .le. 100)) then
          achk = 10
      else
          achk = tsteps/10
      endif

      if (achk .eq. 0) achk = 10
c
c...perform preliminary calculations
      do 10, ichn=1,nchn

c......set up the initial activities for each radionuclide in the chain
         call concen(mxpar,mxpro,nr(ichn),echain(ichn),distrb,
     $       lambda(1,ichn),ipar(1,1,ichn),frac(1,1,ichn),
     $       prfrac(1,0,ichn),nipdis(1,ichn),conc(ichn),con(1,ichn))

c......write chain, cede and activity data to output file
         call wrchn(out,mxpar,mxpro,nr(ichn),nip(1,ichn),ichn,
```

```
     $       chain(ichn),echain(ichn),nuclid(1,0,ichn),ipos(1,ichn),
     $       ipar(1,1,ichn),frac(1,1,ichn),hlif(1,ichn),prfrac(1,0,ichn),
     $       con(1,ichn),dfg(1,0,ichn),dfh(1,0,ichn),dfes(1,0,ichn),
     $       dfer(1,0,ichn),'OCCUPA',ioerr)

          if (ioerr .ne. 0) then
            return
          endif

c......convert dfer, dfes, dfg, and dfh from Bq-Sv system to
c   mrem-H-pCi system
          call convrt(mxpar,mxpro,nr(ichn),nip(1,ichn),
     $       dfg(1,0,ichn),dfh(1,0,ichn),
     $       dfes(1,0,ichn),dfer(1,0,ichn))

c......set up the fractional yield array
          call frctnl(mxpar,nr(ichn),f(1,1,ichn),
     $       frac(1,1,ichn),ipar(1,1,ichn))

   10 continue


c
c...combine radionuclides from all chains into a unique list of
c   radionuclides and tie each radionuclide from each chain into
c   an index number in the list
      call nuclis(mxlist,mxchn,mxpar,mxpro,
     $   nlist,nchn,nr,nip,nuclid,list,indx)

      status_msg = "Calculating up to                "

      write(status_msg(19:23),'(i4)',iostat=ioerr) tsteps

      if (ioerr .ne. 0) then
          call ioretint(ioerr)
          return
      endif

      status_msg = status_msg(1:24)//' timesteps...'C
      call CMessage(status_msg,rc)
c
c...calculate doses

c......initialize some time parameters
      call init0(out,rc,conunt,import,tto,tstart,
     $   tend,period,tcalc,tstop,ioerr)

          if (ioerr .ne. 0) then
            return
          endif

          if (rc .eq. 0) then
              return
          endif

c......initialize maximum values of comparison variables
      call initzd(mxlist*mxpath,dosmx)
      call initzd(mxlist,nucmx)
      call initzd(mxpath,pathmx)
```

```
      call initzd(1,tedemx)
      call initzd(mxlist,sbarml)
      call initlg(1,.false.,dcreas)

   15 continue

c......determine if current calculation time is a report time
      call rpttim(tcalc,tstop,period,skip,histry,report)

c......calculate dose for each radionuclide in each chain
c       and for each pathway
c......initialize update variables for the current time step
      call initzd(mxlist*mxpath,dosup)
      call initzd(mxlist,nucup)
      call initzd(mxpath,pathup)
      call initzd(1,tedeup)
      call initzd(mxlist,cup)
      call initzd(mxlist,sbarup)
      do 20, ichn=1,nchn

c.........calculate dose in chain for each radionuclide
c          for each pathway
         call occdos(mxpar,mxpro,nr(ichn),nip(1,ichn),
     $       tcalc,dfes(1,0,ichn),dfh(1,0,ichn),dfg(1,0,ichn),
     $       to,tto,vo,rfo,go,
     $       con(1,ichn),lambda(1,ichn),f(1,1,ichn),prfrac(1,0,ichn),
     $       c,sbar,dos(1,0,1),dos(1,0,2),dos(1,0,3))

c
c.........calculate pathway dose for chain
         call pthsum(mxpar,mxpro,mxpath,nr(ichn),nip(1,ichn),npath,
     $        dos,path)
c
c.........calculate radionuclide dose for chain
         call nucsum(mxpar,mxpro,mxpath,nr(ichn),nip(1,ichn),npath,
     $        dos,nuc)
c
c.........calculate tede for chain
         call tedsum(mxpar,mxpro,nr(ichn),nip(1,ichn),nuc,tede)
c
c.........update listed radionuclide activity
         call upsbar(mxpar,mxpro,mxlist,nr(ichn),nip(1,ichn),
     $       indx(1,0,ichn),c,cup)
c
c.........update listed radionuclide average activity
         call upsbar(mxpar,mxpro,mxlist,nr(ichn),nip(1,ichn),
     $       indx(1,0,ichn),sbar,sbarup)
c
c.........update listed radionuclide dose for each pathway for all chains
         call updose(mxpar,mxpro,mxpath,mxlist,
     $       nr(ichn),nip(1,ichn),npath,
     $       indx(1,0,ichn),dos,dosup)
c
c.........update pathway dose for all chains
         call uppath(mxpath,npath,path,pathup)
c
c.........update listed radionuclide dose for all chains
         call upnuc(mxpar,mxpro,mxlist,nr(ichn),nip(1,ichn),
     $       indx(1,0,ichn),nuc,nucup)
```

```
c
c.........update total dose for all chains
         call uptede(tede,tedeup)
   20 continue

c
c...if this is a report time write results to history file
      if (report) then
          call wrhist(his,mxlist,mxpath,nlist,npath,
     $        tcalc,tcalc+tto,list,cup,sbarup,dosup,pathup,nucup,
     $        tedeup, ioerr)

          if (ioerr .ne. 0) then
            return
          endif

          printed = printed + 1

      endif

c
c......if integrated activities are increasing, then test for
c      maximum values of total dose (tede) and find its components,
c      pathway dose, listed radionuclide dose, and listed
c      radionuclide dose by pathway
      if (.not. dcreas) then

c......check if total tede is maximum so far; if so, save it,
c      its components, and its time of occurrence
         call tedmax(mxlist,mxpath,nlist,npath,tcalc,period,persv,
     $       ttede,tedeup,tedemx,pathsv,pathup,nucsv,nucup,dossv,dosup,
     $       sbarsv,sbarup)

c......check for maximum pathway doses; if so, save them and their
c      time of occurrence
         call pthmax(mxpath,npath,tcalc,period,tpath,tppath,
     $       pathup,pathmx)

c......check for maximum listed radionuclide doses; if so, save
c      them and their time of occurrence
         call nucmax(mxlist,nlist,tcalc,period,tnuc,tpnuc,
     $       nucup,nucmx)

c......check for maximum listed radionuclide doses per pathway; if so,
c      save them and their time of occurence
         call dosmax(mxpath,mxlist,npath,nlist,tcalc,period,
     $       dosup,tdos,tpdos,dosmx)

      endif

c......determine if the integrated activities are decreasing. if they
c      are all decreasing, set a test value to true and check history
c      value to determine if maximum results should be written out at
c      this time
      iszero = .true.
      call acttst(mxlist,nlist,sbarup,sbarm1,dcreas,iszero)

      if (dcreas .and. .not. histry) then
          status_msg = "All doses are decreasing..."C
```

```
            call CMessage(status_msg,rc)
            goto 25
        endif

        if (dcreas .and. iszero) then
            status_msg = "All doses are decreasing and under minimum..."C
            call CMessage(status_msg,rc)
            goto 25
        endif

        if (tcalc .ge. tstop) goto 25
c......update time
        call uptime(conunt,import,done,period,tcalc,dt,tstop,ioerr)

        if (ioerr .ne. 0) then
            return
        endif

        if (mod(period, achk) .eq. 1) then

            write(status_msg(1:10),'(i10)',iostat=ioerr) period
            if (ioerr .ne. 0) then
                call ioretint(ioerr)
                return
            endif

            status_msg = status_msg(1:10)//'  timestep(s) out of 'C

            write(status_msg(32:36),'(i4)',iostat=ioerr) tsteps
            if (ioerr .ne. 0) then
                call ioretint(ioerr)
                return
            endif

            status_msg = status_msg(1:36)//' calculated...'C
            call CMessage(status_msg,rc)
            if (rc .eq. 0) then
                return
            endif
        else
            call CProcess(rc)
            if (rc .eq. 0) then
                return
            endif
        endif

        goto 15

   25 continue

        write(status_msg(1:10),'(i10)',iostat=ioerr) period

        if (ioerr .ne. 0) then
            call ioretint(ioerr)
            return
        endif

        status_msg = status_msg(1:10)//
       $     ' timesteps were actually calculated'C
```

```
call CMessage(status_msg,rc)

end
```

F.64 OCCOUT.FOR

```
c
c...this subroutine writes out the user specified input data
c   for the building occupancy scenario to the FORTRAN output file
c

      subroutine occout(out,mxchn,adate,atime,short,long,title,
     $   histry,combin,import,distrb,confil,
     $   to,tto,vo,rfo,go,
     $   tstart,tend,dt,skip,
     $   nchn,chain,conc,ioerr,nchnp2)

      character*6 short
      character*10 adate
      character*8 chain
      character*11 atime
      character*27 long
      character*80 confil
      character*80 title
      integer ichn,ioerr
      integer loc,loc1,loc2
      integer mxchn
      integer nchn,nchnp2
      integer out
      integer skip
      logical combin
      logical distrb
      logical equil
      logical histry
      logical import
      double precision conc
      double precision dt
      double precision go
      double precision rfo
      double precision to,tto,tend,tstart
      double precision vo
      dimension chain(mxchn)
      dimension conc(mxchn)
c
      write(out,'(3a)',iostat=ioerr) 'Output From Program ''',short,''''
      if (ioerr .ne. 0) then
         call ioreturn(ioerr)
         return
      endif

      write(out,'(a)',iostat=ioerr) long
      if (ioerr .ne. 0) then
         call ioreturn(ioerr)
         return
      endif

      write(out,'(2a)',iostat=ioerr) 'Run Date: ',adate
      if (ioerr .ne. 0) then
         call ioreturn(ioerr)
         return
      endif

      write(out,'(2a)',iostat=ioerr) 'Run Time: ',atime
```

```
      if (ioerr .ne. 0) then
         call ioreturn(ioerr)
         return
      endif

      write(out,'(a)',iostat=ioerr) ' '
      if (ioerr .ne. 0) then
         call ioreturn(ioerr)
         return
      endif

      write(out,'(a)',iostat=ioerr) 'INPUT DATA:'
      if (ioerr .ne. 0) then
         call ioreturn(ioerr)
         return
      endif

      write(out,'(2a)',iostat=ioerr) 'Title: ',title
      if (ioerr .ne. 0) then
         call ioreturn(ioerr)
         return
      endif

      write(out,'(a)',iostat=ioerr) 'Notes: '
      if (ioerr .ne. 0) then
         call ioreturn(ioerr)
         return
      endif

      if (histry) then
         write(out,'(a)',iostat=ioerr) 'History file will be generated.'
          if (ioerr .ne. 0) then
           call ioreturn(ioerr)
           return
          endif

      else
         write(out,'(a)',iostat=ioerr)
     $      'History file will not be generated.'
          if (ioerr .ne. 0) then
              call ioreturn(ioerr)
              return
          endif

      endif

      if (combin) then
         write(out,'(2a)',iostat=ioerr) 'Implicit progeny doses will',
     $      ' be included with explicit parent.'
          if (ioerr .ne. 0) then
              call ioreturn(ioerr)
              return
          endif

      else
         write(out,'(2a)',iostat=ioerr)
     $      'Implicit progeny doses will not',
     $      ' be included with explicit parent.'
          if (ioerr .ne. 0) then
```

```
            call ioreturn(ioerr)
            return
         endif

      endif

      if (import) then
         write(out,'(4a)',iostat=ioerr)
     $      'Concentration data will be read ',
     $      'from file: ',confil,'.'
         if (ioerr .ne. 0) then
            call ioreturn(ioerr)
            return
         endif

      else
         write(out,'(a)',iostat=ioerr)
     $      'Concentration data will be calculated.'
         if (ioerr .ne. 0) then
            call ioreturn(ioerr)
            return
         endif

         equil=.false.
         do 10, ichn=1,nchn
            if (chain(ichn) .eq. 'U_Nat') then
               loc=0
            elseif(chain(ichn) .eq. 'Th_Nat') then
               loc=6
            else
               loc1=max(index(chain(ichn),'+C'),index(chain(ichn),'+c'))
               loc2=max(index(chain(ichn),'_C'),index(chain(ichn),'_c'))
               loc=max(loc1,loc2)
            endif

            if (loc .gt. 0) equil=.true.
10       continue
         if (equil) then
            write(out,'(2a)',iostat=ioerr)
     $         'Concentrations of radionuclides in ',
     $         'equilibrium chains will be'
              if (ioerr .ne. 0) then
                 call ioreturn(ioerr)
                 return
              endif

            if (distrb) then
               write(out,'(2a)',iostat=ioerr)
     $            '   distributed based on initial ',
     $            'concentration given for parent.'
              if (ioerr .ne. 0) then
                 call ioreturn(ioerr)
                 return
              endif

            else
               write(out,'(2a)',iostat=ioerr)
     $            '   equilibrated based on initial ',
     $            'concentration of parent.'
```

```
          if (ioerr .ne. 0) then
              call ioreturn(ioerr)
              return
          endif

        endif
      endif
    endif

    write(out,'(a)',iostat=ioerr) ' '
    if (ioerr .ne. 0) then
       call ioreturn(ioerr)
       return
    endif

    write(out,'(a)',iostat=ioerr) 'PARAMETER DATA:'
    if (ioerr .ne. 0) then
       call ioreturn(ioerr)
       return
    endif

    write(out,'(a,1p,e10.4,a)',iostat=ioerr)
   $   'Time In Building                    : ',
   $   to,' d/y'
    if (ioerr .ne. 0) then
       call ioreturn(ioerr)
       return
    endif

    write(out,'(a,1p,e10.4,a)',iostat=ioerr)
   $   'Occupancy Period                    : ',
   $   tto,' d'
    if (ioerr .ne. 0) then
       call ioreturn(ioerr)
       return
    endif

    write(out,'(a,1p,e10.4,a)',iostat=ioerr)
   $   'Breathing Rate                      : ',
   $   vo,' m**3/h'
    if (ioerr .ne. 0) then
       call ioreturn(ioerr)
       return
    endif

    write(out,'(a,1p,e10.4,a)',iostat=ioerr)
   $   'Resuspension Factor                 : ',
   $   rfo,' /m'
    if (ioerr .ne. 0) then
       call ioreturn(ioerr)
       return
    endif

    write(out,'(a,1p,e10.4,a)',iostat=ioerr)
   $   'Secondary Ingestion Transfer Rate: ',
   $   go,' m**2/h'
    if (ioerr .ne. 0) then
       call ioreturn(ioerr)
       return
```

```
 endif

 write(out,'(a)',iostat=ioerr) ' '
 if (ioerr .ne. 0) then
    call ioreturn(ioerr)
    return
 endif

 write(out,'(a)',iostat=ioerr) 'TIME DATA:'
 if (ioerr .ne. 0) then
    call ioreturn(ioerr)
    return
 endif

 write(out,'(a,1p,e10.4,a)',iostat=ioerr)
$   'Start Time                : ',tstart,
$   ' d'
 if (ioerr .ne. 0) then
    call ioreturn(ioerr)
    return
 endif

 write(out,'(a,1p,e10.4,a)',iostat=ioerr)
$   'End Time                  : ',tend,' d'
 if (ioerr .ne. 0) then
    call ioreturn(ioerr)
    return
 endif

 write(out,'(a,1p,e10.4,a)',iostat=ioerr)
$   'DT Size                   : ',dt,' d'
 if (ioerr .ne. 0) then
    call ioreturn(ioerr)
    return
 endif

 write(out,'(a,1p,e10.4,a)',iostat=ioerr)
$   'Time Step Size            : ',tto,' d'
 if (ioerr .ne. 0) then
    call ioreturn(ioerr)
    return
 endif

 write(out,'(a,i5,a)',iostat=ioerr)
$   'Write results every       : ',skip,
$   ' calculation times'
 if (ioerr .ne. 0) then
    call ioreturn(ioerr)
    return
 endif


 write(out,'(a)',iostat=ioerr) ' '
 if (ioerr .ne. 0) then
    call ioreturn(ioerr)
    return
 endif

 write(out,'(a)',iostat=ioerr) 'INITIAL ACTIVITIES:'
```

```
if (ioerr .ne. 0) then
   call ioreturn(ioerr)
   return
endif

write(out,'(/a,i3)',iostat=ioerr) 'Number of chains: ',nchnp2
if (ioerr .ne. 0) then
   call ioreturn(ioerr)
   return
endif

write(out,'(/a)',iostat=ioerr) ' Chain      Chain     Initial'
if (ioerr .ne. 0) then
   call ioreturn(ioerr)
   return
endif

write(out,'(a)',iostat=ioerr) 'Number       Name    Activity'
if (ioerr .ne. 0) then
   call ioreturn(ioerr)
   return
endif

write(out,'(a)',iostat=ioerr) '                          dpm/'
if (ioerr .ne. 0) then
   call ioreturn(ioerr)
   return
endif

write(out,'(a)',iostat=ioerr) '                       100 cm^2'
if (ioerr .ne. 0) then
   call ioreturn(ioerr)
   return
endif

write(out,'(i6,2x,a,2x,1p,e10.4)',iostat=ioerr)
$   (ichn,chain(ichn),conc(ichn),ichn=1,nchn)
if (ioerr .ne. 0) then
   call ioreturn(ioerr)
   return
endif

end
```

F.65 PF.FOR

```
c
c...this subroutine calculates the soil and water pathway transfer
c   factors (pt) for each radionuclide in a chain.  it also allows
c   for the special c-14 and tritium models
c   See Sections 5.3 through 5.5
c
      subroutine pf(maxnr,nr,carbon,hydrog,bf,bg,bh,bv,d,
     $   fa,fca,fcd,fcf,fcg,fch,fha,fhd,fhf,fhg,fhh,fhv,
     $   ir,lambda,lambdw,mlf,mlg,mlh,mlv,ps,
     $   qd,qf,qg,qh,qw,rf,rg,rh,rv,sasvh,satac,satah,sawvh,sh,
     $   tbigf,tbigg,tbigh,tbigv,tca,tcv,tff,tfg,tfh,tfw,
     $   tgg,tgh,tgv,tha,thv,ua,uv,
     $   wf,wg,wh,wv,xf,xg,xh,xw,yf,yg,yh,yv,pfs,pfw)

      double precision bf,bg,bh,bv
      double precision d
      double precision fa,fca,fcd,fcf,fcg,fch,fha,fhd,fhf,fhg,fhh,fhv
      double precision ir
      double precision lambda,lambdw
      double precision mlf,mlg,mlh,mlv
      double precision pfs,pfw,pptf,ps
      double precision qd,qf,qg,qh,qw
      double precision rf,rg,rh,rv
      double precision sasvh,satac,satah,sawvh,sh
      double precision tbigf,tbigg,tbigh,tbigv
      double precision tca,tcv,tff,tfg,tfh,tfw,tgg,tgh,tgv,tha,thv
      double precision ua,uv
      double precision wf,wg,wh,wv
      double precision xf,xg,xh,xw
      double precision yf,yg,yh,yv
      double precision zero
      integer a
      integer i
      integer j
      integer maxnr,maxpar
      integer nr
      integer v
      logical carbon,hydrog
      parameter (maxpar=20)
      parameter (zero=0.0d+0)
      dimension bv(4),bf(4),bg(4),bh(4)
      dimension d(maxnr,maxnr)
      dimension fa(4),fca(4),fcf(4),fcg(4),fch(4)
      dimension fha(4),fhf(4),fhg(4),fhh(4)
      dimension fhv(4)
      dimension lambda(maxnr)
      dimension mlf(4),mlg(4),mlh(4),mlv(4)
      dimension pfs(maxnr,maxnr)
      dimension pfw(maxnr,maxnr)
      dimension pptf(maxpar,maxpar,4)
      dimension qd(4),qf(4),qg(4),qh(4),qw(4)
      dimension rv(4)
      dimension tbigf(4),tbigg(4),tbigh(4),tbigv(4)
      dimension tca(4),tcv(4),tff(4),tfg(4),tfh(4),tfw(4)
      dimension tgg(4),tgh(4),tgv(4),tha(4),thv(4)
      dimension ua(4),uv(4)
      dimension wf(4),wg(4),wh(4),wv(4)
```

```
      dimension xf(4),xg(4),xh(4)
      dimension yf(4),yg(4),yh(4),yv(4)
c
c...soil pathway transfer factor
c......initialize the soil pathway transfer factor
      do 20, i=1,maxnr
         do 10, j=1,maxnr
            pfs(j,i)=zero
   10    continue
   20 continue
c......calculate the soil partial pathway transfer factor for plants
c      using special tritium model if necessary
      if (hydrog) then
         call sphh(maxnr,sasvh,sh,fhv,tcv,pptf)
      else
         call sph(maxnr,nr,bv,lambda,d,mlv,tcv,tgv,thv,wv,pptf)
      endif
c......sum soil pptf over each edible plant group to get soil
c      pathway transfer factor
      do 50, i=1,nr
         do 40, j=1,nr
            do 30, v=1,4
               pfs(j,i)=pfs(j,i)+uv(v)*pptf(j,i,v)
   30       continue
   40    continue
   50 continue
c......calculate soil partial pathway transfer factor for animals
c      using special c-14 or tritium model if necessary
      if (carbon) then
         call spahc(maxnr,bf,bg,bh,fca,fcd,fcf,fcg,fch,mlf,mlg,mlh,
     $      qd,qf,qg,qh,satac,tca,tff,tfg,tfh,wf,wg,wh,xf,xg,xh,pptf)
      elseif (hydrog) then
         call spahh(maxnr,fha,fhd,fhf,fhg,fhh,qd,qf,qg,qh,qw,
     $      sasvh,satah,sh,tca,tff,tfg,tfh,tfw,wf,xf,xg,xh,pptf)
      else
         call spah(maxnr,nr,bf,bg,bh,d,fa,lambda,mlf,mlg,mlh,
     $      qd,qf,qg,qh,tca,tff,tgg,tgh,tha,wf,wg,wh,xf,xg,xh,pptf)
      endif
c......sum soil pptf over each animal plant group to get soil
c      pathway transfer factor
      do 80, i=1,nr
         do 70, j=1,nr
            do 60, a=1,4
               pfs(j,i)=pfs(j,i)+ua(a)*pptf(j,i,a)
   60       continue
   70    continue
   80 continue
c
c...water pathway transfer factor
c......initialize the water pathway transfer factor
      do 100, i=1,nr
         do 90, j=1,nr
            pfw(j,i)=zero
   90    continue
  100 continue
c......calculate the water partial pathway transfer factor (pptf)
c      for plants using special tritium model if necessary
      if (hydrog) then
         call wphh(maxnr,fhv,sawvh,tcv,pptf)
```

```
      else
         call wph(maxnr,nr,bv,d,ir,lambda,lambdw,mlv,ps,rv,
     $      tbigv,tcv,tgv,thv,wv,yv,pptf)
      endif
c......sum water pptf over each edible plant group to get water
c      pathway transfer factor
      do 130, i=1,nr
         do 120, j=1,nr
            do 110, v=1,4
               pfw(j,i)=pfw(j,i)+uv(v)*pptf(j,i,v)
  110       continue
  120    continue
  130 continue
c......calculate water partial pathway transfer factor for animals
c      using special c-14 or tritium model if necessary
      if (carbon) then
         call wpahc(maxnr,bf,bg,bh,d,fca,fcd,fcf,fcg,fch,
     $      ir,lambda,lambdw,mlf,mlg,mlh,ps,qd,qf,qg,qh,qw,rf,rg,rh,
     $      satac,tbigf,tbigg,tbigh,tca,tff,tfg,tfh,tfw,tgg,tgh,
     $      wf,wg,wh,xf,xg,xh,xw,yf,yg,yh,pptf)
      elseif (hydrog) then
         call wpahh(maxnr,fha,fhd,fhf,fhg,fhh,ir,ps,qd,qf,qg,qh,qw,
     $      satah,sawvh,tca,tff,tfg,tfh,tfw,wf,xf,xg,xh,xw,pptf)
      else
         call wpah(maxnr,nr,bf,bg,bh,d,fa,ir,lambda,lambdw,
     $      mlf,mlg,mlh,ps,qd,qf,qg,qh,qw,rf,rg,rh,tbigf,tbigg,tbigh,
     $      tca,tff,tgg,tgh,tha,wf,wg,wh,yf,yg,yh,xf,xg,xh,xw,pptf)
      endif
c......sum water pptf over each animal plant group to get water
c      pathway transfer factor
      do 160, i=1,nr
         do 150, j=1,nr
            do 140, a=1,4
               pfw(j,i)=pfw(j,i)+ua(a)*pptf(j,i,a)
  140       continue
  150    continue
  160 continue
      end
```

F.66 PTHMAX.FOR

```
c       This subroutine checks for the maximum pathway doses;
c       save them and their time of occurrence
c
      subroutine pthmax(mxpath,npath,tcalc,period,tpath,tppath,
     $    pathup,pathmx)

      double precision pathmx,pathup
      double precision tcalc,tpath
      integer ipath
      integer mxpath
      integer npath
      integer period,tppath
      dimension pathmx(mxpath),pathup(mxpath)
      dimension tpath(mxpath)
      dimension tppath(mxpath)

      do 10, ipath=1,npath
         if (pathup(ipath) .gt. pathmx(ipath)) then
            pathmx(ipath)=pathup(ipath)
            tpath(ipath)=tcalc
            tppath(ipath)=period
         endif
   10 continue
      end
```

F.67 PTHSUM.FOR

```
c
c...this subroutine calculates pathway dose for a chain
c
      subroutine pthsum(mxnr,mxnip,mxpath,nr,nip,npath,dose,path)

      double precision dose
      double precision path
      double precision zero
      integer iip,ipath,ir
      integer mxnip,mxnr,mxpath
      integer nip,npath,nr
      dimension dose(mxnr,0:mxnip,mxpath)
      dimension nip(mxnr)
      dimension path(mxpath)
      parameter (zero=0.0d+0)
c
c...initialize path doses
      do 10, ipath=1,npath
         path(ipath)=zero
   10 continue
c
c...calculate pathway doses
      do 40, ipath=1,npath
         do 30, ir=1,nr
            do 20, iip=0,nip(ir)
               path(ipath)=path(ipath)+dose(ir,iip,ipath)
   20       continue
   30    continue
   40 continue
      end
```

F.68 RDBIO.FOR

```
c
c...this subroutine reads in the bioaccumulation factors.
c
      subroutine rdbio(in,out,for_rc,maxnr,nr,nuc,parnam,baf,ioerr)

      character*2 nuc
      character*6 parnam
      character*80 string
      double precision baf
      integer i,in,iochk,ioerr
      integer maxnr
      integer nr
      integer out
      integer*4 for_rc
      dimension baf(maxnr)
      dimension nuc(maxnr)
      dimension parnam(maxnr)
c
      do 10, i=1,nr

         rewind(in,iostat=ioerr)

          if (ioerr .ne. 0) then
              call ioreturn(ioerr)
              return
          endif


    5    read(in,'(a)',iostat=iochk) string

         if (iochk .lt. 0) then

            write(out,'(2a)',iostat=ioerr)
     $         'There is no bioaccumulation factor for ',parnam(i)

              if (ioerr .ne. 0) then
                  call ioreturn(ioerr)
                  return
              endif

            for_rc=0
            return

         elseif (iochk .gt. 0) then

            ioerr=iochk
            call ioreturn(ioerr)
            return

         endif

         if (nuc(i) .eq. string(1:2)) then

ccc  format problems          read(string(5:12),'(bz,f8.0)') baf(i)
ccc   changed to the following
             read(string(5:80),'(e12.0)',iostat=ioerr) baf(i)
```

```
            if (ioerr .ne. 0) then
               call ioretint(ioerr)
               return
            endif


         else
            goto 5
         endif

   10 continue
      end
```

F.69 RDCEDE.FOR

```
c     This subroutine reads the internal committed effective dose file
c     from data\cede.dat
c

      include "cdecl.inc"

      subroutine rdcede(in,out,rc,maxpar,maxpro,combin,nr,nip,
     $    nipdis,parnam,prfrac,dfer,dfes,dfg,dfh,ioerr)

      character*6 parnam,partmp
      character*80 string
      character*80 status_msg
      double precision dfer,dfertp,dfes,dfestp,dfg,dfgtmp,dfh,dfhtmp
      double precision one
      double precision prfrac,prftmp
      double precision zero
      integer c_rc,rc,i,in,iochk,ioerr
      integer j
      integer maxpar,maxpro
      integer nip,nipdis,nr
      integer out
      logical combin
      parameter (one=1.0d+0,zero=0.0d+0)
      dimension dfer(maxpar,0:maxpro)
      dimension dfg(maxpar,0:maxpro)
      dimension dfh(maxpar,0:maxpro)
      dimension dfes(maxpar,0:maxpro)
      dimension nip(maxpar),nipdis(maxpar)
      dimension parnam(maxpar,0:maxpro)
      dimension prfrac(maxpar,0:maxpro)
c
c......initialize arrays
      do 20, j=0,maxpro
         do 10, i=1,maxpar
            if (j .ne. 0) parnam(i,j)='      '
            prfrac(i,j)=zero
            dfg(i,j)=zero
            dfh(i,j)=zero
            dfes(i,j)=zero
            dfer(i,j)=zero
   10    continue
   20 continue
c
c...read the internal committed effective dose file, this will fill
c   in the implicit radionuclides
      do 50, i=1,nr
         rewind(in,iostat=ioerr)
          if (ioerr .ne. 0) then
              call ioreturn(ioerr)
              return
          endif

   25    read(in,'(a)',iostat=iochk) string

         if (iochk .lt. 0) then
ccc            write(*,'(2a)') ' There are no CEDE data for ',parnam(i,0)
```

```
        rc = 0

        write(status_msg(1:6),'(a)',iostat=ioerr) parnam(i,0)

        if (ioerr .ne. 0) then
            call ioretint(ioerr)
            return
        endif

        status_msg = status_msg(1:6)//
$               ' does not exist in CEDE database'C
        call CMessage(status_msg,c_rc)

        write(out,'(2a)',iostat=ioerr)
$        'There are no CEDE data for ',parnam(i,0)

        if (ioerr .ne. 0) then
            call ioreturn(ioerr)
            return
        endif

        write(out,'(1a)',iostat=ioerr) 'Exiting Fortran Codes!'
        if (ioerr .ne. 0) then
            call ioreturn(ioerr)
            return
        endif

        return
    endif


    if (iochk .gt. 0) then
        call ioreturn(iochk)
        ioerr = iochk
        return
    endif

    if (string(2:7) .eq. parnam(i,0) .and.
$       string(9:15) .eq. '        ') then

       prfrac(i,0)=one

       read(string(16:25),'(bz,f10.0)',iostat=ioerr) dfg(i,0)
       if (ioerr .ne. 0) then
         call ioretint(ioerr)
         return
       endif

       read(string(26:35),'(bz,f10.0)',iostat=ioerr) dfh(i,0)
       if (ioerr .ne. 0) then
         call ioretint(ioerr)
         return
       endif

       read(string(36:45),'(bz,f10.0)',iostat=ioerr) dfes(i,0)
       if (ioerr .ne. 0) then
         call ioretint(ioerr)
         return
       endif
```

```
            read(string(46:55),'(bz,f10.0)',iostat=ioerr) dfer(i,0)
            if (ioerr .ne. 0) then
              call ioretint(ioerr)
              return
            endif

            nip(i)=0
            nipdis(i)=0
c......now read the implicit progeny
   35       read(in,'(a)',end=45,iostat=ioerr) string
            if (ioerr .gt. 0) then
              call ioreturn(ioerr)
              return
            endif

c......if the implicit progeny fraction column is not blank, then we
c      have an implicit progeny for the i'th explicit radionuclide
            if (string(9:15) .ne. '       ') then
               read(string(2:7),'(a)',iostat=ioerr) partmp
               if (ioerr .ne. 0) then
                  call ioretint(ioerr)
                  return
               endif

               read(string(9:15),'(bz,f7.0)',iostat=ioerr) prftmp
               if (ioerr .ne. 0) then
                  call ioretint(ioerr)
                  return
               endif

               read(string(16:25),'(bz,f10.0)',iostat=ioerr) dfgtmp
               if (ioerr .ne. 0) then
                  call ioretint(ioerr)
                  return
               endif

               read(string(26:35),'(bz,f10.0)',iostat=ioerr) dfhtmp
               if (ioerr .ne. 0) then
                  call ioretint(ioerr)
                  return
               endif

               read(string(36:45),'(bz,f10.0)',iostat=ioerr) dfestp
               if (ioerr .ne. 0) then
                  call ioretint(ioerr)
                  return
               endif

               read(string(46:55),'(bz,f10.0)',iostat=ioerr) dfertp
               if (ioerr .ne. 0) then
                  call ioretint(ioerr)
                  return
               endif

c.........if required combine dose factors into implicit progeny's
c         explicit parent
               if (combin) then
                  nipdis(i)=nipdis(i)+1
                  prfrac(i,nipdis(i))=prftmp
```

```
                  dfg(i,0)=dfg(i,0)+prftmp*dfgtmp
                  dfh(i,0)=dfh(i,0)+prftmp*dfhtmp
                  dfes(i,0)=dfes(i,0)+prftmp*dfestp
                  dfer(i,0)=dfer(i,0)+prftmp*dfertp
c.........otherwise, store as implicit progeny
               else
                  nip(i)=nip(i)+1
                  nipdis(i)=nipdis(i)+1
                  parnam(i,nip(i))=partmp
                  prfrac(i,nip(i))=prftmp
                  dfg(i,nip(i))=dfgtmp
                  dfh(i,nip(i))=dfhtmp
                  dfes(i,nip(i))=dfestp
                  dfer(i,nip(i))=dfertp
               endif
               go to 35
            else
               go to 45
            endif
         else
            go to 25
         endif
   45    continue
   50 continue
      end
```

F.70 RDCHN.FOR

```
c
c...This subroutine reads the chain file (data\chain.dat) to
c   set up the explicit radionuclides in the chain. it also
c   calculates halflifes of the explicit radionuclides.

      include "cdecl.inc"

      subroutine rdchn(in,out,rc,maxpar,maxpro,chain,nr,parnam,
     $   halflf,lambda,ipos,ipar,frac,ioerr)
      character*6 parnam
      character*8 chain
      character*80 string
      character*80 status_msg
      double precision frac
      double precision halflf
      double precision lambda
      double precision one
      double precision two
      double precision zero
      integer rc,i,in,index,iochk,ioerr,ipos,ipar
      integer maxpar,maxpro
      integer nr
      integer out
      integer c_rc
      parameter (one=1.0d+0,two=2.0d+0,zero=0.0d+0)
      dimension frac(2,maxpar)
      dimension halflf(maxpar)
      dimension ipar(2,maxpar)
      dimension ipos(maxpar)
      dimension lambda(maxpar)
      dimension parnam(maxpar,0:maxpro)

c
c...read the chain data
      nr=0
      rewind(in,iostat=ioerr)
      if (ioerr .ne. 0) then
         call ioreturn(ioerr)
         return
      endif

c......read a string of information in order to find parent; if we
c      are at the end of a file and no parent is found, return error
    5 string='
      read(in,'(a)',iostat=iochk) string

      if (iochk .lt. 0) then
          rc = 0

          write(status_msg(1:6),'(a)',iostat=ioerr) chain

          if (ioerr .ne. 0) then
              call ioretint(ioerr)
              return
          endif

          status_msg = status_msg(1:6)//
```

```
$                 ' does not exist in chain database'C

     call CMessage(status_msg,c_rc)

     write(out,'(2a)',iostat=ioerr)
$      chain,' does not exist in database'

         if (ioerr .ne. 0) then
             call ioreturn(ioerr)
             return
         endif

    write(out,'(1a)',iostat=ioerr) 'Exiting Fortran Codes!'
     if (ioerr .ne. 0) then
         call ioreturn(ioerr)
         return
     endif

    return
  endif

  if (iochk .gt. 0) then
    call ioreturn(iochk)
    ioerr = iochk
    return
  endif

c......if the string contains the parent radionuclide, and its
c      position within the chain is 1, store the data;
c      otherwise, read next record as a string
  if (string(2:9) .eq. chain .and. string(18:19) .eq. ' 1') then
     nr=nr+1
     read(string(2:7),'(a)',iostat=ioerr) parnam(nr,0)
      if (ioerr .ne. 0) then
          call ioretint(ioerr)
          return
      endif

     read(string(10:17),'(bz,f8.0)',iostat=ioerr) halflf(nr)
      if (ioerr .ne. 0) then
          call ioretint(ioerr)
          return
      endif

     read(string(18:19),'(bz,i2)',iostat=ioerr) ipos(nr)
      if (ioerr .ne. 0) then
          call ioretint(ioerr)
          return
      endif

     do 10, i=1,2
        index=20+9*(i-1)
        read(string(index:index+1),'(bz,i2)',iostat=ioerr)
$          ipar(i,nr)
        if (ioerr .ne. 0) then
          call ioretint(ioerr)
          return
        endif
```

```
            index=index+3
            read(string(index:index+5),'(bz,f6.0)',iostat=ioerr)
     $         frac(i,nr)
              if (ioerr .ne. 0) then
               call ioretint(ioerr)
               return
              endif

   10     continue
          lambda(nr)=log(two)/halflf(nr)
c.........now read a string for potential explicit progeny;
c          repeat as necessary; if we reach end of file jump
c          out of loop
   15     string='
          read(in,'(a)',end=25,iostat=ioerr) string
           if (ioerr .gt. 0) then
               call ioreturn(ioerr)
               return
           endif

c.........if the chain position characters of the string are blank,
c          then the string is an implicit progeny string, the data
c          are neglected and the next string is read; if the position
c          character is ' 1', the radionuclide represents a new chain
c          and we are done; if the position character is greater than
c          1 the data are stored and the next string is read
          if (string(18:19) .eq. '  ') then
             go to 15
          elseif (string(18:19) .eq. ' 1') then
             go to 25
          elseif (string(18:19) .eq. 'ta') then
             go to 25
          else
             nr=nr+1
             read(string(2:7),'(a)',iostat=ioerr) parnam(nr,0)
               if (ioerr .ne. 0) then
                   call ioretint(ioerr)
                   return
               endif

             read(string(10:17),'(bz,f8.0)',iostat=ioerr) halflf(nr)
               if (ioerr .ne. 0) then
                   call ioretint(ioerr)
                   return
               endif

             read(string(18:19),'(bz,i2)',iostat=ioerr) ipos(nr)
               if (ioerr .ne. 0) then
                   call ioretint(ioerr)
                   return
               endif

             do 20, i=1,2
                index=20+9*(i-1)
                read(string(index:index+1),'(bz,i2)',iostat=ioerr)
     $             ipar(i,nr)
               if (ioerr .ne. 0) then
                   call ioretint(ioerr)
                   return
```

```
          endif

           index=index+3
           read(string(index:index+5),'(bz,f6.0)',iostat=ioerr)
 $            frac(i,nr)
          if (ioerr .ne. 0) then
              call ioretint(ioerr)
              return
          endif

20       continue
         lambda(nr)=log(two)/halflf(nr)
         go to 15
      endif
   else
      go to 5
   endif
25 continue
   end
```

F.71 RDCONC.FOR

```
c
c...This subroutine reads in the soil-to-plant-concentration factors
c
      subroutine rdconc(in,out,for_rc,maxnr,nr,nuc,parnam,bv,bf,bg,bh,
     $                  ioerr)

      character*2 nuc
      character*6 parnam
      character*80 string
      double precision bf,bg,bh,bv
      double precision zero
      integer i,in,iochk,ioerr
      integer j
      integer maxnr
      integer nr
      integer*4 for_rc
      integer out
      dimension bf(4,maxnr),bg(4,maxnr),bh(4,maxnr),bv(4,maxnr)
      dimension nuc(maxnr)
      dimension parnam(maxnr)
      parameter (zero=0.0d+0)
c
      do 30, i=1,nr
         rewind (in, iostat=ioerr)

          if (ioerr .ne. 0) then
            call ioreturn(ioerr)
            return
          endif


    5    read(in,'(a)',iostat=iochk) string

         if (iochk .lt. 0) then

            write(out,'(2a)',iostat=ioerr)
     $         'There is no concentration factor for ',parnam(i)

              if (ioerr .ne. 0) then
                  call ioreturn(ioerr)
                  return
              endif

            for_rc = 0
            return

         elseif (iochk .gt. 0) then

            ioerr=iochk
            call ioreturn(ioerr)
            return

         endif

         if (nuc(i) .eq. string(1:2)) then

ccc            do 10, j=1,4
```

```
ccc                if (string(9*j:9*j+2) .eq. '(-)') then
ccc                   bv(j,i)=zero
ccc                else
ccc                   read(string(9*j:9*j+7),'(bz,f8.0)') bv(j,i)
ccc                endif
ccc   10        continue
ccc   Format problems, the above ccc's were changed to the following
ccc    MAW, 03/05/96

            if (index(string(8:80),'(-)') .ne. 0) then

               do 10 j=1,4
                  bv(j,i) = zero
10             continue

            else

               read(string(8:80),'(e12.0)',iostat=ioerr) (bv(j,i),j=1,4)

              if (ioerr .ne. 0) then
                  call ioretint(ioerr)
                  return
              endif

            endif

            do 20, j=1,4
               bf(j,i)=bv(1,i)
               bg(j,i)=bv(4,i)
               bh(j,i)=bv(4,i)
   20       continue

         else

            goto 5

         endif
   30 continue
      end
```

F.72 RDDFO.FOR

```
c     This subroutine reads in the ingestion organ
c     dose equivalent factors from tables E.3-E.5
c

      subroutine rddfo(in,out,rc,maxpar,maxpro,maxorg,combin,nr,
     $    parnam,dfo,ioerr)

      character*6 parnam
      character*90 string
      double precision dfo,dfotmp
      double precision one
      double precision prftmp
      double precision zero
      integer i,in,indx,iochk,ioerr
      integer j
      integer k
      integer maxorg,maxpar,maxpro,mxorg
      integer nip,nr
      integer out
      integer rc
      logical combin
      parameter (mxorg=7)
      parameter (one=1.0d+0,zero=0.0d+0)
      dimension dfo(maxorg,maxpar,0:maxpro)
      dimension dfotmp(mxorg)
      dimension parnam(maxpar,0:maxpro)

c
c......initialize arrays
      do 30, j=0,maxpro

         do 20, i=1,maxpar

            do 10, k=1,maxorg/3
               dfo(k,i,j)=zero
   10       continue

   20    continue

   30 continue
c
c...read the internal committed effective dose file, this will fill
c   in the implicit radionuclides
      do 100, i=1,nr

         rewind(in,iostat=ioerr)

         if (ioerr .ne. 0) then
              call ioreturn(ioerr)
              return
         endif

   25    read(in,'(a)',iostat=iochk) string

         if (iochk .lt. 0) then
ccc            write(*,'(2a)') ' There are no DFO data for ',parnam(i,0)
            write(out,'(2a)',iostat=ioerr)
```

```
     $          'There are no DFO data for ',parnam(i,0)

              if (ioerr .ne. 0) then
                  call ioreturn(ioerr)
                  return
              endif

            rc = 0
ccc            stop ' Stopping program'
            return

          elseif (iochk .gt. 0) then

            ioerr=iochk
            call ioreturn(iochk)
            return

          endif

          if (string(2:7) .eq. parnam(i,0) .and.
     $       string(9:15) .eq. '        ') then

            if (string(18:22) .eq. 'Note:') then

              do 40, k=1,7
                 dfo(k,i,0)=zero
   40         continue

            else

              do 50, k=1,7
                 indx=k*10+6
                 read(string(indx:indx+9),'(bz,f10.0)',iostat=ioerr)
     $              dfo(k,i,0)

                 if (ioerr .ne. 0) then
                     call ioretint(ioerr)
                     return
                 endif

   50         continue

            endif

            nip=0
c......now read the implicit progeny
   55       read(in,'(a)',end=95,iostat=ioerr) string

              if (ioerr .ne. 0) then
                  call ioreturn(ioerr)
                  return
              endif

c......if the implicit progeny fraction column is not blank, then we
c      have an implicit progeny for the i'th explicit radionuclide
            if (string(9:15) .ne. '        ') then
               read(string(9:15),'(bz,f7.0)',iostat=ioerr) prftmp

                 if (ioerr .ne. 0) then
```

```
                call ioretint(ioerr)
                return
             endif

          if (string(18:22) .eq. 'Note:') then

             do 60, k=1,7
                dfotmp(k)=zero
 60           continue

          else

             do 70, k=1,7
                indx=k*10+6
                read(string(indx:indx+9),'(bz,f10.0)',
  $                iostat=ioerr) dfotmp(k)

                 if (ioerr .ne. 0) then
                     call ioretint(ioerr)
                     return
                 endif

 70          continue
          endif

c.........if required combine dose factors into implicit progeny's
c         explicit parent
          if (combin) then

             do 80, k=1,7
                dfo(k,i,0)=dfo(k,i,0)+prftmp*dfotmp(k)
 80          continue

c.........otherwise, store as implicit progeny
          else
             nip=nip+1

             do 90, k=1,7
                dfo(k,i,nip)=dfotmp(k)
 90          continue

          endif

          go to 55
       else

          go to 95

       endif

    else

       go to 25

    endif
 95 continue
100 continue

    end
```

F.73 RDKD.FOR

```
c
c     This subroutine reads in the partition coefficients from
c     the data\kd.dat file.
c
      subroutine rdkd(in,out,rc,maxnr,nr,nuc,parnam,kd,ioerr)

      character*2 nuc
      character*6 parnam
      character*80 string
      double precision kd
      integer i,in,iochk,ioerr
      integer maxnr
      integer nr
      integer out
      integer rc
      dimension kd(maxnr)
      dimension nuc(maxnr)
      dimension parnam(maxnr)
c
ccc             read(string(5:10),'(bz,f6.0)') kd(i)
      do 10, i=1,nr

         rewind (in,iostat=ioerr)

         if (ioerr .ne. 0) then
              call ioreturn(ioerr)
              return
         endif


    5    read(in,'(a)',iostat=iochk) string

         if (iochk .lt. 0) then
            write(out,'(2a)',iostat=ioerr)
     $          'There is no partition coefficient for ',parnam(i)

              if (ioerr .ne. 0) then
                  call ioreturn(ioerr)
                  return
              endif

            rc = 0
            return

          elseif (iochk .gt. 0) then
              ioerr=iochk
              call ioreturn(ioerr)
              return
          endif

         if (nuc(i) .eq. string(1:2)) then
ccc             read(string(5:12),'(bz,f8.0)') kd(i)
ccc format problems, changed to the following, MAW 03/05/96
            read(string(5:80),'(e11.0)',iostat=ioerr) kd(i)

              if (ioerr .ne. 0) then
                  call ioreturn(ioerr)
```

```
            return
         endif

      else
         goto 5
      endif

10 continue
   end
```

F.74 RDLIST.FOR

```
c
c...this subroutine reads the chain.dat file and makes a
c   unique list of all explicit radionuclides in the file.
c   if a radionuclide is explicit in some chains and implicit
c   in others, it will be treated as explicit in this routine.
c
      subroutine rdlist(inunit,out,maxnam,ierr,nnam,nucnam,ioerr)

      character*6 nuc,nucnam
      character*80 strng
      integer ierr,inam,indx,inunit,ioerr
      integer loc
      integer maxnam
      integer nnam,nnamtm
      integer out
      logical inlist
      dimension nucnam(maxnam)
c
c...rewind chain data file
      rewind(inunit,iostat=ioerr)

      if (ioerr .ne. 0) then
            call ioreturn(ioerr)
            return
      endif

      nnam=0
c
c...read the radionuclide data
    5 read(inunit,'(a)',end=15,iostat=ioerr) strng

      if (ioerr .ne. 0) then
            call ioreturn(ioerr)
            return
      endif


      nuc=strng(2:7)
      loc=max(index(nuc,'+'),index(nuc,'_'))

      if (loc .eq. 0) then
         read(strng,'(1x,a,10x,i2)',iostat=ioerr) nuc,indx

          if (ioerr .ne. 0) then
              call ioretint(ioerr)
              return
          endif

      else

         goto 5

      endif


c......if the radionuclide has a chain index number, it is explicit
c       and can be added to the list
```

```
      if (indx .gt. 0) then

c.........see if it is already in the list
         inlist=.false.
         nnamtm=nnam

         do 10, inam=1,nnamtm
            if (nuc .eq. nucnam(inam)) inlist=.true.
   10    continue

c.........if not, then add it to list

         if (.not. inlist) then
            nnam=nnam+1

            if (nnam .gt. maxnam) then
               write(out,'(2a,2(i4,a))',iostat=ioerr)
     $            'Number of unique explicit ',
     $            ' radionuclides in datafile (',nnam,
     $            ') exceeds maximum (',maxnam,')'

              if (ioerr .ne. 0) then
                  call ioreturn(ioerr)
                  return
              endif

               ierr=201

            endif

            nucnam(nnam)=nuc

         endif

      endif

c......read next record in data file
      goto 5

   15 continue

      end
```

F.75 RDTRAN.FOR

```
c
c...this subroutine reads in the animal product transfer factors
c
      subroutine rdtran(in,out,for_rc,maxnr,nr,nuc,parnam,fa,ioerr)

      character*2 nuc
      character*6 parnam
      character*80 string
      double precision fa
      double precision zero
      integer i,in,iochk,ioerr
      integer j
      integer nr
      integer maxnr
      integer out
      integer*4 for_rc
      dimension parnam(maxnr)
      dimension fa(4,maxnr)
      dimension nuc(maxnr)
      parameter (zero=0.0d+0)
c
      do 20, i=1,nr

         rewind (in, iostat=ioerr)

          if (ioerr .ne. 0) then
              call ioreturn(ioerr)
              return
          endif

    5    read(in,'(a)',iostat=iochk) string

         if (iochk .lt. 0) then
            write(out,'(2a)',iostat=ioerr)
     $         'There is no transfer factor for ',parnam(i)

              if (ioerr .ne. 0) then
                  call ioreturn(ioerr)
                  return
              endif

            for_rc=0
            return

         elseif (iochk .gt. 0) then

            ioerr=iochk
            call ioreturn(ioerr)
            return

         endif


         if (nuc(i) .eq. string(1:2)) then
ccc            do 10, j=1,4
ccc               if (string(9*j:9*j+2) .eq. '(-)') then
ccc                  fa(j,i)=zero
```

```
ccc                 else
ccc                    read(string(9*j:9*j+7),'(bz,f8.0)') fa(j,i)
ccc                 endif
ccc   10         continue
ccc   Format problems, the above ccc's were changed to the following
ccc    MAW, 03/05/96
            if (index(string(8:80),'(-)') .ne. 0) then

               do 10 j=1,4
                  fa(j,i) = zero
10             continue

            else

               read(string(8:80),'(e12.0)',iostat=ioerr) (fa(j,i),j=1,4)

              if (ioerr .ne. 0) then
                  call ioretint(ioerr)
                  return
              endif

            endif

         else
            goto 5
         endif

   20 continue
      end
```

F.76 RENDEF.FOR

```
c
c...this subroutine reads the building renovation scenario default
c   parameter file. The variables read in are:
c   time on the job (t), renovation period (tt), breathing rate (v),
c   dust-loading (dust), ingestion rate (g), start time (tstart),
c   end time (tend), time step size (dt), history file time step
c   skip factor (skip)
c
      subroutine rendef(in,t,tt,v,dust,g,tstart,tend,dt,skip,ioerr)

      double precision dust
      double precision g
      double precision t,tt
      double precision v
      double precision dt
      double precision tend,tstart,tskip
      integer in,ioerr,skip

      read(in,*,iostat=ioerr) t,tt,v,dust,g,tstart,tend,dt,tskip

      if (ioerr .ne. 0) then
            call ioreturn(ioerr)
            return
      endif

      skip = tskip
      end
```

F.77 RENDOS.FOR

```
c
c...this subroutine calculates the doses for each radionuclide
c   in a chain for each pathway for the building renovation scenario
c
      subroutine rendos(mxnr,mxnip,nr,nip,
     $   t,dfer,dfh,dfg,tb,ttb,vb,cdb,gb,
     $   ci,lambda,f,prfrac,
     $   c,sbar,dexb,dhb,dgb)

      double precision c,cdb,ci
      double precision dexb,dfer,dfg,dfh,dgb,dhb
      double precision f
      double precision gb
      double precision lambda
      double precision prfrac
      double precision sbar
      double precision t,tb,ttb,twnty4
      double precision vb
      integer iip,ir
      integer mxnip,mxnr
      integer nip,nr
      dimension c(mxnr,0:mxnip),ci(mxnr)
      dimension dexb(mxnr,0:mxnip),dfer(mxnr,0:mxnip)
      dimension dfg(mxnr,0:mxnip),dfh(mxnr,0:mxnip)
      dimension dgb(mxnr,0:mxnip),dhb(mxnr,0:mxnip)
      dimension nip(mxnr)
      dimension f(mxnr,mxnr)
      dimension lambda(nr)
      dimension prfrac(mxnr,0:mxnip)
      dimension sbar(mxnr,0:mxnip)
      parameter (twnty4=2.4d+1)

c
c...calculate the activity of each radionuclide at the beginning
c   of the time of interest
      call aoper(ci,c,lambda,f,t,nr)

c
c...integrate the activity over the time in building
      call soper(c,sbar,lambda,f,ttb,nr)

c
c...calculate average activity for each explicit radionuclide
      do 10, ir=1,nr
         sbar(ir,0)=sbar(ir,0)/ttb
   10 continue

c
c...calculate the dose for each radionuclide and pathway in the chain
      do 30, ir=1,nr
         do 20, iip=0,nip(ir)
            c(ir,iip)=c(ir,0)*prfrac(ir,iip)
            sbar(ir,iip)=sbar(ir,0)*prfrac(ir,iip)

c......external dose from eq. (3.4)
            dexb(ir,iip)=twnty4*tb*dfer(ir,iip)*sbar(ir,iip)
```

```
c......inhalation dose from eq. (3.8)
            dhb(ir,iip)=twnty4*tb*vb*cdb*dfh(ir,iip)*sbar(ir,iip)

c......ingestion dose from eq. (3.10)
            dgb(ir,iip)=twnty4*tb*gb*dfg(ir,iip)*sbar(ir,iip)

   20    continue
   30 continue
      end
```

F.78 RENMAIN.FOR

```
c***********************************************************************
c
c  PROGRAM RENOVA; v2.0; 9/94
c  C. D. Updegraff, GRAM, Inc. Albuquerque, NM 87112.
c  Solves the building renovation equations from Kennedy, Jr., W.E.
c  and D.L. Strenge, 1992, "Residual Radioactive Contamination From
c  Decommissioning: Technical Basis for Translating Contamination
c  Levels to Annual Total Effective Dose Equivalent," NUREG/CR-5512,
c  PNL-7994, Vol. 1, Pacific Northwest Laboratory, Richland, WA 99352.
c  Benchmarked against RADDECAY Version 3.01 11/92 for the U-238 chain
c
c***********************************************************************
c
      include "cdecl.inc"

      subroutine renova(for_rc)

      character*6 list
      character*6 nuclid
      character*6 short
      character*10 adate
      character*8 chain
      character*11 atime
      character*28 long
      character*80 filfil
      character*80 chainfil,cedefil,occfil
      character*80 hisfil,confil
      character*80 infil
      character*80 outfil
      character*80 title
      character*80 status_msg
      double precision c,cdb,con,conc,cup
      double precision dfer,dfes,dfg,dfh,dos,dosmx,dossv,dosup
      double precision d,dt
      double precision frac
      double precision gb
      double precision hlif
      double precision lambda
      double precision nuc,nucmx,nucsv,nucup
      double precision path,pathmx,pathsv,pathup
      double precision prfrac
      double precision sbar,sbarml,sbarsv,sbarup
      double precision tb,ttb,tdos,tedemx,tend
      double precision tnuc,tpath,tstart,ttede
      double precision vb
      double precision zero
      integer conunt
      integer hisfn,infn,outfn,filefn,chainfn,cedefn
      integer ichn,indx,ioerr,ipar,ipos
      integer mxchn,mxlist,mxpar,mxpath,mxpro
      integer nchn,nchnp2,nip,nipdis,nlist,npath,nr
      integer period,printed,persv,tppath,tpdos,tpnuc
      integer skip
      integer*4 for_rc
      integer*4 c_rc
CCC...added to set the date of the run
      integer*2 month,day,year
```

```
integer*2 hour,minute,second,hsecond
logical combin,import
logical echain
logical histry
logical distrb
parameter (mxchn=22,mxpar=20,mxpath=3,mxpro=6,npath=3)
parameter (mxlist=mxchn*mxpar)
parameter (zero=0.0d+0)
dimension c(mxpar,0:mxpro)
dimension chain(mxchn)
dimension con(mxpar,mxchn)
dimension conc(mxchn)
dimension cup(mxlist)
dimension d(mxpar,mxpar,mxchn)
dimension dfer(mxpar,0:mxpro,mxchn),dfes(mxpar,0:mxpro,mxchn)
dimension dfg(mxpar,0:mxpro,mxchn),dfh(mxpar,0:mxpro,mxchn)
dimension dos(mxpar,0:mxpro,mxpath)
dimension dosmx(mxlist,mxpath)
dimension dossv(mxlist,mxpath)
dimension dosup(mxlist,mxpath)
dimension echain(mxchn)
dimension frac(2,mxpar,mxchn)
dimension hlif(mxpar,mxchn)
dimension indx(mxpar,0:mxpro,mxchn)
dimension ipar(2,mxpar,mxchn)
dimension ipos(mxpar,mxchn)
dimension lambda(mxpar,mxchn)
dimension list(mxlist)
dimension nip(mxpar,mxchn),nipdis(mxpar,mxchn)
dimension nr(mxchn)
dimension nuc(mxpar,0:mxpro)
dimension nuclid(mxpar,0:mxpro,mxchn)
dimension nucmx(mxlist),nucsv(mxlist),nucup(mxlist)
dimension path(mxpath),pathmx(mxpath)
dimension pathsv(mxpath),pathup(mxpath)
dimension prfrac(mxpar,0:mxpro,mxchn)
dimension sbar(mxpar,0:mxpro)
dimension sbarml(mxlist),sbarsv(mxlist),sbarup(mxlist)
dimension tdos(mxlist,mxpath)
dimension tnuc(mxlist)
dimension tpath(mxpath)
dimension tppath(mxpath)
dimension tpdos(mxlist,mxpath)
dimension tpnuc(mxlist)
data long /'Building Renovation Scenario'/
data short /'RENOVA'/

for_rc = 1
c_rc = 1

filefn  = 50
status_msg(1:80) = "Starting Calculational Modules..."C

call CMessage(status_msg,c_rc)
if (c_rc .eq. 0) then
  return
endif
```

```
c
CCC       call date(adate)
CCC       call time(atime)
CCC       call timer(tick1)
CCC...converted to Microsoft Powerstation FORTRAN functions
      call GetDat(year,month,day)
      write(adate,'(I2.2,A,I2.2,A,I4.4)',iostat=ioerr)
     $   month, '/', day, '/', year

      if (ioerr .ne. 0) then
          for_rc = 0
          call ioretint(ioerr)
          call closefiles(filefn,10)
          return
      endif

      call GetTim(hour,minute,second,hsecond)
      write(atime,'(I2.2,A,I2.2,A,I2.2)',iostat=ioerr)
     $   hour, ':', minute, ':',
     $         second

      if (ioerr .ne. 0) then
          for_rc = 0
          call ioretint(ioerr)
          call closefiles(filefn,10)
          return
      endif

      conunt=0
      confil=' '

      status_msg(1:80) = "Reading Input Filenames..."C
      call CMessage(status_msg,c_rc)
      if (c_rc .eq. 0) then
       return
      endif

      filfil = "data\renfnam"
      chainfil = "data\chain.dat"
      cedefil = "data\cede.dat"
      occfil = "data\ren.dat"

c     assign the logical file units
      filefn  = 50
      infn    = filefn + 1
      outfn   = filefn + 2
      hisfn   = filefn + 3
      chainfn = filefn + 4
      cedefn  = filefn + 5

c
c...read the input file number and name
      OPEN(filefn,FILE=filfil,STATUS='OLD',mode='read',
     $     SHARE='DENYNONE',iostat=ioerr)

      if (ioerr .ne. 0) then
          for_rc = 0
          call ioreturn(ioerr)
          call closefiles(filefn,10)
```

```
          return
      endif

      read(filefn,*,iostat=ioerr) infil,outfil,hisfil

      if (ioerr .ne. 0) then
          for_rc = 0
          call ioreturn(ioerr)
          call closefiles(filefn,10)
          return
      endif

      CLOSE(filefn,iostat=ioerr)
      if (ioerr .ne. 0) then
          for_rc = 0
          call ioreturn(ioerr)
          call closefiles(filefn,10)
          return
      endif


c...open output file
      open(outfn,file=outfil,status='unknown',mode='write',
     $     SHARE='DENYNONE',iostat=ioerr)

      if (ioerr .ne. 0) then
          for_rc = 0
          call ioreturn(ioerr)
          call closefiles(filefn,10)
          return
      endif
c

      status_msg(1:80) = "Reading User Input..."C
      call CMessage(status_msg,c_rc)
      if (c_rc .eq. 0) then
         call closefiles(filefn,10)
         return
      endif

c...read the user specified input data
      open(filefn,file=infil,status='old',mode='read',
     $     SHARE='DENYNONE',iostat=ioerr)

      if (ioerr .ne. 0) then
          for_rc = 0
          call ioreturn(ioerr)
          call closefiles(filefn,10)
          return
      endif

         status_msg(1:80)= "Calling input..."C
         call CMessage(status_msg,c_rc)

      ioerr=0
      call input(filefn,mxchn,title,histry,combin,import,distrb,
     $   nchn,chain,conc,ioerr,for_rc,nchnp2)

      if (ioerr .gt. 0) then
```

```
          for_rc = 0
          call closefiles(filefn,10)
          return
      endif

      if (for_rc .eq. 0) then
         status_msg(1:80)= "Error or Abort In Calculational Modules..."C
         call CMessage(status_msg,c_rc)
         call closefiles(filefn,10)
         return
      endif

      close (filefn,iostat=ioerr)

      if (ioerr .ne. 0) then
          for_rc = 0
          call ioreturn(ioerr)
          call closefiles(filefn,10)
          return
      endif
c
c...read the renovation scenario default parameters
      open(filefn,file=occfil,status='old',mode='read',
     $     SHARE='DENYNONE',iostat=ioerr)

      if (ioerr .ne. 0) then
          for_rc = 0
          call ioreturn(ioerr)
          call closefiles(filefn,10)
          return
      endif

      ioerr=0
      call rendef(filefn,tb,ttb,vb,cdb,gb,tstart,tend,dt,skip,ioerr)

      if (ioerr .ne. 0) then
          for_rc = 0
          call closefiles(filefn,10)
          return
      endif

      close (filefn,iostat=ioerr)

      if (ioerr .ne. 0) then
          for_rc = 0
          call ioreturn(ioerr)
          call closefiles(filefn,10)
          return
      endif
c
c...write the user specified input data

      ioerr=0
      call renout(outfn,mxchn,adate,atime,short,long,title,
     $   histry,combin,import,distrb,confil,
     $   tb,ttb,vb,cdb,gb,
     $   tstart,tend,dt,skip,
     $   nchn,chain,conc,ioerr,nchnp2)
```

```
      if (ioerr .ne. 0) then
          for_rc = 0
          call closefiles(filefn,10)
          return
      endif

c
c...check to see if any chains are the 'natural' chains
c   U_Nat or Th_Nat
      call naturl(mxchn,nchn,chain,conc)
c
c...if +C (equilibrium) designation exists on chain, then strip
c   +C designation from chain and set equilibrium flag
      call equil(mxchn,nchn,chain,echain)


      status_msg(1:80) = "Reading Data for Chain(s)..."C
      call CMessage(status_msg,c_rc)
      if (c_rc .eq. 0) then
        call closefiles(filefn,10)
        return
      endif


c
c...read chain data and cede data
      open(chainfn,file=chainfil,status='old',blocksize=39424,
     $    SHARE='DENYNONE',mode='read',iostat=ioerr)

      if (ioerr .ne. 0) then
          for_rc = 0
          call ioreturn(ioerr)
          call closefiles(filefn,10)
          return
      endif

      open(cedefn,file=cedefil,status='old',blocksize=15360,
     $    SHARE='DENYNONE',mode='read',iostat=ioerr)

      if (ioerr .ne. 0) then
          for_rc = 0
          call ioreturn(ioerr)
          call closefiles(filefn,10)
          return
      endif

      for_rc = 1

      do 10, ichn=1,nchn
c......read the chain data
         call rdchn(chainfn,outfn,for_rc,mxpar,mxpro,chain(ichn),
     $      nr(ichn),nuclid(1,0,ichn),hlif(1,ichn),lambda(1,ichn),
     $      ipos(1,ichn),ipar(1,1,ichn),frac(1,1,ichn),ioerr)

          if (ioerr .gt. 0) then
              for_rc = 0
              call closefiles(filefn,10)
              return
          endif
```

```
          if (for_rc .eq. 0) then
              exit
          endif


c......read the cede data
          call rdcede(cedefn,outfn,for_rc,mxpar,mxpro,combin,nr(ichn),
     $      nip(1,ichn),nipdis(1,ichn),nuclid(1,0,ichn),
     $      prfrac(1,0,ichn),dfer(1,0,ichn),dfes(1,0,ichn),
     $      dfg(1,0,ichn),dfh(1,0,ichn),ioerr)

          if (ioerr .gt. 0) then
              for_rc = 0
              call closefiles(filefn,10)
              return
          endif

         if (for_rc .eq. 0) then
            exit
         endif

   10 continue

      close(chainfn,iostat=ioerr)

      if (ioerr .ne. 0) then
          for_rc = 0
          call ioreturn(ioerr)
          call closefiles(filefn,10)
          return
      endif

      close(cedefn,iostat=ioerr)

      if (ioerr .ne. 0) then
          for_rc = 0
          call ioreturn(ioerr)
          call closefiles(filefn,10)
          return
      endif


      if (for_rc .eq. 0) then
         status_msg(1:80) = "Error In Calculational Modules..."C
         call CMessage(status_msg,c_rc)
         call closefiles(filefn,10)
         return
      endif


      status_msg(1:80) = "Starting Calculations..."C
      call CMessage(status_msg,c_rc)
      if (c_rc .eq. 0) then
         call closefiles(filefn,10)
         return
      endif

c
c...perform the maximum dose and history calculations
```

```
      if (histry) then
      open(hisfn,file=hisfil,status='unknown',mode='write',
     $     SHARE='DENYNONE',iostat=ioerr)

          if (ioerr .ne. 0) then
              for_rc = 0
              call ioreturn(ioerr)
              call closefiles(filefn,10)
              return
          endif
      endif

      call renmax(outfn,for_rc,mxchn,mxlist,mxpar,mxpath,mxpro,npath,
     $   hisfn,histry,conunt,import,distrb,
     $   tb,ttb,vb,cdb,gb,
     $   tstart,tend,dt,skip,
     $   nchn,chain,conc,
     $   nuclid,nr,nip,nipdis,hlif,lambda,
     $   ipos,ipar,frac,prfrac,
     $   dfer,dfes,dfg,dfh,
     $   nlist,
     $   ttede,period,printed,persv,
     $   list,sbarsv,dossv,nucsv,pathsv,tedemx,
     $   dosmx,tdos,tpdos,pathmx,tpath,tppath,nucmx,tnuc,tpnuc,
     $   c,cup,con,dos,dosup,echain,d,indx,nuc,nucup,
     $   path,pathup,sbar,sbarml,sbarup,ioerr)

      if (ioerr .ne. 0) then
          for_rc = 0
          call closefiles(filefn,10)
          return
      endif

      close (hisfn,iostat=ioerr)

      if (ioerr .ne. 0) then
          for_rc = 0
          call ioreturn(ioerr)
          call closefiles(filefn,10)
          return
      endif

      if (for_rc .eq. 0) then
         status_msg(1:80)= "Error or Abort In Calculational Modules..."C
         call CMessage(status_msg,c_rc)
         call closefiles(filefn,10)
         return
      endif

      status_msg(1:80) = "Writing Calculational Modules Results..."C
      call CMessage(status_msg,c_rc)
      if (c_rc .eq. 0) then
         call closefiles(filefn,10)
         return
      endif


c
c...write out maximum results
```

```
      call wrout(outfn,mxpath,mxlist,npath,nlist,short,
     $   ttede,ttb,period,printed,persv,
     $   list,sbarsv,dossv,nucsv,pathsv,tedemx,
     $   dosmx,tdos,tpdos,pathmx,tpath,tppath,nucmx,tnuc,tpnuc,ioerr)

      if (ioerr .ne. 0) then
          for_rc = 0
          call closefiles(filefn,10)
          return
      endif

c
      close (outfn,iostat=ioerr)

      if (ioerr .ne. 0) then
          for_rc = 0
          call ioreturn(ioerr)
          call closefiles(filefn,10)
          return
      endif

      for_rc = 1

      status_msg(1:80) = "Completed Calculational Modules..."C
      call CMessage(status_msg,c_rc)
      call closefiles(filefn,10)

      end
```

F.79 RENMAX.FOR

```
c
c   This subroutine performs the maximum dose calculations
c   updates the history file for selected time steps
c
      include "cdecl.inc"

      subroutine renmax(out,rc,mxchn,mxlist,mxpar,mxpath,mxpro,npath,
     $   his,histry,conunt,import,distrb,
     $   tb,ttb,vb,cdb,gb,
     $   tstart,tend,dt,skip,
     $   nchn,chain,conc,
     $   nuclid,nr,nip,nipdis,hlif,lambda,
     $   ipos,ipar,frac,prfrac,
     $   dfer,dfes,dfg,dfh,
     $   nlist,
     $   ttede,period,printed,persv,
     $   list,sbarsv,dossv,nucsv,pathsv,tedemx,
     $   dosmx,tdos,tpdos,pathmx,tpath,tppath,nucmx,tnuc,tpnuc,
     $   c,cup,con,dos,dosup,echain,f,indx,nuc,nucup,
     $   path,pathup,sbar,sbarm1,sbarup,ioerr)

      character*6 list
      character*6 nuclid
      character*8 chain
      character*80 status_msg
      double precision c,cdb,con,conc,cup
      double precision dfer,dfes,dfg,dfh
      double precision dos,dosmx,dossv,dosup
      double precision dt
      double precision f,frac
      double precision gb
      double precision hlif
      double precision lambda
      double precision nuc,nucmx,nucsv,nucup
      double precision path,pathmx,pathup,pathsv,prfrac
      double precision sbar,sbarm1,sbarsv,sbarup
      double precision tb,ttb,tcalc,tdos,tede,tedemx,tedeup,tend
      double precision tnuc,tpath,tstart,tstop,ttede
      double precision vb
      integer conunt
      integer his
      integer ichn,indx,ioerr,ipar,ipos
      integer mxchn,mxlist,mxpar,mxpath,mxpro
      integer nchn,nip,nipdis,nlist,npath,nr
      integer out,rc
      integer period,printed,persv,tppath,tpdos,tpnuc
      integer skip
      integer tsteps,achk
      logical done
      logical echain
      logical histry
      logical import
      logical report
      logical dcreas,distrb,iszero
      dimension c(mxpar,0:mxpro)
      dimension chain(mxchn)
      dimension con(mxpar,mxchn)
```

```
      dimension conc(mxchn)
      dimension cup(mxlist)
      dimension dfer(mxpar,0:mxpro,mxchn),dfes(mxpar,0:mxpro,mxchn)
      dimension dfg(mxpar,0:mxpro,mxchn),dfh(mxpar,0:mxpro,mxchn)
      dimension dos(mxpar,0:mxpro,mxpath)
      dimension dosmx(mxlist,mxpath)
      dimension dossv(mxlist,mxpath),dosup(mxlist,mxpath)
      dimension echain(mxchn)
      dimension f(mxpar,mxpar,mxchn)
      dimension frac(2,mxpar,mxchn)
      dimension hlif(mxpar,mxchn)
      dimension indx(mxpar,0:mxpro,mxchn)
      dimension ipar(2,mxpar,mxchn)
      dimension ipos(mxpar,mxchn)
      dimension lambda(mxpar,mxchn)
      dimension list(mxlist)
      dimension nip(mxpar,mxchn),nipdis(mxpar,mxchn)
      dimension nr(mxchn)
      dimension nuc(mxpar,0:mxpro)
      dimension nuclid(mxpar,0:mxpro,mxchn)
      dimension nucmx(mxlist),nucsv(mxlist),nucup(mxlist)
      dimension path(mxpath),pathmx(mxpath)
      dimension pathsv(mxpath),pathup(mxpath)
      dimension prfrac(mxpar,0:mxpro,mxchn)
      dimension sbar(mxpar,0:mxpro)
      dimension sbarm1(mxlist),sbarsv(mxlist),sbarup(mxlist)
      dimension tdos(mxlist,mxpath)
      dimension tnuc(mxlist)
      dimension tpath(mxpath)
      dimension tppath(mxpath)
      dimension tpdos(mxlist,mxpath)
      dimension tpnuc(mxlist)

      printed = 0
      tsteps = ((tend - tstart)/dt)
      if ((dt .eq. 0) .OR. ((tend - tstart) .le. 100)) then
          achk = 10
      else
          achk = tsteps/10
      endif

      if (achk .eq. 0) achk = 10
c
c...perform preliminary calculations
      do 10, ichn=1,nchn

c......set up the initial activities for each radionuclide in the chain
          call concen(mxpar,mxpro,nr(ichn),echain(ichn),distrb,
     $        lambda(1,ichn),ipar(1,1,ichn),frac(1,1,ichn),
     $        prfrac(1,0,ichn),nipdis(1,ichn),conc(ichn),con(1,ichn))

c......write chain, cede and activity data to output file
          call wrchn(out,mxpar,mxpro,nr(ichn),nip(1,ichn),ichn,
     $        chain(ichn),echain(ichn),nuclid(1,0,ichn),ipos(1,ichn),
     $        ipar(1,1,ichn),frac(1,1,ichn),hlif(1,ichn),prfrac(1,0,ichn),
     $        con(1,ichn),dfg(1,0,ichn),dfh(1,0,ichn),dfes(1,0,ichn),
     $        dfer(1,0,ichn),'RENOVA',ioerr)

          if (ioerr .ne. 0) then
```

```
            return
          endif

c......convert dfer, dfes, dfg, and dfh from Bq-Sv system to
c   mrem-H-pCi system
         call convrt(mxpar,mxpro,nr(ichn),nip(1,ichn),
     $       dfg(1,0,ichn),dfh(1,0,ichn),
     $       dfes(1,0,ichn),dfer(1,0,ichn))

c......set up the fractional yield array
         call frctnl(mxpar,nr(ichn),f(1,1,ichn),
     $       frac(1,1,ichn),ipar(1,1,ichn))

   10 continue
c
c...combine radionuclides from all chains into a unique list of
c   radionuclides and tie each radionuclide from each chain into
c   an index number in the list
      call nuclis(mxlist,mxchn,mxpar,mxpro,
     $   nlist,nchn,nr,nip,nuclid,list,indx)

      status_msg = "Calculating up to                    "
      write(status_msg(19:23),'(i4)',iostat=ioerr) tsteps

      if (ioerr .ne. 0) then
          call ioretint(ioerr)
          return
      endif

      status_msg = status_msg(1:24)//' timesteps...'C
      call CMessage(status_msg,rc)
c
c...calculate doses

c......initialize some time parameters
      call init0(out,rc,conunt,import,ttb,tstart,
     $    tend,period,tcalc,tstop,ioerr)

          if (ioerr .ne. 0) then
            return
          endif

          if (rc .eq. 0) then
              return
          endif

c......initialize maximum values of comparison variables
      call initzd(mxlist*mxpath,dosmx)
      call initzd(mxlist,nucmx)
      call initzd(mxpath,pathmx)
      call initzd(1,tedemx)
      call initzd(mxlist,sbarml)
      call initlg(1,.false.,dcreas)

   15 continue

c......determine if current calculation time is a report time
      call rpttim(tcalc,tstop,period,skip,histry,report)
```

```
c......calculate dose for each radionuclide in each chain
c       and for each pathway
c......initialize update variables for the current time step
      call initzd(mxlist*mxpath,dosup)
      call initzd(mxlist,nucup)
      call initzd(mxpath,pathup)
      call initzd(1,tedeup)
      call initzd(mxlist,cup)
      call initzd(mxlist,sbarup)

      do 20, ichn=1,nchn
c.........calculate dose in chain for each radionuclide
c          for each pathway

         call rendos(mxpar,mxpro,nr(ichn),nip(1,ichn),
     $      tcalc,dfer(1,0,ichn),dfh(1,0,ichn),dfg(1,0,ichn),
     $      tb,ttb,vb,cdb,gb,
     $      con(1,ichn),lambda(1,ichn),f(1,1,ichn),prfrac(1,0,ichn),
     $      c,sbar,dos(1,0,1),dos(1,0,2),dos(1,0,3))
c
c.........calculate pathway dose for chain
         call pthsum(mxpar,mxpro,mxpath,nr(ichn),nip(1,ichn),npath,
     $       dos,path)
c
c.........calculate radionuclide dose for chain
         call nucsum(mxpar,mxpro,mxpath,nr(ichn),nip(1,ichn),npath,
     $       dos,nuc)
c
c.........calculate tede for chain
         call tedsum(mxpar,mxpro,nr(ichn),nip(1,ichn),nuc,tede)
c
c.........update listed radionuclide activity
         call upsbar(mxpar,mxpro,mxlist,nr(ichn),nip(1,ichn),
     $      indx(1,0,ichn),c,cup)
c
c.........update listed radionuclide integrated activity
         call upsbar(mxpar,mxpro,mxlist,nr(ichn),nip(1,ichn),
     $      indx(1,0,ichn),sbar,sbarup)
c
c.........update listed radionuclide dose for each pathway for all chains
         call updose(mxpar,mxpro,mxpath,mxlist,
     $      nr(ichn),nip(1,ichn),npath,
     $      indx(1,0,ichn),dos,dosup)
c
c.........update pathway dose for all chains
         call uppath(mxpath,npath,path,pathup)
c
c.........update listed radionuclide dose for all chains
         call upnuc(mxpar,mxpro,mxlist,nr(ichn),nip(1,ichn),
     $      indx(1,0,ichn),nuc,nucup)
c
c.........update total dose for all chains
         call uptede(tede,tedeup)
   20 continue
c
c...if this is a report time write results to history file
      if (report) then
         call wrhist(his,mxlist,mxpath,nlist,npath,
     $    tcalc,tcalc+ttb,list,cup,sbarup,dosup,pathup,nucup,tedeup,
```

```
     $    ioerr)

          if (ioerr .ne. 0) then
            return
          endif

         printed = printed + 1

      endif

c
c......if integrated activities are increasing, then test for
c      maximum values of total dose (tede) and find its components,
c      pathway dose, listed radionuclide dose, and listed
c      radionuclide dose by pathway

      if (.not. dcreas) then

c......check if total tede is maximum so far; if so, save it,
c      its components, and its time of occurrence
         call tedmax(mxlist,mxpath,nlist,npath,tcalc,period,persv,
     $       ttede,tedeup,tedemx,pathsv,pathup,nucsv,nucup,dossv,dosup,
     $       sbarsv,sbarup)

c......check for maximum pathway doses; if so, save them and their
c      time of occurrence
         call pthmax(mxpath,npath,tcalc,period,tpath,tppath,
     $       pathup,pathmx)

c......check for maximum listed radionuclide doses; if so, save
c      them and their time of occurrence
         call nucmax(mxlist,nlist,tcalc,period,tnuc,tpnuc,nucup,nucmx)

c......check for maximum listed radionuclide doses per pathway; if so,
c      save them and their time of occurence
         call dosmax(mxpath,mxlist,npath,nlist,tcalc,period,dosup,
     $    tdos,tpdos,dosmx)

      endif


c......determine if the integrated activities are decreasing. if they
c      are all decreasing, set a test value to true and check history
c      value to determine if maximum results should be written out at
c      this time

      iszero = .true.
      call acttst(mxlist,nlist,sbarup,sbarm1,dcreas,iszero)

      if (dcreas .and. .not. histry) then
          status_msg = "All doses are decreasing..."C
          call CMessage(status_msg,rc)
          goto 25
      endif

      if (dcreas .and. iszero) then
          status_msg = "All doses are decreasing and under minimum..."C
          call CMessage(status_msg,rc)
          goto 25
```

```
      endif

      if (tcalc .ge. tstop) goto 25
c......update time
      call uptime(conunt,import,done,period,tcalc,dt,tstop,ioerr)

      if (ioerr .ne. 0) then
          return
      endif

      if (mod(period, achk) .eq. 1) then

          write(status_msg(1:10),'(i10)',iostat=ioerr) period

          if (ioerr .ne. 0) then
              call ioretint(ioerr)
              return
          endif

          status_msg = status_msg(1:10)//'  timestep(s) out of 'C

          write(status_msg(32:36),'(i4)',iostat=ioerr) tsteps

          if (ioerr .ne. 0) then
              call ioretint(ioerr)
              return
          endif

          status_msg = status_msg(1:36)//' calculated...'C

          call CMessage(status_msg,rc)
          if (rc .eq. 0) then
              return
          endif
      else
          call CProcess(rc)
          if (rc .eq. 0) then
              return
          endif
      endif

      goto 15
   25 continue

      write(status_msg(1:10),'(i10)',iostat=ioerr) period

      if (ioerr .ne. 0) then
          call ioretint(ioerr)
          return
      endif

      status_msg = status_msg(1:10)//
     $    ' timesteps were actually calculated'C

      call CMessage(status_msg,rc)

      end
```

F.80 RENOUT.FOR

```
c
c...this subroutine writes out the user specified input data
c   for the building renovation scenario
c

      subroutine renout(out,mxchn,adate,atime,short,long,title,
     $   histry,combin,import,distrb,confil,
     $   tb,ttb,vb,cdb,gb,
     $   tstart,tend,dt,skip,
     $   nchn,chain,conc,ioerr,nchnp2)

      character*6 short
      character*10 adate
      character*8 chain
      character*11 atime
      character*28 long
      character*80 title,confil
      integer ichn,ioerr
      integer loc,loc1,loc2
      integer mxchn
      integer nchn,nchnp2
      integer out
      integer skip
      logical combin
      logical histry
      logical distrb
      logical equil
      logical import
      double precision cdb,conc
      double precision dt
      double precision gb
      double precision tb,ttb,tend,tstart
      double precision vb
      dimension chain(mxchn)
      dimension conc(mxchn)
c
      write(out,'(3a)',iostat=ioerr) 'Output From Program ''',short,''''
      if (ioerr .ne. 0) then
        call ioreturn(ioerr)
        return
      endif

      write(out,'(a)',iostat=ioerr) long
      if (ioerr .ne. 0) then
        call ioreturn(ioerr)
        return
      endif

      write(out,'(2a)',iostat=ioerr) 'Run Date: ',adate
      if (ioerr .ne. 0) then
        call ioreturn(ioerr)
        return
      endif

      write(out,'(2a)',iostat=ioerr) 'Run Time: ',atime
      if (ioerr .ne. 0) then
        call ioreturn(ioerr)
```

```
  return
endif

write(out,'(a)',iostat=ioerr) 'INPUT DATA:'
if (ioerr .ne. 0) then
  call ioreturn(ioerr)
  return
endif

write(out,'(2a)',iostat=ioerr) 'Title: ',title
if (ioerr .ne. 0) then
  call ioreturn(ioerr)
  return
endif

write(out,'(a)',iostat=ioerr) 'Notes: '
if (ioerr .ne. 0) then
  call ioreturn(ioerr)
  return
endif

if (histry) then
   write(out,'(a)',iostat=ioerr) 'History file will be generated.'
    if (ioerr .ne. 0) then
        call ioreturn(ioerr)
        return
    endif
else
   write(out,'(a)',iostat=ioerr)
$      'History file will not be generated.'
    if (ioerr .ne. 0) then
        call ioreturn(ioerr)
        return
    endif
endif

if (combin) then
   write(out,'(2a)',iostat=ioerr) 'Implicit progeny doses will',
$      ' be included with explicit parent.'
    if (ioerr .ne. 0) then
        call ioreturn(ioerr)
        return
    endif
else
   write(out,'(2a)',iostat=ioerr)
$      'Implicit progeny doses will not',
$      ' be included with explicit parent.'
    if (ioerr .ne. 0) then
        call ioreturn(ioerr)
        return
    endif
endif

if (import) then
   write(out,'(4a)',iostat=ioerr)
$      'Concentration data will be read ',
$      'from file: ',confil,'.'
    if (ioerr .ne. 0) then
        call ioreturn(ioerr)
```

```
            return
         endif
      else
         write(out,'(a)',iostat=ioerr)
     $      'Concentration data will be calculated.'
         if (ioerr .ne. 0) then
            call ioreturn(ioerr)
            return
         endif

         equil=.false.

         do 10, ichn=1,nchn
            if (chain(ichn) .eq. 'U_Nat') then
               loc=0
            elseif(chain(ichn) .eq. 'Th_Nat') then
               loc=6
            else
               loc1=max(index(chain(ichn),'+C'),index(chain(ichn),'+c'))
               loc2=max(index(chain(ichn),'_C'),index(chain(ichn),'_c'))
               loc=max(loc1,loc2)
            endif
            if (loc .gt. 0) equil=.true.
10       continue

         if (equil) then
            write(out,'(2a)',iostat=ioerr)
     $         'Concentrations of radionuclides in ',
     $         'equilibrium chains will be'
            if (ioerr .ne. 0) then
               call ioreturn(ioerr)
               return
            endif

            if (distrb) then

               write(out,'(2a)',iostat=ioerr)
     $            '   distributed based on initial ',
     $            'concentration given for parent.'
               if (ioerr .ne. 0) then
                  call ioreturn(ioerr)
                  return
               endif

            else

               write(out,'(2a)',iostat=ioerr)
     $            '   equilibrated based on initial ',
     $            'concentration of parent.'
               if (ioerr .ne. 0) then
                  call ioreturn(ioerr)
                  return
               endif
            endif
         endif
      endif

      write(out,'(a)',iostat=ioerr) ' '
      if (ioerr .ne. 0) then
```

```
      call ioreturn(ioerr)
      return
    endif

    write(out,'(a)',iostat=ioerr) 'PARAMETER DATA:'
    if (ioerr .ne. 0) then
      call ioreturn(ioerr)
      return
    endif

    write(out,'(a,1p,e10.4,a)',iostat=ioerr)
$     'Time on Job              : ',tb,' d/y'
    if (ioerr .ne. 0) then
      call ioreturn(ioerr)
      return
    endif

    write(out,'(a,1p,e10.4,a)',iostat=ioerr)
$     'Renovation Period        : ',ttb,' d'
    if (ioerr .ne. 0) then
      call ioreturn(ioerr)
      return
    endif

    write(out,'(a,1p,e10.4,a)',iostat=ioerr)
$     'Breathing Rate           : ',vb,
$     ' m**3/h'
    if (ioerr .ne. 0) then
      call ioreturn(ioerr)
      return
    endif

    write(out,'(a,1p,e10.4,a)',iostat=ioerr)
$     'Air Dust Loading         : ',cdb,
$     ' g/m**3'
    if (ioerr .ne. 0) then
      call ioreturn(ioerr)
      return
    endif

    write(out,'(a,1p,e10.4,a)',iostat=ioerr)
$     'Loose Dust Transfer Rate: ',gb,' g/h'
    if (ioerr .ne. 0) then
      call ioreturn(ioerr)
      return
    endif

    write(out,'(a)',iostat=ioerr) 'TIME DATA:'
    if (ioerr .ne. 0) then
      call ioreturn(ioerr)
      return
    endif

    write(out,'(a,1p,e10.4,a)',iostat=ioerr)
$     'Start Time               : ',tstart,
$     ' d'
    if (ioerr .ne. 0) then
      call ioreturn(ioerr)
      return
```

```
 endif

 write(out,'(a,1p,e10.4,a)',iostat=ioerr)
$   'End Time                  : ',tend,' d'
 if (ioerr .ne. 0) then
   call ioreturn(ioerr)
   return
 endif

 write(out,'(a,1p,e10.4,a)',iostat=ioerr)
$   'DT Size                   : ',dt,' d'
 if (ioerr .ne. 0) then
   call ioreturn(ioerr)
   return
 endif

 write(out,'(a,1p,e10.4,a)',iostat=ioerr)
$   'Time Step Size            : ',ttb,' d'
 if (ioerr .ne. 0) then
   call ioreturn(ioerr)
   return
 endif

 write(out,'(a,i5,a)',iostat=ioerr)
$   'Write results every       : ',skip,
$   ' calculation times'
 if (ioerr .ne. 0) then
   call ioreturn(ioerr)
   return
 endif

 write(out,'(a)',iostat=ioerr) ' '
 if (ioerr .ne. 0) then
   call ioreturn(ioerr)
   return
 endif

 write(out,'(a)',iostat=ioerr) 'INITIAL ACTIVITIES:'
 if (ioerr .ne. 0) then
   call ioreturn(ioerr)
   return
 endif

 write(out,'(/a,i3)',iostat=ioerr) 'Number of chains: ',nchnp2
 if (ioerr .ne. 0) then
   call ioreturn(ioerr)
   return
 endif

 write(out,'(/a)',iostat=ioerr) ' Chain       Chain      Initial'
 if (ioerr .ne. 0) then
   call ioreturn(ioerr)
   return
 endif

 write(out,'(a)',iostat=ioerr) 'Number        Name     Activity'
 if (ioerr .ne. 0) then
   call ioreturn(ioerr)
   return
```

```
      endif

      write(out,'(a)',iostat=ioerr) '                                    (pCi/g)'
      if (ioerr .ne. 0) then
        call ioreturn(ioerr)
        return
      endif

      write(out,'(i6,2x,a,2x,1p,e10.4)',iostat=ioerr)
     $   (ichn,chain(ichn),conc(ichn),ichn=1,nchn)
      if (ioerr .ne. 0) then
        call ioreturn(ioerr)
        return
      endif
      end
```

F.81 REOPER.FOR

```
c*********************************************************************
c
c  subroutine reoper;  5/93
c  C. David Updegraff, GRAM, Inc., Albuquerque, NM 87112.
c  Solves the 're' operator for deposition/accumulation over a time
c  period using equation 2.5, equation B.15 and Table 2.4.
c*********************************************************************
c
      subroutine reoper(c,re,lambdr,lambdw,d,t,nr)

      double precision c
      double precision d,dlmexp
      double precision k
      double precision lambde,lambdr,lambdw
      double precision one
      double precision re
      double precision t
      double precision zero
      integer j
      integer maxr
      integer n,nr
      parameter (maxr=20)
      parameter (one=1.0d+0,zero=0.0d+0)
      dimension c(maxr)
      dimension d(maxr,maxr)
      dimension k(maxr,maxr)
      dimension lambdr(maxr)
      dimension re(maxr)
c
      call kcoef(c,lambdr,d,k,nr,maxr)
c
      do 20, j=1,nr
         re(j)=zero
            do 10, n=1,j
               lambde=lambdr(n)+lambdw
               re(j)=re(j)+k(j,n)*dlmexp(lambde*t)/lambde
   10       continue
CCC...added a check for small numbers that turn negative;
CCC...zero those numbers out
CCC            re(j)=lambdr(j)*re(j)
            re(j)=max(zero,lambdr(j)*re(j))
   20 continue
      end
```

F.82 RESDEF.FOR

```
c
c...This subroutine reads the residential scenario default
c   values specified by Table 6.23 with corrections
c

      subroutine resdef(in,ar,cdg,cdi,cdo,diet,f1,f2,gr,h1,h2,nlay,
     $   i,ir,lambdw,mlf,mlg,mlh,mlv,n1,n2,pd,ps,qd,qf,qg,qh,qw,
     $   rf,rfr,rg,rh,rho1,rho2,rv,sfi,sfo,tstart,tend,dt,skip,
     $   tbigf,tbigg,tbigh,tbigv,
     $   td,tca,tcv,tf,tff,tfg,tfh,tfw,tg,tgf,tgg,tgv,tgh,tha,
     $   thv,ti,ttg,ttr,tx,ua,uf,uv,uw,vdr,vg,vr,vsw,vx,
     $   wf,wg,wh,wv,xf,xg,xh,xw,yf,yg,yh,yv,fca,fcd,fcf,fcg,fch,satac,
     $   fha,fhd,fhf,fhg,fhh,fhv,
     $   sasvh,satah,sawvh,sh,ioerr)

      double precision ar
      double precision cdg,cdi,cdo
      double precision diet
      double precision f1,f2
      double precision gr
      double precision h1,h2
      double precision i,ir
      double precision lambdw
      double precision mlf,mlg,mlh,mlv
      double precision n1,n2
      double precision pd,ps
      double precision qd,qf,qg,qh,qw
      double precision rf,rfr,rg,rh,rho1,rho2,rv
      double precision sfi,sfo
      double precision tbigf,tbigg,tbigh,tbigv
      double precision tca,tcv,td
      double precision tf,tff,tfg,tfh,tfw,tg,tgf,tgg,tgv,tgh,tha
      double precision thv,ti,ttg,ttr,tx
      double precision ua,uf,uv,uw
      double precision vdr,vg,vr,vsw,vx
      double precision wf,wg,wh,wv
      double precision xf,xg,xh,xw
      double precision yf,yg,yh,yv
      double precision dt
      double precision tend,tstart,tskip,tnlay
      double precision fca,fcd,fcf,fcg,fch
      double precision satac
      double precision fha,fhd,fhf,fhg,fhh,fhv
      double precision sasvh,satah,sawvh,sh
      integer in,ioerr,skip
      integer j,nlay
      dimension mlf(4),mlg(4),mlh(4),mlv(4)
      dimension qd(4),qf(4),qg(4),qh(4),qw(4)
      dimension rf(4),rg(4),rh(4),rv(4)
      dimension tbigf(4),tbigg(4),tbigh(4),tbigv(4),tca(4),tcv(4)
      dimension tff(4),tfg(4),tfh(4),tfw(4),tgf(4),tgg(4),tgh(4),tgv(4)
      dimension tha(4),thv(4)
      dimension ua(4),uv(4)
      dimension wf(4),wg(4),wh(4),wv(4)
      dimension xf(4),xg(4),xh(4),xw(4)
      dimension yf(4),yg(4),yh(4),yv(4)
      dimension fca(4),fcf(4),fcg(4),fch(4)
```

```
      dimension fha(4),fhf(4),fhg(4),fhh(4),fhv(4)
c
c...read the default data
      read(in,*,iostat=ioerr)
     $    ti,tx,tg,ttr,ttg,lambdw,sfi,sfo,pd,rfr,cdi,cdo,cdg,vr,
     $    vx,vg,gr,uw,td,h1,h2,tnlay,n1,n2,f1,f2,rho1,rho2,vdr,vsw,i,ar,
     $    ir,tstart,tend,dt,tskip,diet,(uv(j),j=1,4),(ua(j),j=1,4),
     $    uf,tf,(tcv(j),j=1,4),(tca(j),j=1,4),(thv(j),j=1,4),
     $    (tha(j),j=1,4),(tgv(j),j=1,4),(tgf(j),j=1,4),(tgg(j),j=1,4),
     $    (tgh(j),j=1,4),(tff(j),j=1,4),(tfg(j),j=1,4),(tfh(j),j=1,4),
     $    (rv(j),j=1,4),(rf(j),j=1,4),(rg(j),j=1,4),(rh(j),j=1,4),
     $    (tbigv(j),j=1,4),(tbigf(j),j=1,4),(tbigg(j),j=1,4),
     $    (tbigh(j),j=1,4),(xf(j),j=1,4),(xg(j),j=1,4),(xh(j),j=1,4),
     $    (xw(j),j=1,4),(yv(j),j=1,4),(yf(j),j=1,4),(yg(j),j=1,4),
     $    (yh(j),j=1,4),(wv(j),j=1,4),(wf(j),j=1,4),(wg(j),j=1,4),
     $    (wh(j),j=1,4),(qf(j),j=1,4),(qg(j),j=1,4),(qh(j),j=1,4),
     $    (qw(j),j=1,4),(tfw(j),j=1,4),(qd(j),j=1,4),(mlv(j),j=1,4),
     $    (mlf(j),j=1,4),(mlg(j),j=1,4),(mlh(j),j=1,4),(fca(j),j=1,4),
     $    (fcf(j),j=1,4),(fcg(j),j=1,4),
     $    (fch(j),j=1,4),fcd,satac,(fhv(j),j=1,4),(fha(j),j=1,4),
     $    (fhf(j),j=1,4),
     $    (fhg(j),j=1,4),(fhh(j),j=1,4),fhd,satah,sasvh,sawvh,sh

      if (ioerr .ne. 0) call ioreturn(ioerr)

      skip = tskip
      nlay = tnlay

c     Compute the areal soil density
      ps = rho1 * h1 * 1000.0

      end
```

F.83 RESDOS.FOR

```
c
c...this subroutine calculates the doses for each radionuclide
c   in a chain for each pathway for the building renovation scenario
c
      subroutine resdos(rc,mxnr,mxnip,ar,baf,cdg,cdi,cdo,csoili,con,
     $   concen,infil0,atol,ar1old,d,ddd,dfer,dfg,dfh,
     $   diet,dt,gr,h1,ia,ja,kcoef,lambda,nbox,neq,nip,nlay,
     $   nnz,nr,pd,period,pfs,pfw,prfrac,rfr,rhol,root,rtol,sfi,sfo,
     $   sr3new,t,td,tf,tg,ti,ttg,ttr,tx,uf,uw,vg,vr,vsw,vtr,vx,
     $   ar1new,ast,awt,awtswp,dexr,dhr,dgr,dsr,dwr1,dwr2,dar,ioerr)

      double complex ar1
      double complex kcoef
      double complex root
      double complex sr3,sr3old,sr3new
      double precision atol,ar,ar1new,ar1old
      double precision ast,awt,awtswp
      double precision baf
      double precision cdel,cdg,cdi,cint,cdo,csoili
      double precision d,dar,ddd,dexr,dfer,dfg,dfh,dgr
      double precision dhr,diet,dsr,dt,dwr1,dwr2
      double precision factor
      double precision gr
      double precision h1
      double precision lambda
      double precision millon
      double precision one
      double precision pd,pfs,pfw,prfrac
      double precision qt
      double precision rfr,rhol,rsp,rtol
      double precision sasttg,sasttr,sfi,sfo
      double precision t,td,tf,tg,ti,tnew,told,ttg,ttr,tx
      double precision uf,uw
      double precision vg,vr,vsw,vtr,vx
      double precision zero
      integer c,con
      integer esp
      integer flag
      integer ia,ibox,ic,ieq,iip,ioerr,ir,isp,istate,inuc
      integer ja
      integer lratio
      integer maxbox,maxneq,maxnnz,maxnsp,mxnip,mxnr,maxpar
      integer nbox,neq,nip,nlay,nnz,nr,nsp
      integer period
      integer r,rc
      logical concen
      logical infil0
      parameter (lratio=2)
      parameter (maxbox=12,maxpar=20)
      parameter (maxneq=maxbox*maxpar)
      parameter (maxnnz=(maxbox*maxpar*(maxpar+3))/2)
      parameter (maxnsp=8*maxneq+2*maxnnz+2)
      parameter (millon=1.0d+6)
      parameter (one=1.0d+0)
      parameter (zero=0.0d+0)
      dimension ar1(3,maxpar)
      dimension ar1new(maxneq),ar1old(maxneq)
```

```
      dimension ast(mxnr,0:mxnip),awt(mxnr,0:mxnip)
      dimension awtswp(mxnr,0:mxnip)
      dimension baf(mxnr)
      dimension c(maxneq)
      dimension cdel(maxneq)
      dimension cint(maxneq)
      dimension csoili(mxnr)
      dimension d(mxnr,mxnr)
      dimension dar(mxnr,0:mxnip)
      dimension ddd(maxnnz)
      dimension dexr(mxnr,0:mxnip)
      dimension dfer(mxnr,0:mxnip),dfg(mxnr,0:mxnip)
      dimension dfh(mxnr,0:mxnip),dgr(mxnr,0:mxnip)
      dimension dhr(mxnr,0:mxnip),dsr(mxnr,0:mxnip)
      dimension dwr1(mxnr,0:mxnip),dwr2(mxnr,0:mxnip)
      dimension ia(neq+1)
      dimension ic(maxneq)
      dimension isp(lratio*maxnsp)
      dimension ja(nnz)
      dimension kcoef(3,3,mxnr,mxnr)
      dimension lambda(mxnr)
      dimension nip(mxnr)
      dimension pfs(mxnr,mxnr),pfw(mxnr,mxnr)
      dimension prfrac(mxnr,0:mxnip)
      dimension r(maxneq)
      dimension root(3,mxnr)
      dimension rsp(maxnsp)
      dimension sasttg(maxpar),sasttr(maxpar)
      dimension sr3(3,maxpar),sr3new(3,maxpar),sr3old(3,maxpar)
      equivalence (isp(1),rsp(1))
c
c......set a factor for converting aquifer concentration to
c      surface water pond concentrations. if no pond, set factor to
c      zero to imply no pond and zero surface water pond concentrations
      if (vsw .gt. zero .and. vtr .gt. zero) then
         factor=one/(one+vsw/vtr)
      else
         factor=zero
      end if
c
c...calculate factor to change total activity in ground-water
c      to activity concentration
      if (vtr .gt. zero) then
         qt=ar*h1*rho1*millon/(vtr*ttr)
      else
         qt=zero
      endif
c
      if (concen) then
c......read activity from concentration file
         do 10, ir=1,nr
            read(con,*,iostat=ioerr) ast(ir,0),awt(ir,0)

               if (ioerr .ne. 0) then
                  call ioreturn(ioerr)
                  return
               endif
   10    continue
      else
```

```
c
c...calculate activity concentrations
         if (infil0) then
c......0 infiltration case (soil concentrations only)
            call aoper(csoili,ast,lambda,d,t,nr)
            do 20, ir=1,nr
               awt(ir,0)=zero
   20       continue
c
c......single layer case (analytical solution)
         elseif (nlay .eq. 1) then
c......calculate the soil activity of each radionuclide at the
c      beginning of the time of interest
            call aroper(mxnr,mxnr,nr,kcoef,root,t,ar1)
c
c......calculate the ground-water integrated activity over
c      the period of interest
c......first calculate integrated activity at current time
c      and at end time of next dose period
            if (period .eq. 1 .or. dt .ne. ttr) then
               call sroper(mxnr,mxnr,nr,kcoef,root,t,sr3old)
               call sroper(mxnr,mxnr,nr,kcoef,root,t+ttr,sr3new)
c
            else
               do 40, ir=1,nr
                  do 30, ibox=1,nbox
                     sr3old(ibox,ir)=sr3new(ibox,ir)
   30             continue
   40          continue
               call sroper(mxnr,mxnr,nr,kcoef,root,t+ttr,sr3new)
            endif
c
c......subtract integrated activity from end of dose period from
c      integrated activity at current time to get integrated activity
c      for current dose period t to t+ttr
            do 60, ir=1,nr
               do 50, ibox=1,nbox
                  sr3(ibox,ir)=sr3new(ibox,ir)-sr3old(ibox,ir)
   50          continue
   60       continue
c
c......calculate and store soil, groundwater, and surface water
c......activities for each radionuclide
            do 70, ir=1,nr
               ast(ir,0)=dble(ar1(1,ir))
               awt(ir,0)=dble(sr3(3,ir))*qt
   70       continue
c
         else
c...perform numerical solution
c......calculate the average ground-water activity over
c      the period of interest from t to t+ttr
c......first calculate integrated activity at end of current dose time
c      period, ttr
            do 80, ieq=1,neq
               ar1new(ieq)=ar1old(ieq)
   80       continue

            istate=1
```

```
            told=t
            tnew=t+ttr

            call solvde(atol,ar1new,ia,istate,ja,neq,nnz,rtol,
     $         told,tnew)

            if (istate .lt. 0) then
               rc=0
               return
            endif
c
c......store soil box activities at current time, t
            inuc=0
c
            do 90, ieq=1,neq,nbox
               inuc=inuc+1
               ast(inuc,0)=ar1old(ieq)
   90       continue
c
c......solve for average ground-water concentrations
c.........calculate difference in ground-water activities from current
c         time, t, to end of dose time period, t+ttr
            do 100, ieq=1,neq
               cdel(ieq)=ar1new(ieq)-ar1old(ieq)
  100       continue

c           if (t .gt. 1000.0) stop
c......solve for integrated actvities with sparse matrix solver
            do 110, ieq=1,neq
               r(ieq)=ieq
               c(ieq)=ieq
               ic(ieq)=ieq
  110       continue

            nsp=8*neq+2+2*nnz

            call cdrv(neq,r,c,ic,ia,ja,ddd,cdel,cint,
     $         nsp,isp,rsp,esp,5,flag)

            if (esp .lt. 0) then
               nsp=nsp+abs(esp)
               call cdrv(neq,r,c,ic,ia,ja,ddd,cdel,cint,
     $            nsp,isp,rsp,esp,5,flag)
            endif

            call cdrv(neq,r,c,ic,ia,ja,ddd,cdel,cint,
     $         nsp,isp,rsp,esp,2,flag)

            call cdrv(neq,r,c,ic,ia,ja,ddd,cdel,cint,
     $         nsp,isp,rsp,esp,4,flag)
c
c......store average ground-water concentration
            do 120, ir=1,nr
               awt(ir,0)=cint((ir-1)*nbox+nbox)*qt
  120       continue
         endif
c
      endif
c
```

```
c...calculate activities for implicit progeny in surface soil,
c       ground-water, and surface water pond
      do 140, ir=1,nr
         do 130, iip=0,nip(ir)
            ast(ir,iip)=ast(ir,0)*prfrac(ir,iip)
            awt(ir,iip)=awt(ir,0)*prfrac(ir,iip)
            awtswp(ir,iip)=awt(ir,0)*prfrac(ir,iip)*factor
  130    continue
  140 continue
c
c...calculate average soil activities for gardening and indoors
      call soper(ast,sasttg,lambda,d,ttg,nr)
      call soper(ast,sasttr,lambda,d,ttr,nr)
c
c
c...calculate dose by pathway
c......external dose (Eqn. 5.69)
      call extern(mxnr,mxnip,dfer,nip,nr,prfrac,
     $    sasttg,sasttr,sfi,sfo,tg,ti,ttr,ttg,tx,dexr)

c......inhalation dose (Eqn. 5.70)
      call inhale(mxnr,mxnip,cdg,cdi,cdo,dfh,nip,nr,pd,prfrac,
     $    rfr,sasttg,sasttr,tg,ti,ttr,ttg,tx,vg,vr,vx,dhr)

c......ingestion dose from agricultural products (Eqn. 5.71)
      call agricl(mxnr,mxnip,ast,dfg,diet,nip,nr,pfs,prfrac,dgr)

c......soil ingestion dose (Eqn. 5.73)
      call soil(mxnr,mxnip,dfg,gr,nip,nr,prfrac,sasttr,
     $   tg,ti,ttr,tx,dsr)

c......drinking water (Eqn. 5.74, first part)
      call drink(mxnr,mxnip,awt,dfg,nip,nr,prfrac,td,uw,dwr1)

c......ingestion dose from irrigated products (Eqn. 5.74, second part)
      call irigat(mxnr,mxnip,awt,dfg,diet,nip,nr,pfw,prfrac,dwr2)

c......aquatic food ingestion dose (Eqn. 5.77)
      call aquatc(mxnr,mxnip,awtswp,baf,dfg,nip,nr,prfrac,tf,uf,dar)

      end
```

F.84 RESIDE.FOR

```
c***********************************************************************
c
c  PROGRAM RESIDE;  9/94
c  C. D. Updegraff, GRAM, Inc. Albuquerque, NM 87112.
c  Main Driver for the residential scenario equations from
c  NUREG/CR-5512, Vol. 1.
c  Benchmarked against RADDECAY Version 3.01 11/92 for the U-238 chain
c
c***********************************************************************
c
      include "cdecl.inc"

      subroutine RESIDE(for_rc)

      character*2 elemnt
      character*6 list
      character*6 nuclid,nucnam,nucrd
      character*6 short
      character*10 adate
      character*8 chain
      character*11 atime
      character*27 long
      character*80 confil
      character*80 hisfil
      character*80 infil
      character*80 outfil
      character*80 filfil
      character*80 chainfil,cedefil,resfil
      character*80 status_msg
      character*80 title
      double precision ar,astsv,awtsv
      double precision baf,bf,bg,bh,bv
      double precision cdg,cdi,cdo,conc
      double precision dfer,dfes,dfg,dfh,diet,dosmx,dossv
      double precision dt
      double precision f1,f2,fa,fca,fcd,fcf,fcg,fch,frac
      double precision fha,fhd,fhf,fhg,fhh,fhv
      double precision gr
      double precision h1,h2,hlif
      double precision i,ir
      double precision kd
      double precision lambda,lambdw
      double precision mlf,mlg,mlh,mlv
      double precision n1,n2,nucmx,nucsv
      double precision pathmx,pathsv
      double precision pd,ps,prfrac
      double precision qd,qf,qg,qh,qw
      double precision rf,rfr,rg,rh,rho1,rho2,rv
      double precision satac,sasvh,satah,sawvh
      double precision sfi,sfo,sh
      double precision tbigf,tbigg,tbigh,tbigv
      double precision tca,tcv,td
      double precision tdos,tedemx,tend
      double precision tf,tff,tfg,tfh,tfw,tg,tgf,tgg,tgv,tgh,tha,thv
      double precision ti,tnuc,tpath,tstart,ttede,ttg,ttr,tx
      double precision ua,uf,uv,uw
      double precision vdr,vg,vr,vsw,vx
```

```
      double precision wf,wg,wh,wv
      double precision xf,xg,xh,xw
      double precision yf,yg,yh,yv
      double precision zero
      integer filefn,infn,outfn,confn,hisfn
      integer chainfn,parfn,cedefn,kdfn,transfn,concenfn,biofn
      integer ichn,ierr,ioerr,ioerr2,ipar,ipos
      integer maxchn,maxlis,maxnam,maxpar,maxpth,maxpro
      integer nchn,nchnp2,nip,nipdis,nlay,nlist,nnam,npath,nr,nrrd
      integer period,printed,persv,tppath,tpdos,tpnuc
      integer skip
      integer*4 for_rc
      integer*4 c_rc
CCC...added to set the date of the run
      integer*2 month,day,year
      integer*2 hour,minute,second,hsecond
      logical carbon,combin
      logical distrb
      logical echain,exists
      logical histry,hydrog
      logical import
      parameter (maxchn=22,maxnam=1000,maxpar=20,maxpth=7)
      parameter (maxpro=6,npath=7)
      parameter (maxlis=maxchn*maxpar)
      parameter (zero=0.0d+0)
      dimension astsv(maxlis),awtsv(maxlis)
      dimension baf(maxpar,maxchn)
      dimension bf(4,maxpar,maxchn),bg(4,maxpar,maxchn)
      dimension bh(4,maxpar,maxchn),bv(4,maxpar,maxchn)
      dimension carbon(maxchn)
      dimension chain(maxchn)
      dimension conc(maxchn)
      dimension dfer(maxpar,0:maxpro,maxchn)
      dimension dfes(maxpar,0:maxpro,maxchn)
      dimension dfg(maxpar,0:maxpro,maxchn),dfh(maxpar,0:maxpro,maxchn)
      dimension dosmx(maxlis,maxpth)
      dimension dossv(maxlis,maxpth)
      dimension elemnt(maxpar)
      dimension echain(maxchn)
      dimension fa(4,maxpar,maxchn)
      dimension fca(4),fcf(4),fcg(4),fch(4)
      dimension fha(4),fhf(4),fhg(4),fhh(4),fhv(4)
      dimension frac(2,maxpar,maxchn)
      dimension hlif(maxpar,maxchn)
      dimension hydrog(maxchn)
      dimension ipar(2,maxpar,maxchn)
      dimension ipos(maxpar,maxchn)
      dimension kd(maxpar,maxchn)
      dimension lambda(maxpar,maxchn)
      dimension list(maxlis)
      dimension mlf(4),mlg(4),mlh(4),mlv(4)
      dimension nip(maxpar,maxchn),nipdis(maxpar,maxchn)
      dimension nr(maxchn),nrrd(maxchn)
      dimension nuclid(maxpar,0:maxpro,maxchn)
      dimension nucmx(maxlis)
      dimension nucnam(maxnam)
      dimension nucsv(maxlis)
      dimension nucrd(maxpar,maxchn)
      dimension pathmx(maxpth)
```

```
      dimension pathsv(maxpth)
      dimension prfrac(maxpar,0:maxpro,maxchn)
      dimension qd(4),qf(4),qg(4),qh(4),qw(4)
      dimension rf(4),rg(4),rh(4),rv(4)
      dimension tbigf(4),tbigg(4),tbigh(4),tbigv(4),tca(4),tcv(4)
      dimension tdos(maxlis,maxpth)
      dimension tff(4),tfg(4),tfh(4),tfw(4),tgf(4),tgg(4),tgh(4),tgv(4)
      dimension tha(4),thv(4)
      dimension tnuc(maxlis)
      dimension tpath(maxpth)
      dimension tppath(maxpth)
      dimension tpdos(maxlis,maxpth)
      dimension tpnuc(maxlis)
      dimension ua(4),uv(4)
      dimension wf(4),wg(4),wh(4),wv(4)
      dimension xf(4),xg(4),xh(4),xw(4)
      dimension yf(4),yg(4),yh(4),yv(4)
      data long /'Residential Scenario'/
      data short /'RESIDE'/
c

      ierr = 0
      for_rc = 1
      c_rc = 1

      filefn  = 50

      status_msg(1:80) = "Starting Calculational Modules..."C

      call CMessage(status_msg,c_rc)
      if (c_rc .eq. 0) then

            call closefiles(filefn,20)
            return

      endif

CCC...converted to Microsoft Powerstation FORTRAN functions
      call GetDat(year,month,day)
      write(adate,'(I2.2,A,I2.2,A,I4.4)',iostat=ioerr)
     $   month, '/', day, '/', year

      if (ioerr .ne. 0) then
            for_rc = 0
            call ioretint(ioerr)
            call closefiles(filefn,20)
            return
      endif

      call GetTim(hour,minute,second,hsecond)
      write(atime,'(I2.2,A,I2.2,A,I2.2)',iostat=ioerr)
     $   hour, ':', minute, ':',
     $        second

      if (ioerr .ne. 0) then
            for_rc = 0
            call ioretint(ioerr)
            call closefiles(filefn,20)
            return
```

```
      endif
c
c...read the input file number and name
      status_msg(1:80) = "Reading Input Filenames..."C
      call CMessage(status_msg,c_rc)

      if (c_rc .eq. 0) then
            call closefiles(filefn,20)
            return
      endif

      filfil = "data\resfnam"
      chainfil = "data\chain.dat"
      cedefil = "data\cede.dat"
      resfil = "data\res.dat"


c     assign the logical file units
      filefn  = 50
      infn    = filefn + 1
      outfn   = filefn + 2
      hisfn   = filefn + 3
      confn   = filefn + 4
      chainfn = filefn + 5
      parfn   = filefn + 6
      cedefn  = filefn + 7
      kdfn    = filefn + 8
      transfn = filefn + 9
      concenfn  = filefn + 10
      biofn   = filefn + 11

c
c...read the input file number and name
      OPEN(filefn,FILE=filfil,STATUS='OLD',MODE='READ',
     $     SHARE='DENYNONE',iostat=ioerr)

      if (ioerr .ne. 0) then
            for_rc = 0
            call ioreturn(ioerr)
            call closefiles(filefn,20)
            return
      endif

      read(filefn,*,iostat=ioerr) infil,outfil,hisfil,confil
      if (ioerr .ne. 0) then
            for_rc = 0
            call ioreturn(ioerr)
            call closefiles(filefn,20)
            return
      endif

      CLOSE(filefn,iostat=ioerr)

      if (ioerr .ne. 0) then
            for_rc = 0
            call ioreturn(ioerr)
            call closefiles(filefn,20)
            return
      endif
```

```
c
c...open output file
      open(outfn,file=outfil,status='unknown',mode='write',
     $     SHARE='DENYNONE',iostat=ioerr)

      if (ioerr .ne. 0) then
            for_rc = 0
            call ioreturn(ioerr)
            call closefiles(filefn,20)
            return
      endif
c

      status_msg(1:80) = "Reading User Input..."C
      call CMessage(status_msg,c_rc)

      if (c_rc .eq. 0) then
        call closefiles(filefn,20)
        return
      endif

c...read the user specified input data
      open(infn,file=infil,status='old',mode='read',
     $     SHARE='DENYNONE',iostat=ioerr)

      if (ioerr .ne. 0) then
            for_rc = 0
            call ioreturn(ioerr)
            call closefiles(filefn,20)
            return
      endif

      call input(infn,maxchn,title,histry,combin,import,distrb,
     $   nchn,chain,conc,ioerr,for_rc,nchnp2)

ccc     should the following be .GT. or .ne. ???


      if (ioerr .gt. 0) then
            for_rc = 0
            call closefiles(filefn,20)
            return
      endif

      if (for_rc .eq. 0) then
         status_msg(1:80)= "Error or Abort In Calculational Modules..."C
         call CMessage(status_msg,c_rc)
         call closefiles(filefn,20)
         return
      endif

      close (infn,iostat=ioerr)

      if (ioerr .ne. 0) then
            for_rc = 0
            call ioreturn(ioerr)
            call closefiles(filefn,20)
            return
```

```
      endif

      ierr = 0
c
c...if necessary, read chain and radionuclide
c   from concentration file
      if (import) then
          status_msg(1:80) = "Reading Concentrations..."C
          call CMessage(status_msg,c_rc)

          if (c_rc .eq. 0) then
              call closefiles(filefn,20)
              return
          endif

         open(chainfn,file=chainfil,status='old',mode='read',
     $        SHARE='DENYNONE',iostat=ioerr)

         if (ioerr .ne. 0) then
              for_rc = 0
              call ioreturn(ioerr)
              call closefiles(filefn,20)
              return
         endif

         call rdlist(chainfn,outfn,maxnam,ierr,nnam,nucnam,ioerr)

          if (ioerr .gt. 0) then
            for_rc = 0
            call closefiles(filefn,20)
            return
          endif

         close(chainfn,iostat=ioerr)

         if (ioerr .ne. 0) then
              for_rc = 0
              call ioreturn(ioerr)
              call closefiles(filefn,20)
              return
         endif

         if (ierr .ne. 0) then
              for_rc = 0
              call closefiles(filefn,20)
              return
         endif

         inquire(file=confil,exist=exists,iostat=ioerr)

         if (.not. exists) then

            write(outfn,'(3a)',iostat=ioerr2)
     $         'Import concentration file ''',
     $         confil,''' does not exist'

            if (ioerr2 .ne. 0) then
                  call ioreturn(ioerr2)
                  call closefiles(filefn,20)
```

```
                return
            endif
            ierr=1

          elseif (exists .and. ioerr .ne. 0) then
            for_rc = 0
            call ioreturn(ioerr)
            call closefiles(filefn,20)
            return

          endif


CCC     isn't this duplicate code from above???
          if (ierr .ne. 0) then
              for_rc = 0
              call closefiles(filefn,20)
              return
          endif

          open(confn,file=confil,status='old',mode='read',
     $      SHARE='DENYNONE',iostat=ioerr)

          if (ioerr .ne. 0) then
               for_rc = 0
               call ioreturn(ioerr)
               call closefiles(filefn,20)
               return
          endif

          call concrd(confn,outfn,maxchn,maxnam,maxpar,
     $       nnam,nucnam,chain,ierr,nchn,nrrd,nucrd,ioerr)

          if (ioerr .ne. 0) then
               for_rc = 0
               call closefiles(filefn,20)
               return
          endif

          if (ierr .ne. 0) then
               for_rc = 0
               call closefiles(filefn,20)
               return
          endif

        endif

        status_msg(1:80) = "Reading Variable Parameters..."C
        call CMessage(status_msg,c_rc)

        if (c_rc .eq. 0) then
          call closefiles(filefn,20)
          return
        endif

c
c...read the residential scenario default parameters
        open(parfn,file=resfil,status='old',mode='read',
```

```
$      SHARE='DENYNONE',iostat=ioerr)

 if (ioerr .ne. 0) then
       for_rc = 0
       call ioreturn(ioerr)
       call closefiles(filefn,20)
       return
 endif

 call resdef(parfn,ar,cdg,cdi,cdo,diet,f1,f2,gr,h1,h2,nlay,
$   i,ir,lambdw,mlf,mlg,mlh,mlv,n1,n2,pd,ps,qd,qf,qg,qh,qw,
$   rf,rfr,rg,rh,rho1,rho2,rv,sfi,sfo,tstart,tend,dt,skip,
$   tbigf,tbigg,tbigh,tbigv,
$   td,tca,tcv,tf,tff,tfg,tfh,tfw,tg,tgf,tgg,tgv,tgh,tha,
$   thv,ti,ttg,ttr,tx,ua,uf,uv,uw,vdr,vg,vr,vsw,vx,
$   wf,wg,wh,wv,xf,xg,xh,xw,yf,yg,yh,yv,fca,fcd,fcf,fcg,fch,satac,
$   fha,fhd,fhf,fhg,fhh,fhv,sasvh,satah,sawvh,sh,ioerr)

 if (ioerr .ne. 0) then
        for_rc = 0
        call closefiles(filefn,20)
        return
 endif

 close (parfn,iostat=ioerr)

 if (ioerr .ne. 0) then
       for_rc = 0
       call ioreturn(ioerr)
       call closefiles(filefn,20)
       return
 endif


c...write the user specified input data
      call resout(outfn,maxchn,adate,atime,short,long,title,
$   histry,combin,import,distrb,confil,
$   ar,cdg,cdi,cdo,diet,f1,f2,gr,h1,h2,
$   i,ir,lambdw,mlf,mlg,mlh,mlv,n1,n2,nlay,pd,ps,qd,qf,qg,qh,qw,
$   rf,rfr,rg,rh,rho1,rho2,rv,sfi,sfo,tbigf,tbigg,tbigh,tbigv,
$   tca,tcv,td,tf,tff,tfg,tfh,tfw,tg,tgf,tgg,tgv,tgh,tha,
$   thv,ti,ttg,ttr,tx,ua,uf,uv,uw,vdr,vg,vr,vsw,vx,
$   wf,wg,wh,wv,xf,xg,xh,xw,yf,yg,yh,yv,
$   fca,fcd,fcf,fcg,fch,satac,
$   fha,fhd,fhf,fhg,fhh,fhv,satah,sasvh,sawvh,sh,
$   tstart,tend,dt,skip,
$   nchn,chain,conc,ioerr,nchnp2)

      if (ioerr .ne. 0) then
           for_rc = 0
           call closefiles(filefn,20)
           return
      endif

c
c...check to see if any chains are the 'natural' chains
c   U_Nat or Th_Nat
      call naturl(maxchn,nchn,chain,conc)
c
```

```
c...if +C (equilibrium) designation exists on chain, then strip
c   +C designation from chain and set equilibrium flag
      call equil(maxchn,nchn,chain,echain)

c
c...determine if any of the chains are either C-14
c   or H-3 (tritium)
      call corh(maxchn,nchn,chain,carbon,hydrog)
c
      status_msg(1:80) = "Reading Data for Chain(s)..."C

      call CMessage(status_msg,c_rc)
      if (c_rc .eq. 0) then
          call closefiles(filefn,20)
          return
      endif

c...read chain data, cede data, kd data, animal product transfer
c   factors, soil-to-plant concentration factors, and
c   bioaccumulation factors

      open(chainfn,file='data\chain.dat',status='old',mode='read',
     $   SHARE='DENYNONE',iostat=ioerr,blocksize=39424)

      if (ioerr .ne. 0) then
            for_rc = 0
            call ioreturn(ioerr)
            call closefiles(filefn,20)
            return
      endif

      open(cedefn,file='data\cede.dat',status='old',mode='read',
     $   SHARE='DENYNONE',iostat=ioerr,blocksize=15360)

      if (ioerr .ne. 0) then
            for_rc = 0
            call ioreturn(ioerr)
            call closefiles(filefn,20)
            return
      endif

      open(kdfn,file='data\kd.dat',status='old',mode='read',
     $   SHARE='DENYNONE',iostat=ioerr)

      if (ioerr .ne. 0) then
            for_rc = 0
            call ioreturn(ioerr)
            call closefiles(filefn,20)
            return
      endif

      open(transfn,file='data\transfer.dat',status='old',mode='read',
     $   SHARE='DENYNONE',iostat=ioerr)

      if (ioerr .ne. 0) then
            for_rc = 0
            call ioreturn(ioerr)
            call closefiles(filefn,20)
            return
```

```
      endif

      open(concenfn,file='data\concen.dat',status='old',mode='read',
     $   SHARE='DENYNONE',iostat=ioerr)

      if (ioerr .ne. 0) then
            for_rc = 0
            call ioreturn(ioerr)
            call closefiles(filefn,20)
            return
      endif

      open(biofn,file='data\bioaccum.dat',status='old',mode='read',
     $   SHARE='DENYNONE',iostat=ioerr)

      if (ioerr .ne. 0) then
            for_rc = 0
            call ioreturn(ioerr)
            call closefiles(filefn,20)
            return
      endif


      for_rc = 1

      do 10, ichn=1,nchn

c......read the chain data
         call rdchn(chainfn,outfn,for_rc,maxpar,maxpro,chain(ichn),
     $      nr(ichn), nuclid(1,0,ichn),hlif(1,ichn),lambda(1,ichn),
     $      ipos(1,ichn),ipar(1,1,ichn),frac(1,1,ichn),ioerr)

          if (ioerr .gt. 0) then
            for_rc = 0
            call closefiles(filefn,20)
            return
          endif

         if (for_rc .eq. 0) then
            exit
         endif

c......if import concentration file is used, check for chain
c      compatibility; if not, stop
          if (import)
     $      call contst(outfn,maxpar,nr(ichn),nrrd(ichn),
     $         nuclid(1,0,ichn),nucrd(1,ichn),ioerr)

              if (ioerr .ne. 0) then
                  for_rc = 0
                  call closefiles(filefn,20)
                  return
              endif

c......read the cede data
         call rdcede(cedefn,outfn,for_rc,maxpar,maxpro,combin,nr(ichn),
     $      nip(1,ichn),nipdis(1,ichn),nuclid(1,0,ichn),
     $      prfrac(1,0,ichn),dfer(1,0,ichn),dfes(1,0,ichn),
     $      dfg(1,0,ichn),dfh(1,0,ichn),ioerr)
```

```
            if (ioerr .ne. 0) then
                for_rc = 0
                call closefiles(filefn,20)
                return
            endif

            if (for_rc .eq. 0) then
                exit
            endif

c......find the elemental names of radionuclides in chain
          call nucit(maxpar,nr(ichn),nuclid(1,0,ichn),elemnt)

c......read the partition coefficient (kd) data
          call rdkd(kdfn,outfn,for_rc,maxpar,nr(ichn),elemnt,
     $      nuclid(1,0,ichn),
     $      kd(1,ichn),ioerr)

            if (ioerr .ne. 0) then
                for_rc = 0
                call closefiles(filefn,20)
                return
            endif

            if (for_rc .eq. 0) then
                exit
            endif


c......read the animal product transfer factor data
         call rdtran(transfn,outfn,for_rc,maxpar,nr(ichn),elemnt,
     $      nuclid(1,0,ichn),
     $      fa(1,1,ichn),ioerr)

            if (ioerr .ne. 0) then
                for_rc = 0
                call closefiles(filefn,20)
                return
            endif

            if (for_rc .eq. 0) then
                exit
            endif

c......read the soil-to-plant concentration data
         call rdconc(concenfn,outfn,for_rc,maxpar,nr(ichn),elemnt,
     $      nuclid(1,0,ichn),
     $      bv(1,1,ichn),bf(1,1,ichn),bg(1,1,ichn),bh(1,1,ichn),ioerr)

            if (ioerr .ne. 0) then
                for_rc = 0
                call closefiles(filefn,20)
                return
            endif

            if (for_rc .eq. 0) then
                exit
            endif
```

```
c......read the bioaccumulation data
         call rdbio(biofn,outfn,for_rc,maxpar,nr(ichn),elemnt,
     $       nuclid(1,0,ichn),
     $       baf(1,ichn),ioerr)

              if (ioerr .ne. 0) then
                  for_rc = 0
                  call closefiles(filefn,20)
                  return
              endif

              if (for_rc .eq. 0) then
                  exit
              endif

   10 continue


      close (chainfn,iostat=ioerr)
      if (ioerr .ne. 0) then
            for_rc = 0
            call ioreturn(ioerr)
            call closefiles(filefn,20)
            return
      endif

      close (cedefn,iostat=ioerr)
      if (ioerr .ne. 0) then
            for_rc = 0
            call ioreturn(ioerr)
            call closefiles(filefn,20)
            return
      endif

      close (kdfn,iostat=ioerr)
      if (ioerr .ne. 0) then
            for_rc = 0
            call ioreturn(ioerr)
            call closefiles(filefn,20)
            return
      endif

      close (transfn,iostat=ioerr)
      if (ioerr .ne. 0) then
            for_rc = 0
           call ioreturn(ioerr)
            call closefiles(filefn,20)
            return
      endif

      close (concenfn,iostat=ioerr)
      if (ioerr .ne. 0) then
            for_rc = 0
            call ioreturn(ioerr)
            call closefiles(filefn,20)
            return
      endif

      close (biofn,iostat=ioerr)
```

```
      if (ioerr .ne. 0) then
            for_rc = 0
            call ioreturn(ioerr)
            call closefiles(filefn,20)
            return
      endif


      if (for_rc .eq. 0) then
          status_msg(1:80) = "Error In Calculational Modules..."C
          call CMessage(status_msg,c_rc)
          call closefiles(filefn,20)
          return
      endif

c

      status_msg(1:80) = "Starting Calculations..."C

      call CMessage(status_msg,c_rc)
      if (c_rc .eq. 0) then
          call closefiles(filefn,20)
          return
      endif

c...perform the maximum dose and history calculations
      if (histry) then
         open(hisfn,file=hisfil,status='unknown',mode='write',
     $        SHARE='DENYNONE',iostat=ioerr)

         if (ioerr .ne. 0) then
              for_rc = 0
              call ioreturn(ioerr)
              call closefiles(filefn,20)
              return
         endif

      endif

      call resmax(outfn,for_rc,maxchn,maxlis,maxpar,maxpth,maxpro,npath,
     $   hisfn,histry,confn,import,distrb,carbon,hydrog,
     $   ar,cdg,cdi,cdo,diet,f1,f2,
     $   gr,h1,h2,i,ir,lambdw,mlf,mlg,mlh,mlv,n1,n2,nlay,pd,ps,
     $   qd,qf,qg,qh,qw,rf,rfr,rg,rh,rho1,rho2,rv,
     $   sfi,sfo,tbigf,tbigg,tbigh,tbigv,tca,tcv,td,tf,tff,tfg,
     $   tfh,tfw,tg,tgg,tgv,tgh,tha,thv,ti,ttg,ttr,tx,ua,uf,uv,uw,
     $   vdr,vg,vr,vsw,vx,wf,wg,wh,wv,xf,xg,xh,xw,yf,yg,yh,yv,
     $   baf,bf,bg,bh,bv,dfer,dfes,dfg,dfh,fa,kd,fca,fcd,fcf,fcg,fch,
     $   satac,fha,fhd,fhf,fhg,fhh,fhv,sasvh,satah,sawvh,sh,
     $   tstart,tend,dt,skip,
     $   nchn,chain,echain,conc,nuclid,nr,nip,nipdis,hlif,lambda,
     $   ipos,ipar,frac,prfrac,
     $   nlist,ttede,period,printed,persv,
     $   list,astsv,awtsv,dossv,nucsv,pathsv,tedemx,
     $   dosmx,tdos,tpdos,pathmx,tpath,tppath,nucmx,tnuc,tpnuc,ioerr)

      if (ioerr .ne. 0) then
              for_rc = 0
              call closefiles(filefn,20)
```

```
            return
      endif

      if (for_rc .eq. 0) then
            call closefiles(filefn,20)
            return
      endif


      if (histry) then

         close (hisfn,iostat=ioerr)

         if (ioerr .ne. 0) then
              for_rc = 0
              call ioreturn(ioerr)
              call closefiles(filefn,20)
              return
         endif

      endif

      if (import) then

         close (confn,iostat=ioerr)

         if (ioerr .ne. 0) then
              for_rc = 0
              call ioreturn(ioerr)
              call closefiles(filefn,20)
              return
         endif

      endif
c

      if (for_rc .eq. 0) then
          status_msg(1:80)="Error or Abort In Calculational Modules..."C
          call CMessage(status_msg,c_rc)
          call closefiles(filefn,20)
          return
      endif

      status_msg(1:80) = "Writing Calculational Modules Results..."C

      call CMessage(status_msg,c_rc)

      if (c_rc .eq. 0) then
          call closefiles(filefn,20)
          return
      endif

c...write out maximum results

      call wrres(outfn,maxpth,maxlis,npath,nlist,
     $   ttede,ttr,period,printed,persv,
     $   list,astsv,awtsv,dossv,nucsv,pathsv,tedemx,
     $   dosmx,tdos,tpdos,pathmx,tpath,tppath,nucmx,tnuc,tpnuc,ioerr)
```

```
      if (ioerr .ne. 0) then
          for_rc = 0
          call closefiles(filefn,20)
          return
      endif
c

      close(outfn,iostat=ioerr)

      if (ioerr .ne. 0) then
          for_rc = 0
          call closefiles(filefn,20)
          return
      endif

      for_rc = 1

      status_msg(1:80) = "Completed Calculational Modules..."C
      call CMessage(status_msg,c_rc)
      call closefiles(filefn,20)

      end
```

F.85 RESMAX.FOR

```
c
c  This subroutine performs the maximum dose calculations
c  and updates the history file for selected time steps
c
      include "cdecl.inc"

      subroutine resmax(out,rc,mxchn,mxlist,mxpar,mxpth,mxpro,npath,
     $   his,histry,conunt,import,distrb,carbon,hydrog,
     $   ar,cdg,cdi,cdo,diet,f1,f2,
     $   gr,h1,h2,i,irr,lambdw,mlf,mlg,mlh,mlv,n1,n2,nlay,pd,ps,
     $   qd,qf,qg,qh,qw,rf,rfr,rg,rh,rho1,rho2,rv,
     $   sfi,sfo,tbigf,tbigg,tbigh,tbigv,tca,tcv,td,tf,tff,tfg,
     $   tfh,tfw,tg,tgg,tgv,tgh,tha,thv,ti,ttg,ttr,tx,ua,uf,uv,uw,
     $   vdr,vg,vr,vsw,vx,wf,wg,wh,wv,xf,xg,xh,xw,yf,yg,yh,yv,
     $   baf,bf,bg,bh,bv,dfer,dfes,dfg,dfh,fa,kd,fca,fcd,fcf,fcg,fch,
     $   satac,fha,fhd,fhf,fhg,fhh,fhv,sasvh,satah,sawvh,sh,
     $   tstart,tend,dt,skip,
     $   nchn,chain,echain,conc,nuclid,nr,nip,nipdis,hlif,lambda,
     $   ipos,ipar,frac,prfrac,
     $   nlist,ttede,period,printed,persv,
     $   list,astsv,awtsv,dossv,nucsv,pathsv,tedemx,
     $   dosmx,tdos,tpdos,pathmx,tpath,tppath,nucmx,tnuc,tpnuc,ioerr)

      character*6 list
      character*6 nuclid
      character*8 chain
      character*80 status_msg
      double complex kcoef
      double complex root
      double complex sr3new
      double precision ar,ar1new,ar1old
      double precision ast,astm1,astsv,astup,atol
      double precision awt,awtm1,awtsv,awtswp,awtup
      double precision baf,bf,bg,bh,bv
      double precision cdg,cdi,cdo,con,conc,conr
      double precision d,dd,ddd,dfer,dfes,dfg,dfh,diet
      double precision dos,dosmx,dossv,dosup
      double precision dt
      double precision fa,f1,f2,fca,fcd,fcf,fcg,fch
      double precision fha,fhd,fhf,fhg,fhh,fhv,fr,frac
      double precision gr
      double precision h1,h2,hlif
      double precision i,irr
      double precision kd
      double precision l,l12,l23,lambda,lambdw
      double precision mlf,mlg,mlh,mlv
      double precision n1,n2,nuc,nucmx,nucsv,nucup
      double precision path,pathmx,pathup,pathsv,pd,pfs,pfw
      double precision ps,prfrac
      double precision qd,qf,qg,qh,qw
      double precision rf,rfr,rg,rh,rho1,rho2,rtol,rv
      double precision sasgm1,sasgup,sasrm1,sasrup
      double precision sasvh,satac,satah,sawvh
      double precision sfi,sfo,sh,smalli
      double precision tbigf,tbigg,tbigh,tbigv
      double precision tca,tcalc,tcv,td,tdos,tede,tedemx,tedeup,tend
      double precision tf,tff,tfg,tfh,tfw,tg,tgg,tgv,tgh,tha,thv
```

```
double precision ti,tnew,tnuc,told,tpath,tsav,tstart,tstop
double precision ttede,ttg,ttr,tx
double precision ua,uf,uv,uw
double precision vdr,vg,vr,vsw,vtr,vx
double precision wf,wg,wh,wr,wv
double precision xf,xg,xh,xw
double precision yf,yg,yh,yv
double precision zero
integer achk
integer chnnum,conunt
integer his
integer ia,ibox,ichn,iconn,ieq,indx,ioerr,ipar,ipos,ir
integer istate
integer ja
integer maxbox,maxchn,maxlis,maxneq,maxnnz,maxpar,maxpth,maxpro
integer mxchn,mxlist,mxpar,mxpth,mxpro
integer nbox,nchn,neq,nip,nipdis,nlay,nlist,nnz,npath,nr
integer out
integer period,printed,persv
integer rc
integer skip
integer tppath,tpdos,tpnuc,tsteps
logical carbon
logical distrb,done
logical echain
logical histry,hydrog
logical import
logical infil0
logical iszero
logical report
logical dcreas
parameter (maxbox=12,maxchn=22,maxpar=20,maxpro=6,maxpth=7)
parameter (maxlis=maxchn*maxpar)
parameter (maxneq=maxbox*maxpar)
parameter (maxnnz=(maxbox*maxpar*(maxpar+3))/2)
parameter (smalli=0.0d+0)
parameter (zero=0.0d+0)
dimension ar1new(maxneq,maxchn),ar1old(maxneq,maxchn)
dimension ast(maxpar,0:maxpro),astm1(maxlis)
dimension astsv(mxlist)
dimension awtswp(maxpar,0:maxpro)
dimension astup(maxlis)
dimension atol(maxchn)
dimension awt(maxpar,0:maxpro),awtm1(maxlis)
dimension awtsv(mxlist),awtup(maxlis)
dimension baf(mxpar,mxchn)
dimension bf(4,mxpar,mxchn),bg(4,mxpar,mxchn)
dimension bh(4,mxpar,mxchn),bv(4,mxpar,mxchn)
dimension carbon(mxchn)
dimension chain(mxchn)
dimension con(maxpar,maxchn)
dimension conc(mxchn)
dimension conr(maxbox,maxpar,maxchn)
dimension d(maxpar,maxpar,maxchn)
dimension dd((maxpar*(maxpar-1))/2,maxchn)
dimension ddd(maxnnz,maxchn)
dimension dfer(mxpar,0:mxpro,mxchn),dfes(mxpar,0:mxpro,mxchn)
dimension dfg(mxpar,0:mxpro,mxchn),dfh(mxpar,0:mxpro,mxchn)
dimension dos(maxpar,0:maxpro,maxpth)
```

```
      dimension dosmx(mxlist,mxpth)
      dimension dossv(mxlist,mxpth),dosup(maxlis,maxpth)
      dimension echain(mxchn)
      dimension fa(4,mxpar,mxchn)
      dimension fca(4),fcf(4),fcg(4),fch(4)
      dimension fha(4),fhf(4),fhg(4),fhh(4),fhv(4)
      dimension frac(2,mxpar,mxchn)
      dimension hlif(mxpar,mxchn)
      dimension hydrog(mxchn)
      dimension ia(maxneq+1,maxchn)
      dimension iconn(maxpar,maxchn)
      dimension indx(maxpar,0:maxpro,maxchn)
      dimension ipar(2,mxpar,mxchn)
      dimension ipos(mxpar,mxchn)
      dimension ja(maxnnz,maxchn)
      dimension kcoef(3,3,maxpar,maxpar,maxchn)
      dimension kd(mxpar,mxchn)
      dimension l(maxbox,maxpar,maxchn)
      dimension l12(maxpar,maxchn),l23(maxpar,maxchn)
      dimension lambda(mxpar,mxchn)
      dimension list(mxlist)
      dimension mlf(4),mlg(4),mlh(4),mlv(4)
      dimension neq(maxchn)
      dimension nip(mxpar,mxchn),nipdis(mxpar,mxchn)
      dimension nnz(maxchn)
      dimension nr(mxchn)
      dimension nuc(maxpar,0:maxpro)
      dimension nuclid(mxpar,0:mxpro,mxchn)
      dimension nucmx(mxlist),nucsv(mxlist),nucup(maxlis)
      dimension path(maxpth),pathmx(mxpth)
      dimension pathsv(mxpth),pathup(maxpth)
      dimension pfs(maxpar,maxpar,maxchn),pfw(maxpar,maxpar,maxchn)
      dimension prfrac(mxpar,0:mxpro,mxchn)
      dimension qd(4),qf(4),qg(4),qh(4),qw(4)
      dimension rf(4),rg(4),rh(4)
      dimension root(3,maxpar,maxchn)
      dimension rtol(maxchn)
      dimension rv(4)
      dimension sasgml(maxlis),sasgup(maxlis)
      dimension sasrml(maxlis),sasrup(maxlis)
      dimension sr3new(3,maxpar,maxchn)
      dimension tbigf(4),tbigg(4),tbigh(4),tbigv(4),tca(4),tcv(4)
      dimension tdos(mxlist,mxpth)
      dimension tff(4),tfg(4),tfh(4),tfw(4)
      dimension tgg(4),tgv(4),tgh(4),tha(4),thv(4)
      dimension tnuc(mxlist)
      dimension tpath(mxpth)
      dimension tpdos(mxlist,mxpth)
      dimension tpnuc(mxlist)
      dimension tppath(mxpth)
      dimension ua(4),uv(4)
      dimension wf(4),wg(4),wh(4),wv(4)
      dimension xf(4),xg(4),xh(4),xw(4)
      dimension yf(4),yg(4),yh(4),yv(4)
      common /ddd/ ddd
      common /ia/ ia
      common /ja/ ja
      common /chnnum/ chnnum
c
```

```
c...
      tsteps=((tend-tstart)/dt)
      if ((dt .eq. zero) .or. ((tend-tstart) .le. 100.0d+0)) then
         achk=10
      else
         achk=tsteps/20
      endif
      if (achk .eq. 0) achk=10
c
c...determine if infiltration is small enough to be 0
      if (i .lt. smalli) then
         infil0=.true.
      else
         infil0=.false.
      end if
c
c...perform preliminary calculations
      nbox=nlay+2

c...if import concentration file is used
      if (import) then

         do 10, ichn=1,nchn

c......set up the initial activities for each radionuclide in the chain
            call initzd(mxpar,con(1,ichn))

c......write chain, cede and activity data to output file
            call wrchn(out,mxpar,mxpro,nr(ichn),nip(1,ichn),ichn,
     $         chain(ichn),echain(ichn),nuclid(1,0,ichn),ipos(1,ichn),
     $         ipar(1,1,ichn),frac(1,1,ichn),hlif(1,ichn),
     $         prfrac(1,0,ichn),con(1,ichn),dfg(1,0,ichn),
     $         dfh(1,0,ichn),dfes(1,0,ichn),dfer(1,0,ichn),'RESIDE',
     $         ioerr)

              if (ioerr .ne. 0) then
                 return
              endif

c......convert dfer, dfes, dfg, and dfh from Bq-Sv system to
c      mrem-H-pCi system
            call convrt(mxpar,mxpro,nr(ichn),nip(1,ichn),
     $         dfg(1,0,ichn),dfh(1,0,ichn),
     $         dfes(1,0,ichn),dfer(1,0,ichn))

c......set up the fractional yield array
            call frctnl(mxpar,nr(ichn),d(1,1,ichn),
     $         frac(1,1,ichn),ipar(1,1,ichn))

c...calculate pathway transfer factors for water and soil
            call pf(mxpar,nr(ichn),carbon(ichn),hydrog(ichn),
     $         bf(1,1,ichn),bg(1,1,ichn),bh(1,1,ichn),bv(1,1,ichn),
     $         d(1,1,ichn),
     $         fa(1,1,ichn),fca,fcd,fcf,fcg,fch,fha,fhd,fhf,fhg,fhh,fhv,
     $         irr,lambda(1,ichn),lambdw,mlf,mlg,mlh,mlv,ps,
     $         qd,qf,qg,qh,qw,rf,rg,rh,rv,sasvh,satac,satah,sawvh,sh,
     $         tbigf,tbigg,tbigh,tbigv,tca,tcv,tff,tfg,tfh,tfw,
     $         tgg,tgh,tgv,tha,thv,ua,uv,
     $         wf,wg,wh,wv,xf,xg,xh,xw,yf,yg,yh,yv,
```

```
     $          pfs(1,1,ichn),pfw(1,1,ichn))

c......calculate transfer coefficients between boxes
c          Inserted the following call because vtr needs to be
c          calculated or else you get a divide by zero. 3/21/96 MAW

            call trnsfr(maxbox,mxpar,nbox,nlay,nr(ichn),h1,h2,n1,n2,
     $          f1,f2,rho1,rho2,irr,i,kd(1,ichn),ar,vdr,vtr,fr,wr,
     $          l(1,1,ichn))

   10    continue

      else

         do 20, ichn=1,nchn

c......set up the initial activities for each radionuclide in the chain
            call concen(mxpar,mxpro,nr(ichn),echain(ichn),distrb,
     $          lambda(1,ichn),ipar(1,1,ichn),frac(1,1,ichn),
     $          prfrac(1,0,ichn),nipdis(1,ichn),conc(ichn),con(1,ichn))

c......write chain, cede and activity data to output file
            call wrchn(out,mxpar,mxpro,nr(ichn),nip(1,ichn),ichn,
     $          chain(ichn),echain(ichn),nuclid(1,0,ichn),ipos(1,ichn),
     $          ipar(1,1,ichn),frac(1,1,ichn),hlif(1,ichn),
     $          prfrac(1,0,ichn),con(1,ichn),dfg(1,0,ichn),
     $          dfh(1,0,ichn),dfes(1,0,ichn),dfer(1,0,ichn),'RESIDE',
     $          ioerr)

               if (ioerr .ne. 0) then
                  return
               endif

c......convert dfer, dfes, dfg, and dfh from Bq-Sv system to
c      mrem-H-pCi system
            call convrt(mxpar,mxpro,nr(ichn),nip(1,ichn),
     $          dfg(1,0,ichn),dfh(1,0,ichn),
     $          dfes(1,0,ichn),dfer(1,0,ichn))

c......set up the fractional yield array
            call frctnl(mxpar,nr(ichn),d(1,1,ichn),
     $          frac(1,1,ichn),ipar(1,1,ichn))

c...calculate pathway transfer factors for water and soil
            call pf(mxpar,nr(ichn),carbon(ichn),hydrog(ichn),
     $          bf(1,1,ichn),bg(1,1,ichn),bh(1,1,ichn),bv(1,1,ichn),
     $          d(1,1,ichn),
     $          fa(1,1,ichn),fca,fcd,fcf,fcg,fch,fha,fhd,fhf,fhg,fhh,fhv,
     $          irr,lambda(1,ichn),lambdw,mlf,mlg,mlh,mlv,ps,
     $          qd,qf,qg,qh,qw,rf,rg,rh,rv,sasvh,satac,satah,sawvh,sh,
     $          tbigf,tbigg,tbigh,tbigv,tca,tcv,tff,tfg,tfh,tfw,
     $          tgg,tgh,tgv,tha,thv,ua,uv,
     $          wf,wg,wh,wv,xf,xg,xh,xw,yf,yg,yh,yv,
     $          pfs(1,1,ichn),pfw(1,1,ichn))

c......calculate transfer coefficients between boxes
            call trnsfr(maxbox,mxpar,nbox,nlay,nr(ichn),h1,h2,n1,n2,
     $          f1,f2,rho1,rho2,irr,i,kd(1,ichn),ar,vdr,vtr,fr,wr,
     $          l(1,1,ichn))
```

```
c......set up initial radionuclide activities into format
c      useable by n-box model
            call conres(maxbox,mxpar,nbox,nr(ichn),con(1,ichn),
     $         conr(1,1,ichn))

c......set up fractional yield array in vector form
c      for use by 3-box model analytical or n-box numerical solution
            if (nlay .eq. 1) then

               call fract1(mxpar,nr(ichn),ipar(1,1,ichn),
     $            frac(1,1,ichn),dd(1,ichn))
c......set up the coefficients for the 3-box model

               status_msg(1:80)="Calculating coefficients for decaying"C
               call CMessage(status_msg,rc)

               status_msg(1:80)="exponential terms for analytical"C
               call CMessage(status_msg,rc)

               status_msg(1:80)="solution to 3-box model for "C

               write(status_msg(29:39),'(a)',iostat=ioerr) chain(ichn)

              if (ioerr .ne. 0) then
                 call ioretint(ioerr)
                 return
              endif

               call CMessage(status_msg,rc)

               if (rc .eq. 0) then
                  return
               endif
c
               do 240, ir=1,nr(ichn)
                  l12(ir,ichn)=l(1,ir,ichn)
                  l23(ir,ichn)=l(2,ir,ichn)
  240          continue

               call boxset(rc,maxbox,mxpar,nr(ichn),
     $            kcoef(1,1,1,1,ichn),root(1,1,ichn),iconn(1,ichn),
     $            conr(1,1,ichn),dd(1,ichn),l12(1,ichn),l23(1,ichn),
     $            lambda(1,ichn),fr,wr,ioerr)

              if (ioerr .ne. 0) then
                 return
              endif

               if (rc .eq. 0) then
                  return
               endif

            else
c......set up coefficients and tolerances for n-box model
               call sparse(d(1,1,ichn),fr,l(1,1,ichn),
     $            lambda(1,ichn),nbox,nr(ichn),ddd(1,ichn),
     $            ia(1,ichn),ja(1,ichn),neq(ichn),nnz(ichn))

               call toler(conr(1,1,ichn),maxneq,
```

```
     $            atol(ichn),rtol(ichn))

            endif

   20     continue
       endif
c

c...combine radionnuclides from all chains into a unique list of
c   radionuclides and tie each radionuclide from each chain into
c   an index number in the list
       call nuclis(mxlist,mxchn,mxpar,mxpro,
      $   nlist,nchn,nr,nip,nuclid,list,indx)
c
       printed=0

       status_msg="Calculating up to              "
       write(status_msg(19:23),'(i4)',iostat=ioerr) tsteps

       if (ioerr .ne. 0) then
           call ioretint(ioerr)
           return
       endif

       status_msg=status_msg(1:24)//' timesteps...'C
       call CMessage(status_msg,rc)
c
c...calculate doses
c......initialize some time parameters
       call init0(out,rc,conunt,import,ttr,tstart,
      $   tend,period,tcalc,tstop,ioerr)

       if (ioerr .ne. 0) then
           return
       endif

       if (rc .eq. 0) then
           return
       endif

c......initialize maximum values of comparison variables
       call initzd(mxlist*mxpth,dosmx)
       call initzd(mxlist,nucmx)
       call initzd(mxpth,pathmx)
       call initzd(1,tedemx)
       call initzd(maxlis,sasgml)
       call initzd(maxlis,sasrml)
       call initzd(maxlis,astml)
       call initzd(maxlis,awtml)
       call initlg(1,.false.,dcreas)
       call initzc(3*maxpar*maxchn,sr3new)
       call initzd(maxneq*maxchn,ar1old)

c...find concentrations at start time if the numerical model is used
       if (nlay .gt. 1) then

          do 50, ichn=1,nchn

             chnnum=ichn
```

```
            do 40, ir=1,nr(ichn)

               do 30, ibox=1,nbox
                  ar1old((ir-1)*nbox+ibox,ichn)=conr(ibox,ir,ichn)
   30          continue

   40       continue

            told=zero
            tnew=tstart
            istate=1

            call solvde(atol(ichn),ar1old(1,ichn),ia(1,ichn),istate,
     $         ja(1,ichn),neq(ichn),nnz(ichn),rtol(ichn),told,
     $         tnew)

            if (istate .lt. 0) then
               rc=0
               return
            endif

   50    continue

      endif

   55 continue

c......determine if current calculation time is a report time
      call rpttim(tcalc,tstop,period,skip,histry,report)

c......calculate dose for each radionuclide in each chain
c       and for each pathway
c......initialize update variables for the current time step
      call initzd(mxlist*mxpth,dosup)
      call initzd(mxlist,nucup)
      call initzd(mxpth,pathup)
      call initzd(mxlist,sasrup)
      call initzd(mxlist,sasgup)
      call initzd(mxlist,astup)
      call initzd(mxlist,awtup)
      call initzd(1,tedeup)
c
      do 60, ichn=1,nchn

c.........calculate dose in chain for each radionuclide
c          for each pathway
         chnnum=ichn
         call resdos(rc,mxpar,mxpro,ar,baf(1,ichn),cdg,cdi,cdo,
     $      con(1,ichn),conunt,import,infil0,atol(ichn),ar1old(1,ichn),
     $      d(1,1,ichn),ddd(1,ichn),dfer(1,0,ichn),dfg(1,0,ichn),
     $      dfh(1,0,ichn),diet,dt,gr,h1,ia(1,ichn),ja(1,ichn),
     $      kcoef(1,1,1,1,ichn),lambda(1,ichn),nbox,
     $      neq(ichn),nip(1,ichn),nlay,nnz(ichn),nr(ichn),pd,period,
     $      pfs(1,1,ichn),pfw(1,1,ichn),prfrac(1,0,ichn),rfr,rho1,
     $      root(1,1,ichn),rtol(ichn),sfi,sfo,sr3new(1,1,ichn),tcalc,
     $      td,tf,tg,ti,ttg,ttr,tx,
     $      uf,uw,vg,vr,vsw,vtr,vx,ar1new(1,ichn),ast,awt,awtswp,
     $      dos(1,0,1),dos(1,0,2),dos(1,0,3),dos(1,0,4),dos(1,0,5),
     $      dos(1,0,6),dos(1,0,7),ioerr)
```

```
            if (ioerr .ne. 0) then
                return
            endif

          if (rc .eq. 0) then
             return
          endif
c
c.........calculate pathway dose for chain
         call pthsum(mxpar,mxpro,mxpth,nr(ichn),nip(1,ichn),npath,
     $       dos,path)
c
c.........calculate radionuclide dose for chain
         call nucsum(mxpar,mxpro,mxpth,nr(ichn),nip(1,ichn),npath,
     $       dos,nuc)
c
c.........calculate tede for chain
         call tedsum(mxpar,mxpro,nr(ichn),nip(1,ichn),nuc,tede)
c
c.........update listed radionuclide soil activity
         call upsbar(mxpar,mxpro,mxlist,nr(ichn),nip(1,ichn),
     $      indx(1,0,ichn),ast,astup)
c
c.........update listed radionuclide average groundwater activity
         call upsbar(mxpar,mxpro,mxlist,nr(ichn),nip(1,ichn),
     $      indx(1,0,ichn),awt,awtup)
c
c.........update listed radionuclide dose for each pathway for all chains
         call updose(mxpar,mxpro,mxpth,mxlist,
     $      nr(ichn),nip(1,ichn),npath,
     $      indx(1,0,ichn),dos,dosup)
c
c.........update pathway dose for all chains
         call uppath(mxpth,npath,path,pathup)
c
c.........update listed radionuclide dose for all chains
         call upnuc(mxpar,mxpro,mxlist,nr(ichn),nip(1,ichn),
     $      indx(1,0,ichn),nuc,nucup)
c
c.........update total dose for all chains
         call uptede(tede,tedeup)

   60 continue

c
c...if this is a report time write results to history file
      if (report) then

         call wrhist(his,mxlist,mxpth,nlist,npath,
     $      tcalc,tcalc+ttr,list,astup,awtup,dosup,
     $      pathup,nucup,tedeup,ioerr)

          if (ioerr .ne. 0) then
              return
          endif

         printed=printed+1

      endif
```

```
c
c......if integrated activities are increasing, then test for
c      maximum values of total dose (tede) and find its components,
c      pathway dose, listed radionuclide dose, and listed
c      radionuclide dose by pathway
      if (.not. dcreas) then

c......check if total tede is maximum so far; if so, save it,
c      its components, and its time of occurrence
         call tedmxr(mxlist,mxpth,nlist,npath,tcalc,period,persv,
     $      ttede,tedeup,tedemx,pathsv,pathup,nucsv,nucup,dossv,dosup,
     $      astsv,astup,awtsv,awtup)

c......check for maximum pathway doses; if so, save them and their
c      time of occurrence
         call pthmax(mxpth,npath,tcalc,period,tpath,tppath,
     $      pathup,pathmx)

c......check for maximum listed radionuclide doses; if so, save
c      them and their time of occurrence
         call nucmax(mxlist,nlist,tcalc,period,tnuc,tpnuc,nucup,nucmx)

c......check for maximum listed radionuclide doses per pathway; if so,
c      save them and their time of occurence
         call dosmax(mxpth,mxlist,npath,nlist,tcalc,period,dosup,
     $      tdos,tpdos,dosmx)
      endif

c......determine if the soil activities, soil integrated activities,
c      are and water integrated activities are decreasing. if they
c      are all decreasing, set a test value to true and check history
c      value to determine if maximum results should be written out at
c      this time
      iszero=.true.
      call initlg(1,.true.,dcreas)
      call valtst(nlist,sasgup,sasgml,dcreas,iszero)
      call valtst(nlist,sasrup,sasrml,dcreas,iszero)
      call valtst(nlist,astup,astml,dcreas,iszero)
      if (.not. infil0) call valtst(nlist,awtup,awtml,dcreas,iszero)

      if (dcreas .and. .not. histry) then
         status_msg="All doses are decreasing..."C
         call CMessage(status_msg,rc)
         goto 95
      endif

      if (dcreas .and. iszero) then
         status_msg="All doses are decreasing and under minimum..."C
         call CMessage(status_msg,rc)
         goto 95
      endif

      if (tcalc .ge. tstop) goto 95

c......update time
      tsav=tcalc
      call uptime(conunt,import,done,period,tcalc,dt,tstop,ioerr)

      if (ioerr .ne. 0) then
```

```
            return
      endif
c
      if (mod(period,achk) .eq. 1) then

         write(status_msg(1:10),'(i10)',iostat=ioerr) period

          if (ioerr .ne. 0) then
              call ioretint(ioerr)
              return
          endif

         status_msg=status_msg(1:10)//' timestep(s) out of '

         write(status_msg(31:34),'(i4)',iostat=ioerr) tsteps

          if (ioerr .ne. 0) then
              call ioretint(ioerr)
              return
          endif

         status_msg=status_msg(1:34)//' calculated...'C

         call CMessage(status_msg,rc)

         if (rc .eq. 0) then
            return
         endif

      else

         call CProcess(rc)
         if (rc .eq. 0) then
            return
         endif

      endif
c
      if (done) goto 95

c......for numerical solution, update activities if another
c      time step will occur
      if (.not. import) then

         if (nlay .gt. 1) then

            if (dt .ne. ttr .or. tcalc .lt. told+dt) then

               do 70, ichn=1,nchn

                  told=tsav
                  tnew=tcalc
                  chnnum=ichn

                  call solvde(atol(ichn),arlold(1,ichn),ia(1,ichn),1,
     $               ja(1,ichn),neq(ichn),nnz(ichn),rtol(ichn),
     $               told,tnew)
```

```
70          continue

         else

            do 90, ichn=1,nchn

               do 80, ieq=1,neq(ichn)
                  ar1old(ieq,ichn)=ar1new(ieq,ichn)
80             continue

90          continue

         endif

      endif

   endif

   goto 55

95 continue

   write(status_msg(1:10),'(i10)',iostat=ioerr) period

       if (ioerr .ne. 0) then
           call ioretint(ioerr)
           return
       endif

   status_msg=status_msg(1:10)//
  $   ' timesteps were actually calculated'C
   call CMessage(status_msg,rc)


   end
```

F.86 RESOUT.FOR

```
c
c...this subroutine writes out the user sp· ified input data
c   for the building occupancy scenario
c
      subroutine resout(out,mxchn,adate,atime,short,long,title,
     $   histry,combin,import,distrb,confil,
     $   ar,cdg,cdi,cdo,diet,f1,f2,gr,h1,h2,
     $   i,ir,lambdw,mlf,mlg,mlh,mlv,n1,n2,nlay,pd,ps,qd,qf,qg,qh,qw,
     $   rf,rfr,rg,rh,rho1,rho2,rv,sfi,sfo,tbigf,tbigg,tbigh,tbigv,
     $   tca,tcv,td,tf,tff,tfg,tfh,tfw,tg,tgf,tgg,tgv,tgh,tha,
     $   thv,ti,ttg,ttr,tx,ua,uf,uv,uw,vdr,vg,vr,vsw,vx,
     $   wf,wg,wh,wv,xf,xg,xh,xw,yf,yg,yh,yv,
     $   fca,fcd,fcf,fcg,fch,satac,
     $   fha,fhd,fhf,fhg,fhh,fhv,satah,sasvh,sawvh,sh,
     $   tstart,tend,dt,skip,
     $   nchn,chain,conc,ioerr,nchnp2)

      character*6 short
      character*10 adate
      character*8 chain
      character*11 atime
      character*27 long
      character*80 confil
      character*80 title
      integer ichn,ioerr
      integer mxchn
      integer nchn,nchnp2,nlay
      integer out
      integer skip
      integer loc, loc1, loc2
      logical combin
      logical distrb
      logical equil
      logical histry
      logical import
      double precision ar
      double precision conc,cdg,cdi,cdo
      double precision diet,dt
      double precision f1,f2,fca,fcd,fcf,fcg,fch
      double precision fha,fhd,fhf,fhg,fhh,fhv
      double precision gr
      double precision h1,h2
      double precision i,ir
      double precision lambdw
      double precision mlf,mlg,mlh,mlv
      double precision n1,n2
      double precision pd,ps
      double precision qd,qf,qg,qh,qw
      double precision rf,rfr,rg,rh,rho1,rho2,rv
      double precision satac,satah,sasvh,sawvh,sfi,sfo,sh
      double precision tbigf,tbigg,tbigh,tbigv
      double precision tca,tcv,td,tf,tff,tfg,tfh,tfw
      double precision tg,tgf,tgg,tgv,tgh,tha,thv,ti,ttg,ttr,tx
      double precision tend,tstart
      double precision ua,uf,uv,uw
      double precision vdr,vg,vr,vsw,vx
      double precision wf,wg,wh,wv
```

```
      double precision xf,xg,xh,xw
      double precision yf,yg,yh,yv
      dimension chain(mxchn)
      dimension conc(mxchn)
      dimension fca(4),fcf(4),fcg(4),fch(4)
      dimension fha(4),fhf(4),fhg(4),fhh(4),fhv(4)
      dimension mlf(4),mlg(4),mlh(4),mlv(4)
      dimension qd(4),qf(4),qg(4),qh(4),qw(4)
      dimension rf(4),rg(4),rh(4),rv(4)
      dimension tbigf(4),tbigg(4),tbigh(4),tbigv(4)
      dimension tca(4),tcv(4),tff(4),tfg(4),tfh(4),tfw(4)
      dimension tgf(4),tgg(4),tgh(4),tgv(4),tha(4),thv(4)
      dimension ua(4),uv(4)
      dimension wf(4),wg(4),wh(4),wv(4)
      dimension xf(4),xg(4),xh(4),xw(4)
      dimension yf(4),yg(4),yh(4),yv(4)
c
      write(out,'(3a)',iostat=ioerr) 'Output From Program ''',short,''''
      if (ioerr .ne. 0) then
        call ioreturn(ioerr)
        return
      endif

      write(out,'(a)',iostat=ioerr) long
      if (ioerr .ne. 0) then
        call ioreturn(ioerr)
        return
      endif

      write(out,'(2a)',iostat=ioerr) 'Run Date: ',adate
      if (ioerr .ne. 0) then
        call ioreturn(ioerr)
        return
      endif

      write(out,'(2a)',iostat=ioerr) 'Run Time: ',atime
      if (ioerr .ne. 0) then
        call ioreturn(ioerr)
        return
      endif

      write(out,'(a)',iostat=ioerr) 'INPUT DATA:'
      if (ioerr .ne. 0) then
        call ioreturn(ioerr)
        return
      endif

      write(out,'(2a)',iostat=ioerr) 'Title: ',title
      if (ioerr .ne. 0) then
        call ioreturn(ioerr)
        return
      endif

      write(out,'(a)',iostat=ioerr) 'Notes:'
      if (ioerr .ne. 0) then
        call ioreturn(ioerr)
        return
      endif
```

```
 if (histry) then
    write(out,'(a)',iostat=ioerr) 'History file will be generated.'
    if (ioerr .ne. 0) then
     call ioreturn(ioerr)
     return
     endif

 else

    write(out,'(a)',iostat=ioerr)
$      'History file will not be generated.'
    if (ioerr .ne. 0) then
     call ioreturn(ioerr)
     return
    endif

 endif

 if (combin) then

    write(out,'(2a)',iostat=ioerr) 'Implicit progeny doses will',
$      ' be included with explicit parent.'
    if (ioerr .ne. 0) then
     call ioreturn(ioerr)
     return
    endif

 else

    write(out,'(2a)',iostat=ioerr)
$      'Implicit progeny doses will not',
$      ' be included with explicit parent.'
    if (ioerr .ne. 0) then
     call ioreturn(ioerr)
     return
    endif
 endif

 if (import) then

    write(out,'(4a)',iostat=ioerr)
$      'Concentration data will be read',
$      ' from file: ',confil,'.'
    if (ioerr .ne. 0) then
     call ioreturn(ioerr)
     return
    endif

 else

    write(out,'(a)',iostat=ioerr)
$      'Concentration data will be calculated.'
    if (ioerr .ne. 0) then
     call ioreturn(ioerr)
     return
    endif

    equil=.false.
    do 10, ichn=1,nchn
```

```
         if (chain(ichn) .eq. 'U_Nat') then
            loc=0
         elseif (chain(ichn) .eq. 'Th_Nat') then
            loc=6
         else
            loc1=max(index(chain(ichn),'+C'),index(chain(ichn),'+c'))
            loc2=max(index(chain(ichn),'_C'),index(chain(ichn),'_c'))
            loc=max(loc1,loc2)
         endif

         if (loc .gt. 0) equil=.true.

10    continue

      if (equil) then
         write(out,'(2a))',iostat=ioerr)
  $         'Concentrations of radionuclides in ',
  $         'equilibrium chains will be'
         if (ioerr .ne. 0) then
           call ioreturn(ioerr)
           return
         endif

         if (distrb) then

            write(out,'(2a)',iostat=ioerr)
  $            '   distributed based on initial ',
  $            'concentration given for parent.'
               if (ioerr .ne. 0) then
                   call ioreturn(ioerr)
                   return
               endif

         else

            write(out,'(2a)',iostat=ioerr)
  $            '   equilibrated based on initial ',
  $            'concentration of parent.'
               if (ioerr .ne. 0) then
                   call ioreturn(ioerr)
                   return
               endif

         endif
      endif
   endif

   write(out,'(a)',iostat=ioerr) ' '
   if (ioerr .ne. 0) then
     call ioreturn(ioerr)
     return
   endif

   write(out,'(a)',iostat=ioerr) 'PARAMETER DATA:'
   if (ioerr .ne. 0) then
     call ioreturn(ioerr)
     return
   endif
```

```
 write(out,'(a,t60,a,1p,e10.4,a)',iostat=ioerr)
$   'Time Spent Indoors',': ',ti,' d/y'
 if (ioerr .ne. 0) then
   call ioreturn(ioerr)
   return
 endif

 write(out,'(a,t60,a,1p,e10.4,a)',iostat=ioerr)
$   'Time Spent Outdoors',': ',tx,' d/y'
 if (ioerr .ne. 0) then
   call ioreturn(ioerr)
   return
 endif

 write(out,'(a,t60,a,1p,e10.4,a)',iostat=ioerr)
$   'Time Spent Gardening',': ',tg,' d/y'
 if (ioerr .ne. 0) then
   call ioreturn(ioerr)
   return
 endif

 write(out,'(a,t60,a,1p,e10.4,a)',iostat=ioerr)
$   'Total Time in One Year Exposure Period',': ',ttr,' d'
 if (ioerr .ne. 0) then
   call ioreturn(ioerr)
   return
 endif

 write(out,'(a,t60,a,1p,e10.4,a)',iostat=ioerr)
$   'Total Time in Gardening Period',': ',ttg,' d'
 if (ioerr .ne. 0) then
   call ioreturn(ioerr)
   return
 endif

 write(out,'(a,t60,a,1p,e10.4,a)',iostat=ioerr)
$   'Weathering Rate for Activity Removal from Plants',
$   ': ',lambdw,' /d'
 if (ioerr .ne. 0) then
   call ioreturn(ioerr)
   return
 endif

 write(out,'(a,t60,a,1p,e10.4)',iostat=ioerr)
$   'Indoor Shielding Factor',': ',sfi
 if (ioerr .ne. 0) then
   call ioreturn(ioerr)
   return
 endif

 write(out,'(a,t60,a,1p,e10.4)',iostat=ioerr)
$   'Outdoor Shielding Factor',': ',sfo
 if (ioerr .ne. 0) then
   call ioreturn(ioerr)
   return
 endif

 write(out,'(a,t60,a,1p,e10.4,a)',iostat=ioerr)
$   'Floor Dust Loading',': ',pd,' g/m^2'
```

```
 if (ioerr .ne. 0) then
   call ioreturn(ioerr)
   return
 endif

 write(out,'(a,t60,a,1p,e10.4,a)',iostat=ioerr)
$   'Resuspension Factor for Indoor Dust',': ',rfr,' /m'
 if (ioerr .ne. 0) then
   call ioreturn(ioerr)
   return
 endif

 write(out,'(a,t60,a,1p,e10.4,a)',iostat=ioerr)
$   'Indoor Air-Dust Loading Factor',': ',cdi,' g/m^3'
 if (ioerr .ne. 0) then
   call ioreturn(ioerr)
   return
 endif

 write(out,'(a,t60,a,1p,e10.4,a)',iostat=ioerr)
$   'Outdoor Air-Dust Loading Factor',': ',cdo,' g/m^3'
 if (ioerr .ne. 0) then
   call ioreturn(ioerr)
   return
 endif

 write(out,'(a,t60,a,1p,e10.4,a)',iostat=ioerr)
$   'Gardening Air-Dust Loading Factor',': ',cdg,' g/m^3'
 if (ioerr .ne. 0) then
   call ioreturn(ioerr)
   return
 endif

 write(out,'(a,t60,a,1p,e10.4,a)',iostat=ioerr)
$   'Indoor Volumetric Breathing Rate',': ',vr,' m^3/h'
 if (ioerr .ne. 0) then
   call ioreturn(ioerr)
   return
 endif

 write(out,'(a,t60,a,1p,e10.4,a)',iostat=ioerr)
$   'Outdoor Volumetric Breathing Rate',': ',vx,' m^3/h'
 if (ioerr .ne. 0) then
   call ioreturn(ioerr)
   return
 endif

 write(out,'(a,t60,a,1p,e10.4,a)',iostat=ioerr)
$   'Gardening Volumetric Breathing Rate',': ',vg,' m^3/h'
 if (ioerr .ne. 0) then
   call ioreturn(ioerr)
   return
 endif

 write(out,'(a,t60,a,1p,e10.4,a)',iostat=ioerr)
$   'Soil Ingestion Transfer Rate',': ',gr,' g/d'
 if (ioerr .ne. 0) then
   call ioreturn(ioerr)
   return
```

```
 endif

 write(out,'(a,t60,a,1p,e10.4,a)',iostat=ioerr)
$   'Drinking Water Ingestion Rate',': ',uw,' L/d'
 if (ioerr .ne. 0) then
   call ioreturn(ioerr)
   return
 endif

 write(out,'(a,t60,a,1p,e10.4,a)',iostat=ioerr)
$   'Drinking Water Consumption Period',': ',td,' d'
 if (ioerr .ne. 0) then
   call ioreturn(ioerr)
   return
 endif

 write(out,'(a,t60,a,1p,e10.4,a)',iostat=ioerr)
$   'Surface Soil Layer Thickness',': ',h1,' m'
 if (ioerr .ne. 0) then
   call ioreturn(ioerr)
   return
 endif

 write(out,'(a,t60,a,1p,e10.4,a)',iostat=ioerr)
$   'Unsaturated Zone Layer Thickness',': ',h2,' m'
 if (ioerr .ne. 0) then
   call ioreturn(ioerr)
   return
 endif

 write(out,'(a,t60,a,1p,i2)',iostat=ioerr)
$   'Number of Unsaturated Zone Layers',': ',nlay
 if (ioerr .ne. 0) then
   call ioreturn(ioerr)
   return
 endif

 write(out,'(a,t60,a,1p,e10.4)',iostat=ioerr)
$   'Surface Soil Porosity',': ',n1
 if (ioerr .ne. 0) then
   call ioreturn(ioerr)
   return
 endif

 write(out,'(a,t60,a,1p,e10.4)',iostat=ioerr)
$   'Unsaturated Zone Porosity',': ',n2
 if (ioerr .ne. 0) then
   call ioreturn(ioerr)
   return
 endif

 write(out,'(a,t60,a,1p,e10.4)',iostat=ioerr)
$   'Surface Soil Saturation',': ',f1
 if (ioerr .ne. 0) then
   call ioreturn(ioerr)
   return
 endif

 write(out,'(a,t60,a,1p,e10.4)',iostat=ioerr)
```

```
$   'Unsaturated Zone Saturation',': ',f2
 if (ioerr .ne. 0) then
   call ioreturn(ioerr)
   return
 endif

 write(out,'(a,t60,a,1p,e10.4,a)',iostat=ioerr)
$   'Surface Soil Bulk Soil Density',': ',rho1,' g/cm^3'
 if (ioerr .ne. 0) then
   call ioreturn(ioerr)
   return
 endif

 write(out,'(a,t60,a,1p,e10.4,a)',iostat=ioerr)
$   'Unsaturated Zone Bulk Soil Density',': ',rho2,' g/cm^3'
 if (ioerr .ne. 0) then
   call ioreturn(ioerr)
   return
 endif

 write(out,'(a,t60,a,1p,e10.4,a)',iostat=ioerr)
$   'Water Volume Removed From Aquifer Per Year',': ',vdr,' L'
 if (ioerr .ne. 0) then
   call ioreturn(ioerr)
   return
 endif

 write(out,'(a,t60,a,1p,e10.4,a)',iostat=ioerr)
$   'Volume of Surface Water Pond',': ',vsw,' L'
 if (ioerr .ne. 0) then
   call ioreturn(ioerr)
   return
 endif

 write(out,'(a,t60,a,1p,e10.4,a)',iostat=ioerr)
$   'Infiltration Rate',': ',i,' m/y'
 if (ioerr .ne. 0) then
   call ioreturn(ioerr)
   return
 endif

 write(out,'(a,t60,a,1p,e10.4,a)',iostat=ioerr)
$   'Area of Cultivated Land',': ',ar,' m^2'
 if (ioerr .ne. 0) then
   call ioreturn(ioerr)
   return
 endif

 write(out,'(a,t60,a,1p,e10.4,a)',iostat=ioerr)
$   'Irrigation Water Application Rate',': ',ir,' L/m^2-d'
 if (ioerr .ne. 0) then
   call ioreturn(ioerr)
   return
 endif

 write(out,'(a,t60,a,1p,e10.4,a)',iostat=ioerr)
$   'Soil Areal Density of Surface Plow Layer',': ',ps,' kg/m^2'
 if (ioerr .ne. 0) then
   call ioreturn(ioerr)
```

```
      return
    endif

    write(out,'(a,t60,a,1p,e10.4)',iostat=ioerr)
   $   'Fraction of Diet From Garden',': ',diet
    if (ioerr .ne. 0) then
      call ioreturn(ioerr)
      return
    endif

    write(out,'(a,t60,a,1p,e10.4,a)',iostat=ioerr)
   $   'Human Diet of Leafy Vegetables',': ',uv(1),' kg/y'
    if (ioerr .ne. 0) then
      call ioreturn(ioerr)
      return
    endif

    write(out,'(a,t60,a,1p,e10.4,a)',iostat=ioerr)
   $   'Human Diet of Other Vegetables',': ',uv(2),' kg/y'
    if (ioerr .ne. 0) then
      call ioreturn(ioerr)
      return
    endif

    write(out,'(a,t60,a,1p,e10.4,a)',iostat=ioerr)
   $   'Human Diet of Fruits',': ',uv(3),' kg/y'
    if (ioerr .ne. 0) then
      call ioreturn(ioerr)
      return
    endif

    write(out,'(a,t60,a,1p,e10.4,a)',iostat=ioerr)
   $   'Human Diet of Grains',': ',uv(4),' kg/y'
    if (ioerr .ne. 0) then
      call ioreturn(ioerr)
      return
    endif

    write(out,'(a,t60,a,1p,e10.4,a)',iostat=ioerr)
   $   'Human Diet of Beef',': ',ua(1),' kg/y'
    if (ioerr .ne. 0) then
      call ioreturn(ioerr)
      return
    endif

    write(out,'(a,t60,a,1p,e10.4,a)',iostat=ioerr)
   $   'Human Diet of Poultry',': ',ua(2),' kg/y'
    if (ioerr .ne. 0) then
      call ioreturn(ioerr)
      return
    endif

    write(out,'(a,t60,a,1p,e10.4,a)',iostat=ioerr)
   $   'Human Diet of Milk',': ',ua(3),' L/y'
    if (ioerr .ne. 0) then
      call ioreturn(ioerr)
      return
    endif
```

```
 write(out,'(a,t60,a,1p,e10.4,a)',iostat=ioerr)
$   'Human Diet of Eggs',': ',ua(4),' kg/y'
 if (ioerr .ne. 0) then
   call ioreturn(ioerr)
   return
 endif

 write(out,'(a,t60,a,1p,e10.4,a)',iostat=ioerr)
$   'Human Diet of Fish',': ',uf,' kg/y'
 if (ioerr .ne. 0) then
   call ioreturn(ioerr)
   return
 endif

 write(out,'(a,t60,a,1p,e10.4,a)',iostat=ioerr)
$   'Consumption Period for Fish',': ',tf,' d'
 if (ioerr .ne. 0) then
   call ioreturn(ioerr)
   return
 endif

 write(out,'(a,t60,a,1p,e10.4,a)',iostat=ioerr)
$   'Food Consumption Period for Leafy Vegetables',': ',tcv(1),' d'
 if (ioerr .ne. 0) then
   call ioreturn(ioerr)
   return
 endif

 write(out,'(a,t60,a,1p,e10.4,a)',iostat=ioerr)
$   'Food Consumption Period for Other Vegetables',': ',tcv(2),' d'
 if (ioerr .ne. 0) then
   call ioreturn(ioerr)
   return
 endif

 write(out,'(a,t60,a,1p,e10.4,a)',iostat=ioerr)
$   'Food Consumption Period for Fruits',': ',tcv(3),' d'
 if (ioerr .ne. 0) then
   call ioreturn(ioerr)
   return
 endif

 write(out,'(a,t60,a,1p,e10.4,a)',iostat=ioerr)
$   'Food Consumption Period for Grains',': ',tcv(4),' d'
 if (ioerr .ne. 0) then
   call ioreturn(ioerr)
   return
 endif

 write(out,'(a,t60,a,1p,e10.4,a)',iostat=ioerr)
$   'Food Consumption Period for Beef',': ',tca(1),' d'
 if (ioerr .ne. 0) then
   call ioreturn(ioerr)
   return
 endif

 write(out,'(a,t60,a,1p,e10.4,a)',iostat=ioerr)
$   'Food Consumption Period for Poultry',': ',tca(2),' d'
 if (ioerr .ne. 0) then
```

```
      call ioreturn(ioerr)
      return
    endif

    write(out,'(a,t60,a,1p,e10.4,a)',iostat=ioerr)
$   'Food Consumption Period for Milk',': ',tca(3),' d'
    if (ioerr .ne. 0) then
      call ioreturn(ioerr)
      return
    endif

    write(out,'(a,t60,a,1p,e10.4,a)',iostat=ioerr)
$   'Food Consumption Period for Eggs',': ',tca(4),' d'
    if (ioerr .ne. 0) then
      call ioreturn(ioerr)
      return
    endif

    write(out,'(a,t60,a,1p,e10.4,a)',iostat=ioerr)
$   'Holdup Period For Leafy Vegetables',': ',thv(1),' d'
    if (ioerr .ne. 0) then
      call ioreturn(ioerr)
      return
    endif

    write(out,'(a,t60,a,1p,e10.4,a)',iostat=ioerr)
$   'Holdup Period For Other Vegetables',': ',thv(2),' d'
    if (ioerr .ne. 0) then
      call ioreturn(ioerr)
      return
    endif

    write(out,'(a,t60,a,1p,e10.4,a)',iostat=ioerr)
$   'Holdup Period For Fruits',': ',thv(3),' d'
    if (ioerr .ne. 0) then
      call ioreturn(ioerr)
      return
    endif

    write(out,'(a,t60,a,1p,e10.4,a)',iostat=ioerr)
$   'Holdup Period For Grains',': ',thv(4),' d'
    if (ioerr .ne. 0) then
      call ioreturn(ioerr)
      return
    endif

    write(out,'(a,t60,a,1p,e10.4,a)',iostat=ioerr)
$   'Holdup Period For Beef',': ',tha(1),' d'
    if (ioerr .ne. 0) then
      call ioreturn(ioerr)
      return
    endif

    write(out,'(a,t60,a,1p,e10.4,a)',iostat=ioerr)
$   'Holdup Period For Poultry',': ',tha(2),' d'
    if (ioerr .ne. 0) then
      call ioreturn(ioerr)
      return
    endif
```

```
 write(out,'(a,t60,a,1p,e10.4,a)',iostat=ioerr)
$   'Holdup Period For Milk',': ',tha(3),' d'
 if (ioerr .ne. 0) then
   call ioreturn(ioerr)
   return
 endif

 write(out,'(a,t60,a,1p,e10.4,a)',iostat=ioerr)
$   'Holdup Period For Eggs',': ',tha(4),' d'
 if (ioerr .ne. 0) then
   call ioreturn(ioerr)
   return
 endif

 write(out,'(a,t60,a,1p,e10.4,a)',iostat=ioerr)
$   'Minimum Growing Period For Leafy Vegetables',': ',tgv(1),' d'
 if (ioerr .ne. 0) then
   call ioreturn(ioerr)
   return
 endif

 write(out,'(a,t60,a,1p,e10.4,a)',iostat=ioerr)
$   'Minimum Growing Period For Other Vegetables',': ',tgv(2),' d'
 if (ioerr .ne. 0) then
   call ioreturn(ioerr)
   return
 endif

 write(out,'(a,t60,a,1p,e10.4,a)',iostat=ioerr)
$   'Minimum Growing Period For Fruits',': ',tgv(3),' d'
 if (ioerr .ne. 0) then
   call ioreturn(ioerr)
   return
 endif

 write(out,'(a,t60,a,1p,e10.4,a)',iostat=ioerr)
$   'Minimum Growing Period For Grains',': ',tgv(4),' d'
 if (ioerr .ne. 0) then
   call ioreturn(ioerr)
   return
 endif

 write(out,'(a,t60,a,1p,e10.4,a)',iostat=ioerr)
$   'Minimum Growing Period For Beef Forage',': ',tgf(1),' d'
 if (ioerr .ne. 0) then
   call ioreturn(ioerr)
   return
 endif

 write(out,'(a,t60,a,1p,e10.4,a)',iostat=ioerr)
$   'Minimum Growing Period For Poultry Forage',': ',tgf(2),' d'
 if (ioerr .ne. 0) then
   call ioreturn(ioerr)
   return
 endif

 write(out,'(a,t60,a,1p,e10.4,a)',iostat=ioerr)
$   'Minimum Growing Period For Milk Cow Forage',': ',tgf(3),' d'
 if (ioerr .ne. 0) then
```

```
      call ioreturn(ioerr)
      return
    endif

    write(out,'(a,t60,a,1p,e10.4,a)',iostat=ioerr)
     $   'Minimum Growing Period For Layer Hen Forage',': ',tgf(4),' d'
    if (ioerr .ne. 0) then
      call ioreturn(ioerr)
      return
    endif

    write(out,'(a,t60,a,1p,e10.4,a)',iostat=ioerr)
     $   'Minimum Growing Period For Beef Grain',': ',tgg(1),' d'
    if (ioerr .ne. 0) then
      call ioreturn(ioerr)
      return
    endif

    write(out,'(a,t60,a,1p,e10.4,a)',iostat=ioerr)
     $   'Minimum Growing Period For Poultry Grain',': ',tgg(2),' d'
    if (ioerr .ne. 0) then
      call ioreturn(ioerr)
      return
    endif

    write(out,'(a,t60,a,1p,e10.4,a)',iostat=ioerr)
     $   'Minimum Growing Period For Milk Cow Grain',': ',tgg(3),' d'
    if (ioerr .ne. 0) then
      call ioreturn(ioerr)
      return
    endif

    write(out,'(a,t60,a,1p,e10.4,a)',iostat=ioerr)
     $   'Minimum Growing Period For Layer Hen Grain',': ',tgg(4),' d'
    if (ioerr .ne. 0) then
      call ioreturn(ioerr)
      return
    endif

    write(out,'(a,t60,a,1p,e10.4,a)',iostat=ioerr)
     $   'Minimum Growing Period For Beef Hay',': ',tgh(1),' d'
    if (ioerr .ne. 0) then
      call ioreturn(ioerr)
      return
    endif

    write(out,'(a,t60,a,1p,e10.4,a)',iostat=ioerr)
     $   'Minimum Growing Period For Poultry Hay',': ',tgh(2),' d'
    if (ioerr .ne. 0) then
      call ioreturn(ioerr)
      return
    endif

    write(out,'(a,t60,a,1p,e10.4,a)',iostat=ioerr)
     $   'Minimum Growing Period For Milk Cow Hay',': ',tgh(3),' d'
    if (ioerr .ne. 0) then
      call ioreturn(ioerr)
      return
    endif
```

```
 write(out,'(a,t60,a,1p,e10.4,a)',iostat=ioerr)
$   'Minimum Growing Period For Layer Hen Hay',': ',tgh(4),' d'
 if (ioerr .ne. 0) then
   call ioreturn(ioerr)
   return
 endif

 write(out,'(a,t60,a,1p,e10.4,a)',iostat=ioerr)
$   'Time Period For Feeding Forage to Beef Cattle',
$   ': ',tff(1),' d'
 if (ioerr .ne. 0) then
   call ioreturn(ioerr)
   return
 endif

 write(out,'(a,t60,a,1p,e10.4,a)',iostat=ioerr)
$   'Time Period For Feeding Forage to Poultry',
$   ': ',tff(2),' d'
 if (ioerr .ne. 0) then
   call ioreturn(ioerr)
   return
 endif

 write(out,'(a,t60,a,1p,e10.4,a)',iostat=ioerr)
$   'Time Period For Feeding Forage to Milk Cattle',
$   ': ',tff(3),' d'
 if (ioerr .ne. 0) then
   call ioreturn(ioerr)
   return
 endif

 write(out,'(a,t60,a,1p,e10.4,a)',iostat=ioerr)
$   'Time Period For Feeding Forage to Layer Hens',
$   ': ',tff(4),' d'
 if (ioerr .ne. 0) then
   call ioreturn(ioerr)
   return
 endif

 write(out,'(a,t60,a,1p,e10.4,a)',iostat=ioerr)
$   'Time Period For Feeding Grain to Beef Cattle',
$   ': ',tfg(1),' d'
 if (ioerr .ne. 0) then
   call ioreturn(ioerr)
   return
 endif

 write(out,'(a,t60,a,1p,e10.4,a)',iostat=ioerr)
$   'Time Period For Feeding Grain to Poultry',
$   ': ',tfg(2),' d'
 if (ioerr .ne. 0) then
   call ioreturn(ioerr)
   return
 endif

 write(out,'(a,t60,a,1p,e10.4,a)',iostat=ioerr)
$   'Time Period For Feeding Grain to Milk Cattle',
$   ': ',tfg(3),' d'
 if (ioerr .ne. 0) then
```

```
      call ioreturn(ioerr)
      return
    endif

    write(out,'(a,t60,a,1p,e10.4,a)',iostat=ioerr)
$     'Time Period For Feeding Grain to Layer Hens',
$     ': ',tfg(4),' d'
    if (ioerr .ne. 0) then
      call ioreturn(ioerr)
      return
    endif

    write(out,'(a,t60,a,1p,e10.4,a)',iostat=ioerr)
$     'Time Period For Feeding Hay to Beef Cattle',
$     ': ',tfh(1),' d'
    if (ioerr .ne. 0) then
      call ioreturn(ioerr)
      return
    endif

    write(out,'(a,t60,a,1p,e10.4,a)',iostat=ioerr)
$     'Time Period For Feeding Hay to Poultry',
$     ': ',tfh(2),' d'
    if (ioerr .ne. 0) then
      call ioreturn(ioerr)
      return
    endif

    write(out,'(a,t60,a,1p,e10.4,a)',iostat=ioerr)
$     'Time Period For Feeding Hay to Milk Cattle',
$     ': ',tfh(3),' d'
    if (ioerr .ne. 0) then
      call ioreturn(ioerr)
      return
    endif

    write(out,'(a,t60,a,1p,e10.4,a)',iostat=ioerr)
$     'Time Period For Feeding Hay to Layer Hens',
$     ': ',tfh(4),' d'
    if (ioerr .ne. 0) then
      call ioreturn(ioerr)
      return
    endif

    write(out,'(a,t60,a,1p,e10.4)',iostat=ioerr)
$     'Interception Fraction For Leafy Vegetables',': ',rv(1)
    if (ioerr .ne. 0) then
      call ioreturn(ioerr)
      return
    endif

    write(out,'(a,t60,a,1p,e10.4)',iostat=ioerr)
$     'Interception Fraction For Other Vegetables',': ',rv(2)
    if (ioerr .ne. 0) then
      call ioreturn(ioerr)
      return
    endif

    write(out,'(a,t60,a,1p,e10.4)',iostat=ioerr)
```

```
$   'Interception Fraction For Fruits',': ',rv(3)
 if (ioerr .ne. 0) then
   call ioreturn(ioerr)
   return
 endif

 write(out,'(a,t60,a,1p,e10.4)',iostat=ioerr)
$   'Interception Fraction For Grains',': ',rv(4)
 if (ioerr .ne. 0) then
   call ioreturn(ioerr)
   return
 endif

 write(out,'(a,t60,a,1p,e10.4)',iostat=ioerr)
$   'Interception Fraction For Beef Forage',': ',rf(1)
 if (ioerr .ne. 0) then
   call ioreturn(ioerr)
   return
 endif

 write(out,'(a,t60,a,1p,e10.4)',iostat=ioerr)
$   'Interception Fraction For Poultry Forage',': ',rf(2)
 if (ioerr .ne. 0) then
   call ioreturn(ioerr)
   return
 endif

 write(out,'(a,t60,a,1p,e10.4)',iostat=ioerr)
$   'Interception Fraction For Milk Cow Forage',': ',rf(3)
 if (ioerr .ne. 0) then
   call ioreturn(ioerr)
   return
 endif

 write(out,'(a,t60,a,1p,e10.4)',iostat=ioerr)
$   'Interception Fraction For Layer Hen Forage',': ',rf(4)
 if (ioerr .ne. 0) then
   call ioreturn(ioerr)
   return
 endif

 write(out,'(a,t60,a,1p,e10.4)',iostat=ioerr)
$   'Interception Fraction For Beef Grain',': ',rg(1)
 if (ioerr .ne. 0) then
   call ioreturn(ioerr)
   return
 endif

 write(out,'(a,t60,a,1p,e10.4)',iostat=ioerr)
$   'Interception Fraction For Poultry Grain',': ',rg(2)
 if (ioerr .ne. 0) then
   call ioreturn(ioerr)
   return
 endif

 write(out,'(a,t60,a,1p,e10.4)',iostat=ioerr)
$   'Interception Fraction For Milk Cow Grain',': ',rg(3)
 if (ioerr .ne. 0) then
   call ioreturn(ioerr)
```

```
      return
    endif

    write(out,'(a,t60,a,1p,e10.4)',iostat=ioerr)
$   'Interception Fraction For Layer Hen Grain',': ',rg(4)
    if (ioerr .ne. 0) then
      call ioreturn(ioerr)
      return
    endif

    write(out,'(a,t60,a,1p,e10.4)',iostat=ioerr)
$   'Interception Fraction For Beef Hay',': ',rh(1)
    if (ioerr .ne. 0) then
      call ioreturn(ioerr)
      return
    endif

    write(out,'(a,t60,a,1p,e10.4)',iostat=ioerr)
$   'Interception Fraction For Poultry Hay',': ',rh(2)
    if (ioerr .ne. 0) then
      call ioreturn(ioerr)
      return
    endif

    write(out,'(a,t60,a,1p,e10.4)',iostat=ioerr)
$   'Interception Fraction For Milk Cow Hay',': ',rh(3)
    if (ioerr .ne. 0) then
      call ioreturn(ioerr)
      return
    endif

    write(out,'(a,t60,a,1p,e10.4)',iostat=ioerr)
$   'Interception Fraction For Layer Hen Hay',': ',rh(4)
    if (ioerr .ne. 0) then
      call ioreturn(ioerr)
      return
    endif

    write(out,'(a,t60,a,1p,e10.4)',iostat=ioerr)
$   'Translocation Factor For Leafy Vegetables',': ',tbigv(1)
    if (ioerr .ne. 0) then
      call ioreturn(ioerr)
      return
    endif

    write(out,'(a,t60,a,1p,e10.4)',iostat=ioerr)
$   'Translocation Factor For Other Vegetables',': ',tbigv(2)
    if (ioerr .ne. 0) then
      call ioreturn(ioerr)
      return
    endif

    write(out,'(a,t60,a,1p,e10.4)',iostat=ioerr)
$   'Translocation Factor For Fruits',': ',tbigv(3)
    if (ioerr .ne. 0) then
      call ioreturn(ioerr)
      return
    endif
```

```
 write(out,'(a,t60,a,1p,e10.4)',iostat=ioerr)
$   'Translocation Factor For Grains',': ',tbigv(4)
 if (ioerr .ne. 0) then
   call ioreturn(ioerr)
   return
 endif

 write(out,'(a,t60,a,1p,e10.4)',iostat=ioerr)
$   'Translocation Factor For Beef Forage',': ',tbigf(1)
 if (ioerr .ne. 0) then
   call ioreturn(ioerr)
   return
 endif

 write(out,'(a,t60,a,1p,e10.4)',iostat=ioerr)
$   'Translocation Factor For Poultry Forage',': ',tbigf(2)
 if (ioerr .ne. 0) then
   call ioreturn(ioerr)
   return
 endif

 write(out,'(a,t60,a,1p,e10.4)',iostat=ioerr)
$   'Translocation Factor For Milk Cow Forage',': ',tbigf(3)
 if (ioerr .ne. 0) then
   call ioreturn(ioerr)
   return
 endif

 write(out,'(a,t60,a,1p,e10.4)',iostat=ioerr)
$   'Translocation Factor For Layer Hen Forage',': ',tbigf(4)
 if (ioerr .ne. 0) then
   call ioreturn(ioerr)
   return
 endif

 write(out,'(a,t60,a,1p,e10.4)',iostat=ioerr)
$   'Translocation Factor For Beef Grain',': ',tbigg(1)
 if (ioerr .ne. 0) then
   call ioreturn(ioerr)
   return
 endif

 write(out,'(a,t60,a,1p,e10.4)',iostat=ioerr)
$   'Translocation Factor For Poultry Grain',': ',tbigg(2)
 if (ioerr .ne. 0) then
   call ioreturn(ioerr)
   return
 endif

 write(out,'(a,t60,a,1p,e10.4)',iostat=ioerr)
$   'Translocation Factor For Milk Cow Grain',': ',tbigg(3)
 if (ioerr .ne. 0) then
   call ioreturn(ioerr)
   return
 endif

 write(out,'(a,t60,a,1p,e10.4)',iostat=ioerr)
$   'Translocation Factor For Layer Hen Grain',': ',tbigg(4)
 if (ioerr .ne. 0) then
```

```
      call ioreturn(ioerr)
      return
    endif

    write(out,'(a,t60,a,1p,e10.4)',iostat=ioerr)
$   'Translocation Factor For Beef Hay',': ',tbigh(1)
    if (ioerr .ne. 0) then
      call ioreturn(ioerr)
      return
    endif

    write(out,'(a,t60,a,1p,e10.4)',iostat=ioerr)
$   'Translocation Factor For Poultry Hay',': ',tbigh(2)
    if (ioerr .ne. 0) then
      call ioreturn(ioerr)
      return
    endif

    write(out,'(a,t60,a,1p,e10.4)',iostat=ioerr)
$   'Translocation Factor For Milk Cow Hay',': ',tbigh(3)
    if (ioerr .ne. 0) then
      call ioreturn(ioerr)
      return
    endif

    write(out,'(a,t60,a,1p,e10.4)',iostat=ioerr)
$   'Translocation Factor For Layer Hen Hay',': ',tbigh(4)
    if (ioerr .ne. 0) then
      call ioreturn(ioerr)
      return
    endif

    write(out,'(a,t60,a,1p,e10.4)',iostat=ioerr)
$   'Forage Fraction Consumed By Beef Cattle',': ',xf(1)
    if (ioerr .ne. 0) then
      call ioreturn(ioerr)
      return
    endif

    write(out,'(a,t60,a,1p,e10.4)',iostat=ioerr)
$   'Forage Fraction Consumed By Poultry',': ',xf(2)
    if (ioerr .ne. 0) then
      call ioreturn(ioerr)
      return
    endif

    write(out,'(a,t60,a,1p,e10.4)',iostat=ioerr)
$   'Forage Fraction Consumed By Milk Cows',': ',xf(3)
    if (ioerr .ne. 0) then
      call ioreturn(ioerr)
      return
    endif

    write(out,'(a,t60,a,1p,e10.4)',iostat=ioerr)
$   'Forage Fraction Consumed By Layer Hens',': ',xf(4)
    if (ioerr .ne. 0) then
      call ioreturn(ioerr)
      return
    endif
```

```
 write(out,'(a,t60,a,1p,e10.4)',iostat=ioerr)
$   'Grain  Fraction Consumed By Beef Cattle',': ',xg(1)
 if (ioerr .ne. 0) then
   call ioreturn(ioerr)
   return
 endif

 write(out,'(a,t60,a,1p,e10.4)',iostat=ioerr)
$   'Grain Fraction Consumed By Poultry',': ',xg(2)
 if (ioerr .ne. 0) then
   call ioreturn(ioerr)
   return
 endif

 write(out,'(a,t60,a,1p,e10.4)',iostat=ioerr)
$   'Grain Fraction Consumed By Milk Cows',': ',xg(3)
 if (ioerr .ne. 0) then
   call ioreturn(ioerr)
   return
 endif

 write(out,'(a,t60,a,1p,e10.4)',iostat=ioerr)
$   'Grain Fraction Consumed By Layer Hens',': ',xg(4)
 if (ioerr .ne. 0) then
   call ioreturn(ioerr)
   return
 endif

 write(out,'(a,t60,a,1p,e10.4)',iostat=ioerr)
$   'Hay Fraction Consumed By Beef Cattle',': ',xh(1)
 if (ioerr .ne. 0) then
   call ioreturn(ioerr)
   return
 endif

 write(out,'(a,t60,a,1p,e10.4)',iostat=ioerr)
$   'Hay Fraction Consumed By Poultry',': ',xh(2)
 if (ioerr .ne. 0) then
   call ioreturn(ioerr)
   return
 endif

 write(out,'(a,t60,a,1p,e10.4)',iostat=ioerr)
$   'Hay Fraction Consumed By Milk Cows',': ',xh(3)
 if (ioerr .ne. 0) then
   call ioreturn(ioerr)
   return
 endif

 write(out,'(a,t60,a,1p,e10.4)',iostat=ioerr)
$   'Hay Fraction Consumed By Layer Hens',': ',xh(4)
 if (ioerr .ne. 0) then
   call ioreturn(ioerr)
   return
 endif

 write(out,'(a,t60,a,1p,e10.4)',iostat=ioerr)
$   'Water Fraction Consumed By Beef Cattle',': ',xw(1)
 if (ioerr .ne. 0) then
```

```
      call ioreturn(ioerr)
      return
    endif

    write(out,'(a,t60,a,1p,e10.4)',iostat=ioerr)
   $   'Water Fraction Consumed By Poultry',': ',xw(2)
    if (ioerr .ne. 0) then
      call ioreturn(ioerr)
      return
    endif

    write(out,'(a,t60,a,1p,e10.4)',iostat=ioerr)
   $   'Water Fraction Consumed By Milk Cows',': ',xw(3)
    if (ioerr .ne. 0) then
      call ioreturn(ioerr)
      return
    endif

    write(out,'(a,t60,a,1p,e10.4)',iostat=ioerr)
   $   'Water Fraction Consumed By Layer Hens',': ',xw(4)
    if (ioerr .ne. 0) then
      call ioreturn(ioerr)
      return
    endif

    write(out,'(a,t60,a,1p,e10.4,a)',iostat=ioerr)
   $   'Crop Yield For Leafy Vegetables',': ',yv(1),' kg/m^2'
    if (ioerr .ne. 0) then
      call ioreturn(ioerr)
      return
    endif

    write(out,'(a,t60,a,1p,e10.4,a)',iostat=ioerr)
   $   'Crop Yield For Other Vegetables',': ',yv(2),' kg/m^2'
    if (ioerr .ne. 0) then
      call ioreturn(ioerr)
      return
    endif

    write(out,'(a,t60,a,1p,e10.4,a)',iostat=ioerr)
   $   'Crop Yield For Fruits',': ',yv(3),' kg/m^2'
    if  ioerr .ne. 0) then
      call ioreturn(ioerr)
      return
    endif

    write(out,'(a,t60,a,1p,e10.4,a)',iostat=ioerr)
   $   'Crop Yield For Grains',': ',yv(4),' kg/m^2'
    if (ioerr .ne. 0) then
      call ioreturn(ioerr)
      return
    endif

    write(out,'(a,t60,a,1p,e10.4,a)',iostat=ioerr)
   $   'Crop Yield For Beef Cattle Forage',': ',yf(1),' kg/m^2'
    if (ioerr .ne. 0) then
      call ioreturn(ioerr)
      return
    endif
```

```
 write(out,'(a,t60,a,1p,e10.4,a)',iostat=ioerr)
$   'Crop Yield For Poultry Forage',': ',yf(2),' kg/m^2'
 if (ioerr .ne. 0) then
   call ioreturn(ioerr)
   return
 endif

 write(out,'(a,t60,a,1p,e10.4,a)',iostat=ioerr)
$   'Crop Yield For Milk Cow Forage',': ',yf(3),' kg/m^2'
 if (ioerr .ne. 0) then
   call ioreturn(ioerr)
   return
 endif

 write(out,'(a,t60,a,1p,e10.4,a)',iostat=ioerr)
$   'Crop Yield For Layer Hen Forage',': ',yf(4),' kg/m^2'
 if (ioerr .ne. 0) then
   call ioreturn(ioerr)
   return
 endif

 write(out,'(a,t60,a,1p,e10.4,a)',iostat=ioerr)
$   'Crop Yield For Beef Cattle Grain',': ',yg(1),' kg/m^2'
 if (ioerr .ne. 0) then
   call ioreturn(ioerr)
   return
 endif

 write(out,'(a,t60,a,1p,e10.4,a)',iostat=ioerr)
$   'Crop Yield For Poultry Grain',': ',yg(2),' kg/m^2'
 if (ioerr .ne. 0) then
   call ioreturn(ioerr)
   return
 endif

 write(out,'(a,t60,a,1p,e10.4,a)',iostat=ioerr)
$   'Crop Yield For Milk Cow Grain',': ',yg(3),' kg/m^2'
 if (ioerr .ne. 0) then
   call ioreturn(ioerr)
   return
 endif

 write(out,'(a,t60,a,1p,e10.4,a)',iostat=ioerr)
$   'Crop Yield For Layer Hen Grain',': ',yg(4),' kg/m^2'
 if (ioerr .ne. 0) then
   call ioreturn(ioerr)
   return
 endif

 write(out,'(a,t60,a,1p,e10.4,a)',iostat=ioerr)
$   'Crop Yield For Beef Cattle Hay',': ',yh(1),' kg/m^2'
 if (ioerr .ne. 0) then
   call ioreturn(ioerr)
   return
 endif

 write(out,'(a,t60,a,1p,e10.4,a)',iostat=ioerr)
$   'Crop Yield For Poultry Hay',': ',yh(2),' kg/m^2'
 if (ioerr .ne. 0) then
```

```
      call ioreturn(ioerr)
      return
    endif

    write(out,'(a,t60,a,1p,e10.4,a)',iostat=ioerr)
$   'Crop Yield For Milk Cow Hay',': ',yh(3),' kg/m^2'
    if (ioerr .ne. 0) then
      call ioreturn(ioerr)
      return
    endif

    write(out,'(a,t60,a,1p,e10.4,a)',iostat=ioerr)
$   'Crop Yield For Layer Hen Hay',': ',yh(4),' kg/m^2'
    if (ioerr .ne. 0) then
      call ioreturn(ioerr)
      return
    endif

    write(out,'(a,t60,a,1p,e10.4)',iostat=ioerr)
$   'Wet-to-Dry Weight Conversion Factor For Leafy Vegetables',
$   ': ',wv(1)
    if (ioerr .ne. 0) then
      call ioreturn(ioerr)
      return
    endif

    write(out,'(a,t60,a,1p,e10.4)',iostat=ioerr)
$   'Wet-to-Dry Weight Conversion Factor For Other Vegetables',
$   ': ',wv(2)
    if (ioerr .ne. 0) then
      call ioreturn(ioerr)
      return
    endif

    write(out,'(a,t60,a,1p,e10.4)',iostat=ioerr)
$   'Wet-to-Dry Weight Conversion Factor For Fruits',
$   ': ',wv(3)
    if (ioerr .ne. 0) then
      call ioreturn(ioerr)
      return
    endif

    write(out,'(a,t60,a,1p,e10.4)',iostat=ioerr)
$   'Wet-to-Dry Weight Conversion Factor For Grains',
$   ': ',wv(4)
    if (ioerr .ne. 0) then
      call ioreturn(ioerr)
      return
    endif

    write(out,'(a,t60,a,1p,e10.4)',iostat=ioerr)
$   'Wet-to-Dry Weight Conversion Factor For Beef Cattle Forage',
$   ': ',wf(1)
    if (ioerr .ne. 0) then
      call ioreturn(ioerr)
      return
    endif

    write(out,'(a,t60,a,1p,e10.4)',iostat=ioerr)
```

```
$   'Wet-to-Dry Weight Conversion Factor For Poultry Forage',
$   ': ',wf(2)
 if (ioerr .ne. 0) then
   call ioreturn(ioerr)
   return
 endif

 write(out,'(a,t60,a,1p,e10.4)',iostat=ioerr)
$   'Wet-to-Dry Weight Conversion Factor For Milk Cow Forage',
$   ': ',wf(3)
 if (ioerr .ne. 0) then
   call ioreturn(ioerr)
   return
 endif

 write(out,'(a,t60,a,1p,e10.4)',iostat=ioerr)
$   'Wet-to-Dry Weight Conversion Factor For Layer Hen Forage',
$   ': ',wf(4)
 if (ioerr .ne. 0) then
   call ioreturn(ioerr)
   return
 endif

 write(out,'(a,t60,a,1p,e10.4)',iostat=ioerr)
$   'Wet-to-Dry Weight Conversion Factor For Beef Cattle Grain',
$   ': ',wg(1)
 if (ioerr .ne. 0) then
   call ioreturn(ioerr)
   return
 endif

 write(out,'(a,t60,a,1p,e10.4)',iostat=ioerr)
$   'Wet-to-Dry Weight Conversion Factor For Poultry Grain',
$   ': ',wg(2)
 if (ioerr .ne. 0) then
   call ioreturn(ioerr)
   return
 endif

 write(out,'(a,t60,a,1p,e10.4)',iostat=ioerr)
$   'Wet-to-Dry Weight Conversion Factor For Milk Cow Grain',
$   ': ',wg(3)
 if (ioerr .ne. 0) then
   call ioreturn(ioerr)
   return
 endif

 write(out,'(a,t60,a,1p,e10.4)',iostat=ioerr)
$   'Wet-to-Dry Weight Conversion Factor For Layer Hen Grain',
$   ': ',wg(4)
 if (ioerr .ne. 0) then
   call ioreturn(ioerr)
   return
 endif

 write(out,'(a,t60,a,1p,e10.4)',iostat=ioerr)
$   'Wet-to-Dry Weight Conversion Factor For Beef Cattle Hay',
$   ': ',wh(1)
 if (ioerr .ne. 0) then
```

```
      call ioreturn(ioerr)
      return
    endif

    write(out,'(a,t60,a,1p,e10.4)',iostat=ioerr)
$     'Wet-to-Dry Weight Conversion Factor For Poultry Hay',
$     ': ',wh(2)
    if (ioerr .ne. 0) then
      call ioreturn(ioerr)
      return
    endif

    write(out,'(a,t60,a,1p,e10.4)',iostat=ioerr)
$     'Wet-to-Dry Weight Conversion Factor For Milk Cow Hay',
$     ': ',wh(3)
    if (ioerr .ne. 0) then
      call ioreturn(ioerr)
      return
    endif

    write(out,'(a,t60,a,1p,e10.4)',iostat=ioerr)
$     'Wet-to-Dry Weight Conversion Factor For Layer Hen Hay',
$     ': ',wh(4)
    if (ioerr .ne. 0) then
      call ioreturn(ioerr)
      return
    endif

    write(out,'(a,t60,a,1p,e10.4,a)',iostat=ioerr)
$     'Animal Feed Intake Rate For Forage Consumed By Beef Cattle',
$     ': ',qf(1),' kg/d'
    if (ioerr .ne. 0) then
      call ioreturn(ioerr)
      return
    endif

    write(out,'(a,t60,a,1p,e10.4,a)',iostat=ioerr)
$     'Animal Feed Intake Rate For Forage Consumed By Poultry',
$     ': ',qf(2),' kg/d'
    if (ioerr .ne. 0) then
      call ioreturn(ioerr)
      return
    endif

    write(out,'(a,t60,a,1p,e10.4,a)',iostat=ioerr)
$     'Animal Feed Intake Rate For Forage Consumed By Milk Cows',
$     ': ',qf(3),' kg/d'
    if (ioerr .ne. 0) then
      call ioreturn(ioerr)
      return
    endif

    write(out,'(a,t60,a,1p,e10.4,a)',iostat=ioerr)
$     'Animal Feed Intake Rate For Forage Consumed By Layer Hens',
$     ': ',qf(4),' kg/d'
    if (ioerr .ne. 0) then
      call ioreturn(ioerr)
      return
    endif
```

```
 write(out,'(a,t60,a,1p,e10.4,a)',iostat=ioerr)
$   'Animal Feed Intake Rate For Grain Consumed By Beef Cattle',
$   ': ',qg(1),' kg/d'
 if (ioerr .ne. 0) then
   call ioreturn(ioerr)
   return
 endif

 write(out,'(a,t60,a,1p,e10.4,a)',iostat=ioerr)
$   'Animal Feed Intake Rate For Grain Consumed By Poultry',
$   ': ',qg(2),' kg/d'
 if (ioerr .ne. 0) then
   call ioreturn(ioerr)
   return
 endif

 write(out,'(a,t60,a,1p,e10.4,a)',iostat=ioerr)
$   'Animal Feed Intake Rate For Grain Consumed By Milk Cows',
$   ': ',qg(3),' kg/d'
 if (ioerr .ne. 0) then
   call ioreturn(ioerr)
   return
 endif

 write(out,'(a,t60,a,1p,e10.4,a)',iostat=ioerr)
$   'Animal Feed Intake Rate For Grain Consumed By Layer Hens',
$   ': ',qg(4),' kg/d'
 if (ioerr .ne. 0) then
   call ioreturn(ioerr)
   return
 endif

 write(out,'(a,t60,a,1p,e10.4,a)',iostat=ioerr)
$   'Animal Feed Intake Rate For Hay Consumed By Beef Cattle',
$   ': ',qh(1),' kg/d'
 if (ioerr .ne. 0) then
   call ioreturn(ioerr)
   return
 endif

 write(out,'(a,t60,a,1p,e10.4,a)',iostat=ioerr)
$   'Animal Feed Intake Rate For Hay Consumed By Poultry',
$   ': ',qh(2),' kg/d'
 if (ioerr .ne. 0) then
   call ioreturn(ioerr)
   return
 endif

 write(out,'(a,t60,a,1p,e10.4,a)',iostat=ioerr)
$   'Animal Feed Intake Rate For Hay Consumed By Milk Cows',
$   ': ',qh(3),' kg/d'
 if (ioerr .ne. 0) then
   call ioreturn(ioerr)
   return
 endif

 write(out,'(a,t60,a,1p,e10.4,a)',iostat=ioerr)
$   'Animal Feed Intake Rate For Hay Consumed By Layer Hens',
$   ': ',qh(4),' kg/d'
```

```
 if (ioerr .ne. 0) then
   call ioreturn(ioerr)
   return
 endif

 write(out,'(a,t60,a,1p,e10.4,a)',iostat=ioerr)
$   'Water Intake Rate For Beef Cattle',': ',qw(1),' L/d'
 if (ioerr .ne. 0) then
   call ioreturn(ioerr)
   return
 endif

 write(out,'(a,t60,a,1p,e10.4,a)',iostat=ioerr)
$   'Water Intake Rate For Poultry',': ',qw(2),' L/d'
 if (ioerr .ne. 0) then
   call ioreturn(ioerr)
   return
 endif

 write(out,'(a,t60,a,1p,e10.4,a)',iostat=ioerr)
$   'Water Intake Rate For Milk Cows',': ',qw(3),' L/d'
 if (ioerr .ne. 0) then
   call ioreturn(ioerr)
   return
 endif

 write(out,'(a,t60,a,1p,e10.4,a)',iostat=ioerr)
$   'Water Intake Rate For Layer Hens',': ',qw(4),' L/d'
 if (ioerr .ne. 0) then
   call ioreturn(ioerr)
   return
 endif

 write(out,'(a,t60,a,1p,e10.4,a)',iostat=ioerr)
$   'Water Intake Consumption Period for Beef Cattle',': ',
$   tfw(1),' d'
 if (ioerr .ne. 0) then
   call ioreturn(ioerr)
   return
 endif

 write(out,'(a,t60,a,1p,e10.4,a)',iostat=ioerr)
$   'Water Intake Consumption Period for Poultry',': ',
$   tfw(2),' d'
 if (ioerr .ne. 0) then
   call ioreturn(ioerr)
   return
 endif

 write(out,'(a,t60,a,1p,e10.4,a)',iostat=ioerr)
$   'Water Intake Consumption Period for Milk Cows',': ',
$   tfw(3),' d'
 if (ioerr .ne. 0) then
   call ioreturn(ioerr)
   return
 endif

 write(out,'(a,t60,a,1p,e10.4,a)',iostat=ioerr)
$   'Water Intake Consumption Period for Layer Hens',': ',
```

```
$   tfw(4),' d'
 if (ioerr .ne. 0) then
   call ioreturn(ioerr)
   return
 endif

 write(out,'(a,t60,a,1p,e10.4)',iostat=ioerr)
$   'Soil Intake Fraction For Beef Cattle',': ',qd(1)
 if (ioerr .ne. 0) then
   call ioreturn(ioerr)
   return
 endif

 write(out,'(a,t60,a,1p,e10.4)',iostat=ioerr)
$   'Soil Intake Fraction For Poultry',': ',qd(2)
 if (ioerr .ne. 0) then
   call ioreturn(ioerr)
   return
 endif

 write(out,'(a,t60,a,1p,e10.4)',iostat=ioerr)
$   'Soil Intake Fraction For Milk Cows',': ',qd(3)
 if (ioerr .ne. 0) then
   call ioreturn(ioerr)
   return
 endif

 write(out,'(a,t60,a,1p,e10.4)',iostat=ioerr)
$   'Soil Intake Fraction For Layer Hens',': ',qd(4)
 if (ioerr .ne. 0) then
   call ioreturn(ioerr)
   return
 endif

 write(out,'(a,t60,a,1p,e10.4,a)',iostat=ioerr)
$   'Plant Mass Loading Factor For Leafy Vegetables',': ',
$   mlv(1),' g/g'
 if (ioerr .ne. 0) then
   call ioreturn(ioerr)
   return
 endif

 write(out,'(a,t60,a,1p,e10.4,a)',iostat=ioerr)
$   'Plant Mass Loading Factor For Other Vegetables',': ',
$   mlv(2),' g/g'
 if (ioerr .ne. 0) then
   call ioreturn(ioerr)
   return
 endif

 write(out,'(a,t60,a,1p,e10.4,a)',iostat=ioerr)
$   'Plant Mass Loading Factor For Fruits',': ',
$   mlv(3),' g/g'
 if (ioerr .ne. 0) then
   call ioreturn(ioerr)
   return
 endif

 write(out,'(a,t60,a,1p,e10.4,a)',iostat=ioerr)
```

```
$   'Plant Mass Loading Factor For Grains',': ',
$   mlv(4),' g/g'
 if (ioerr .ne. 0) then
   call ioreturn(ioerr)
   return
 endif

 write(out,'(a,t60,a,1p,e10.4,a)',iostat=ioerr)
$   'Plant Mass Loading Factor For Beef Cattle Forage',': ',
$   mlf(1),' g/g'
 if (ioerr .ne. 0) then
   call ioreturn(ioerr)
   return
 endif

 write(out,'(a,t60,a,1p,e10.4,a)',iostat=ioerr)
$   'Plant Mass Loading Factor For Poultry Forage',': ',
$   mlf(2),' g/g'
 if (ioerr .ne. 0) then
   call ioreturn(ioerr)
   return
 endif

 write(out,'(a,t60,a,1p,e10.4,a)',iostat=ioerr)
$   'Plant Mass Loading Factor For Milk Cow Forage',': ',
$   mlf(3),' g/g'
 if (ioerr .ne. 0) then
   call ioreturn(ioerr)
   return
 endif

 write(out,'(a,t60,a,1p,e10.4,a)',iostat=ioerr)
$   'Plant Mass Loading Factor For Layer Hen Forage',': ',
$   mlf(4),' g/g'
 if (ioerr .ne. 0) then
   call ioreturn(ioerr)
   return
 endif

 write(out,'(a,t60,a,1p,e10.4,a)',iostat=ioerr)
$   'Plant Mass Loading Factor For Beef Cattle Grain',': ',
$   mlg(1),' g/g'
 if (ioerr .ne. 0) then
   call ioreturn(ioerr)
   return
 endif

 write(out,'(a,t60,a,1p,e10.4,a)',iostat=ioerr)
$   'Plant Mass Loading Factor For Poultry Grain',': ',
$   mlg(2),' g/g'
 if (ioerr .ne. 0) then
   call ioreturn(ioerr)
   return
 endif

 write(out,'(a,t60,a,1p,e10.4,a)',iostat=ioerr)
$   'Plant Mass Loading Factor For Milk Cow Grain',': ',
$   mlg(3),' g/g'
 if (ioerr .ne. 0) then
```

```
    call ioreturn(ioerr)
    return
  endif

  write(out,'(a,t60,a,1p,e10.4,a)',iostat=ioerr)
$   'Plant Mass Loading Factor For Layer Hen Grain',': ',
$   mlg(4),' g/g'
  if (ioerr .ne. 0) then
    call ioreturn(ioerr)
    return
  endif

  write(out,'(a,t60,a,1p,e10.4,a)',iostat=ioerr)
$   'Plant Mass Loading Factor For Beef Cattle Hay',': ',
$   mlh(1),' g/g'
  if (ioerr .ne. 0) then
    call ioreturn(ioerr)
    return
  endif

  write(out,'(a,t60,a,1p,e10.4,a)',iostat=ioerr)
$   'Plant Mass Loading Factor For Poultry Hay',': ',
$   mlh(2),' g/g'
  if (ioerr .ne. 0) then
    call ioreturn(ioerr)
    return
  endif

  write(out,'(a,t60,a,1p,e10.4,a)',iostat=ioerr)
$   'Plant Mass Loading Factor For Milk Cow Hay',': ',
$   mlh(3),' g/g'
  if (ioerr .ne. 0) then
    call ioreturn(ioerr)
    return
  endif

  write(out,'(a,t60,a,1p,e10.4,a)',iostat=ioerr)
$   'Plant Mass Loading Factor For Layer Hen Hay',': ',
$   mlh(4),' g/g'
  if (ioerr .ne. 0) then
    call ioreturn(ioerr)
    return
  endif

  write(out,'(a,t60,a,1p,e10.4)',iostat=ioerr)
$   'Fraction of Carbon in Beef',': ',fca(1)
  if (ioerr .ne. 0) then
    call ioreturn(ioerr)
    return
  endif

  write(out,'(a,t60,a,1p,e10.4)',iostat=ioerr)
$   'Fraction of Carbon in Poultry',': ',fca(2)
  if (ioerr .ne. 0) then
    call ioreturn(ioerr)
    return
  endif

  write(out,'(a,t60,a,1p,e10.4)',iostat=ioerr)
```

```
$   'Fraction of Carbon in Milk',': ',fca(3)
 if (ioerr .ne. 0) then
   call ioreturn(ioerr)
   return
 endif

 write(out,'(a,t60,a,1p,e10.4)',iostat=ioerr)
$   'Fraction of Carbon in Eggs',': ',fca(4)
 if (ioerr .ne. 0) then
   call ioreturn(ioerr)
   return
 endif

 write(out,'(a,t60,a,1p,e10.4)',iostat=ioerr)
$   'Fraction of Carbon in Beef Cattle Forage',': ',fcf(1)
 if (ioerr .ne. 0) then
   call ioreturn(ioerr)
   return
 endif

 write(out,'(a,t60,a,1p,e10.4)',iostat=ioerr)
$   'Fraction of Carbon in Poultry Forage',': ',fcf(2)
 if (ioerr .ne. 0) then
   call ioreturn(ioerr)
   return
 endif

 write(out,'(a,t60,a,1p,e10.4)',iostat=ioerr)
$   'Fraction of Carbon in Milk Cow Forage',': ',fcf(3)
 if (ioerr .ne. 0) then
   call ioreturn(ioerr)
   return
 endif

 write(out,'(a,t60,a,1p,e10.4)',iostat=ioerr)
$   'Fraction of Carbon in Layer Hen Forage',': ',fcf(4)
 if (ioerr .ne. 0) then
   call ioreturn(ioerr)
   return
 endif

 write(out,'(a,t60,a,1p,e10.4)',iostat=ioerr)
$   'Fraction of Carbon in Beef Cattle Grain',': ',fcg(1)
 if (ioerr .ne. 0) then
   call ioreturn(ioerr)
   return
 endif

 write(out,'(a,t60,a,1p,e10.4)',iostat=ioerr)
$   'Fraction of Carbon in Poultry Grain',': ',fcg(2)
 if (ioerr .ne. 0) then
   call ioreturn(ioerr)
   return
 endif

 write(out,'(a,t60,a,1p,e10.4)',iostat=ioerr)
$   'Fraction of Carbon in Milk Cow Grain',': ',fcg(3)
 if (ioerr .ne. 0) then
   call ioreturn(ioerr)
```

```
    return
  endif

  write(out,'(a,t60,a,1p,e10.4)',iostat=ioerr)
$   'Fraction of Carbon in Layer Hen Grain',': ',fcg(4)
  if (ioerr .ne. 0) then
    call ioreturn(ioerr)
    return
  endif

  write(out,'(a,t60,a,1p,e10.4)',iostat=ioerr)
$   'Fraction of Carbon in Beef Cattle Hay',': ',fch(1)
  if (ioerr .ne. 0) then
    call ioreturn(ioerr)
    return
  endif

  write(out,'(a,t60,a,1p,e10.4)',iostat=ioerr)
$   'Fraction of Carbon in Poultry Hay',': ',fch(2)
  if (ioerr .ne. 0) then
    call ioreturn(ioerr)
    return
  endif

  write(out,'(a,t60,a,1p,e10.4)',iostat=ioerr)
$   'Fraction of Carbon in Milk Cow Hay',': ',fch(3)
  if (ioerr .ne. 0) then
    call ioreturn(ioerr)
    return
  endif

  write(out,'(a,t60,a,1p,e10.4)',iostat=ioerr)
$   'Fraction of Carbon in Layer Hen Hay',': ',fch(4)
  if (ioerr .ne. 0) then
    call ioreturn(ioerr)
    return
  endif

  write(out,'(a,t60,a,1p,e10.4)',iostat=ioerr)
$   'Fraction of Carbon in Soil',': ',fcd
  if (ioerr .ne. 0) then
    call ioreturn(ioerr)
    return
  endif

  write(out,'(a,t60,a,1p,e10.4)',iostat=ioerr)
$   'Specific Activity Equivalence of Animal',': ',satac
  if (ioerr .ne. 0) then
    call ioreturn(ioerr)
    return
  endif

  write(out,'(a,t60,a,1p,e10.4)',iostat=ioerr)
$   'Fraction of Hydrogen in Leafy Vegetables',': ',fhv(1)
  if (ioerr .ne. 0) then
    call ioreturn(ioerr)
    return
  endif
```

```
 write(out,'(a,t60,a,1p,e10.4)',iostat=ioerr)
$   'Fraction of Hydrogen in Other Vegetables',': ',fhv(2)
 if (ioerr .ne. 0) then
   call ioreturn(ioerr)
   return
 endif

 write(out,'(a,t60,a,1p,e10.4)',iostat=ioerr)
$   'Fraction of Hydrogen in Fruits',': ',fhv(3)
 if (ioerr .ne. 0) then
   call ioreturn(ioerr)
   return
 endif

 write(out,'(a,t60,a,1p,e10.4)',iostat=ioerr)
$   'Fraction of Hydrogen in Grains',': ',fhv(4)
 if (ioerr .ne. 0) then
   call ioreturn(ioerr)
   return
 endif

 write(out,'(a,t60,a,1p,e10.4)',iostat=ioerr)
$   'Fraction of Hydrogen in Beef',': ',fha(1)
 if (ioerr .ne. 0) then
   call ioreturn(ioerr)
   return
 endif

 write(out,'(a,t60,a,1p,e10.4)',iostat=ioerr)
$   'Fraction of Hydrogen in Poultry',': ',fha(2)
 if (ioerr .ne. 0) then
   call ioreturn(ioerr)
   return
 endif

 write(out,'(a,t60,a,1p,e10.4)',iostat=ioerr)
$   'Fraction of Hydrogen in Milk',': ',fha(3)
 if (ioerr .ne. 0) then
   call ioreturn(ioerr)
   return
 endif

 write(out,'(a,t60,a,1p,e10.4)',iostat=ioerr)
$   'Fraction of Hydrogen in Eggs',': ',fha(4)
 if (ioerr .ne. 0) then
   call ioreturn(ioerr)
   return
 endif

 write(out,'(a,t60,a,1p,e10.4)',iostat=ioerr)
$   'Fraction of Hydrogen in Beef Cattle Forage',': ',fhf(1)
 if (ioerr .ne. 0) then
   call ioreturn(ioerr)
   return
 endif

 write(out,'(a,t60,a,1p,e10.4)',iostat=ioerr)
$   'Fraction of Hydrogen in Poultry Forage',': ',fhf(2)
 if (ioerr .ne. 0) then
```

```
    call ioreturn(ioerr)
    return
  endif

  write(out,'(a,t60,a,1p,e10.4)',iostat=ioerr)
$   'Fraction of Hydrogen in Milk Cow Forage',': ',fhf(3)
  if (ioerr .ne. 0) then
    call ioreturn(ioerr)
    return
  endif

  write(out,'(a,t60,a,1p,e10.4)',iostat=ioerr)
$   'Fraction of Hydrogen in Layer Hen Forage',': ',fhf(4)
  if (ioerr .ne. 0) then
    call ioreturn(ioerr)
    return
  endif

  write(out,'(a,t60,a,1p,e10.4)',iostat=ioerr)
$   'Fraction of Hydrogen in Beef Cattle Grain',': ',fhg(1)
  if (ioerr .ne. 0) then
    call ioreturn(ioerr)
    return
  endif

  write(out,'(a,t60,a,1p,e10.4)',iostat=ioerr)
$   'Fraction of Hydrogen in Poultry Grain',': ',fhg(2)
  if (ioerr .ne. 0) then
    call ioreturn(ioerr)
    return
  endif

  write(out,'(a,t60,a,1p,e10.4)',iostat=ioerr)
$   'Fraction of Hydrogen in Milk Cow Grain',': ',fhg(3)
  if (ioerr .ne. 0) then
    call ioreturn(ioerr)
    return
  endif

  write(out,'(a,t60,a,1p,e10.4)',iostat=ioerr)
$   'Fraction of Hydrogen in Layer Hen Grain',': ',fhg(4)
  if (ioerr .ne. 0) then
    call ioreturn(ioerr)
    return
  endif

  write(out,'(a,t60,a,1p,e10.4)',iostat=ioerr)
$   'Fraction of Hydrogen in Beef Cattle Hay',': ',fhh(1)
  if (ioerr .ne. 0) then
    call ioreturn(ioerr)
    return
  endif

  write(out,'(a,t60,a,1p,e10.4)',iostat=ioerr)
$   'Fraction of Hydrogen in Poultry Hay',': ',fhh(2)
  if (ioerr .ne. 0) then
    call ioreturn(ioerr)
    return
  endif
```

```
 write(out,'(a,t60,a,1p,e10.4)',iostat=ioerr)
$   'Fraction of Hydrogen in Milk Cow Hay',': ',fhh(3)
 if (ioerr .ne. 0) then
   call ioreturn(ioerr)
   return
 endif

 write(out,'(a,t60,a,1p,e10.4)',iostat=ioerr)
$   'Fraction of Hydrogen in Layer Hen Hay',': ',fhh(4)
 if (ioerr .ne. 0) then
   call ioreturn(ioerr)
   return
 endif

 write(out,'(a,t60,a,1p,e10.4)',iostat=ioerr)
$   'Fraction of Hydrogen in Soil',': ',fhd
 if (ioerr .ne. 0) then
   call ioreturn(ioerr)
   return
 endif

 write(out,'(a,t60,a,1p,e10.4)',iostat=ioerr)
$   'Tritium Equivalence Between Animal and Food',': ',satah
 if (ioerr .ne. 0) then
   call ioreturn(ioerr)
   return
 endif

 write(out,'(a,t60,a,1p,e10.4)',iostat=ioerr)
$   'Tritium Equivalence Between Plant and Soil',': ',sasvh
 if (ioerr .ne. 0) then
   call ioreturn(ioerr)
   return
 endif

 write(out,'(a,t60,a,1p,e10.4)',iostat=ioerr)
$   'Tritium Equivalence Between Plant and Water',': ',sawvh
 if (ioerr .ne. 0) then
   call ioreturn(ioerr)
   return
 endif

 write(out,'(a,t60,a,1p,e10.4,a)',iostat=ioerr)
$   'Moisture Content of Soil',': ',sh,' L/kg'
 if (ioerr .ne. 0) then
   call ioreturn(ioerr)
   return
 endif

 write(out,'(a)',iostat=ioerr) 'TIME DATA:'
 if (ioerr .ne. 0) then
   call ioreturn(ioerr)
   return
 endif

 write(out,'(a,t60,a,1p,e10.4,a)',iostat=ioerr)
$   'Start Time',': ',tstart,' d'
 if (ioerr .ne. 0) then
   call ioreturn(ioerr)
```

```
    return
  endif

  write(out,'(a,t60,a,1p,e10.4,a)',iostat=ioerr)
$    'End Time',': ',tend,' d'
  if (ioerr .ne. 0) then
    call ioreturn(ioerr)
    return
  endif

  write(out,'(a,t60,a,1p,e10.4,a)',iostat=ioerr)
$    'DT Size',': ',dt,' d'
  if (ioerr .ne. 0) then
    call ioreturn(ioerr)
    return
  endif

  write(out,'(a,t60,a,1p,e10.4,a)',iostat=ioerr)
$    'Time Step Size',': ',ttr,' d'
  if (ioerr .ne. 0) then
    call ioreturn(ioerr)
    return
  endif

  write(out,'(a,t60,a,i5,a)',iostat=ioerr)
$    'Write results every',': ',skip,' calculation times'
  if (ioerr .ne. 0) then
    call ioreturn(ioerr)
    return
  endif

  write(out,'(a)',iostat=ioerr) ' '
  if (ioerr .ne. 0) then
    call ioreturn(ioerr)
    return
  endif

  write(out,'(a)',iostat=ioerr) 'INITIAL ACTIVITIES:'
  if (ioerr .ne. 0) then
    call ioreturn(ioerr)
    return
  endif

  write(out,'(/a,i3)',iostat=ioerr) 'Number of chains: ',nchnp2
  if (ioerr .ne. 0) then
    call ioreturn(ioerr)
    return
  endif

  write(out,'(a)',iostat=ioerr) ' Chain       Chain      Initial'
  if (ioerr .ne. 0) then
    call ioreturn(ioerr)
    return
  endif

  write(out,'(a)',iostat=ioerr) 'Number        Name     Activity'
  if (ioerr .ne. 0) then
    call ioreturn(ioerr)
    return
```

```
      endif

      write(out,'(a)',iostat=ioerr) '                                    (pCi/g)'
      if (ioerr .ne. 0) then
        call ioreturn(ioerr)
        return
      endif

      write(out,'(i6,2x,a,2x,1p,e10.4)',iostat=ioerr)
     $   (ichn,chain(ichn),conc(ichn),ichn=1,nchn)
      if (ioerr .ne. 0) then
        call ioreturn(ioerr)
        return
      endif
      end
```

F.87 ROPER.FOR

```
c***********************************************************************
c
c  subroutine roper; 5/93
c  C. David Updegraff, GRAM, Inc., Albuquerque, NM 87112.
c  Solves the 'r' operator for deposition/accumulation over a time
c  period using equation 2.5, equation B.2 and Table 2.4.
c***********************************************************************
c
      subroutine roper(c,r,lambdr,d,t,nr)

      double precision c
      double precision d,dlmexp
      double precision k
      double precision lambdr
      double precision one
      double precision r
      double precision t
      double precision zero
      integer j
      integer maxr
      integer n,nr
      parameter (maxr=20)
      parameter (one=1.0d+0,zero=0.0d+0)
      dimension c(maxr)
      dimension d(maxr,maxr)
      dimension k(maxr,maxr)
      dimension lambdr(maxr)
      dimension r(maxr)
c
      call kcoef(c,lambdr,d,k,nr,maxr)
c
      do 20, j=1,nr
         r(j)=zero
            do 10, n=1,j
               r(j)=r(j)+k(j,n)*dlmexp(lambdr(n)*t)/lambdr(n)
   10       continue
CCC...added a check for small numbers that turn negative;
CCC...zero those numbers out
CCC            r(j)=lambdr(j)*r(j)
            r(j)=max(zero,lambdr(j)*r(j))
   20 continue
      end
```

F.88 RPTTIM.FOR

```
c
c...this subroutine determines if the current time is a
c   report time
c
      subroutine rpttim(tcalc,tstop,period,skip,histry,report)

      double precision tcalc,tstop
      integer period
      integer skip
      logical histry
      logical report

      report=.false.

      if (skip .eq. 1) report=.true.
      if (tcalc .ge. tstop) report=.true.
      if (skip .ne. 0) then
         if (mod(period,skip) .eq. 1) report=.true.
      else
         report=.false.
      endif
      if (.not. histry) report=.false.
      end
```

F.89 SDOPER.FOR

```
c***********************************************************************
c
c  subroutine sdoper; ver. 1.0; 11/94
c  C. David Updegraff, GRAM, Inc., Albuquerque, NM 87112.
c  Solves the 3 box model integrated drinking water operator
c  for integrated activity over a time period, eq. (B.2-B.5)
c
c***********************************************************************
c

      subroutine sdoper(c,s,lambdr,d,t,nr)

      double precision c
      double precision d,d1mexp
      double precision k
      double precision lambdr
      double precision one
      double precision s
      double precision t
      double precision zero
      integer j
      integer maxr
      integer n,nr
      parameter (maxr=60)
      parameter (one=1.0d+0,zero=0.0d+0)
      dimension c(maxr)
      dimension d(maxr,maxr)
      dimension k(maxr,maxr)
      dimension lambdr(maxr)
      dimension s(maxr)
c
      call kdcoef(c,lambdr,d,k,nr,maxr)
c
      do 20, j=1,nr
         s(j)=zero
            do 10, n=1,j
               s(j)=s(j)+k(j,n)*d1mexp(lambdr(n)*t)/lambdr(n)
   10       continue
            s(j)=max(zero,s(j))
   20 continue
      end
```

F.90 SOIL.FOR

```
c
c...this subroutine calculates ingestion dose to soil according
c   to Eqn. 5.73 as corrected in corrections memo
c
      subroutine soil(nrmax,npmax,dfg,gr,nip,nr,prfrac,sasttr,
     $   tg,ti,ttr,tx,dsr)

      double precision dfg,dsr
      double precision factor
      double precision gr
      double precision prfrac
      double precision tg,ti,ttr,tx
      double precision sasttr
      integer j
      integer k
      integer nip,npmax,nr,nrmax
      dimension dfg(nrmax,0:npmax)
      dimension dsr(nrmax,0:npmax)
      dimension nip(nrmax)
      dimension prfrac(nrmax,0:npmax)
      dimension sasttr(nrmax)
      factor=(ti+tx+tg)/ttr
      do 20, j=1,nr
         do 10, k=0,nip(j)
            dsr(j,k)=gr*factor*sasttr(j)*prfrac(j,k)*dfg(j,k)
   10    continue
   20 continue
      end
```

F.91 SOLVDE.FOR

```
      include "cdecl.inc"

      subroutine solvde(atol,conc,ia,istate,ja,neq,nnz,rtol,t,tout)

      external conclc
      external jacobn
      character*80 status_msg
      double precision atol
      double precision conc
      double precision rtol,rwork
      double precision t,tout
      double precision year
      double precision zero
      integer ia,ieq,inz,iopt,istate,itask,itol,iwork
      integer ja
      integer lenrat,liw,lrw,lwm
      integer maxbox,maxlwm,maxneq,maxnuc,maxnnz,mf
      integer neq,nnz
      integer c_rc
      parameter (lenrat=1)
      parameter (maxbox=12,maxnuc=20)
      parameter (maxneq=maxbox*maxnuc)
      parameter (maxnnz=(maxbox*maxnuc*(maxnuc+3))/2)
      parameter (maxlwm=2*maxnnz+2*maxneq+(maxnnz+9*maxneq)/lenrat)
      parameter (year=365.25d+0)
      parameter (zero=0.0d+0)
      dimension conc(neq)
      dimension ia(neq+1)
      dimension iwork(31+maxneq+maxnnz)
      dimension ja(nnz)
      dimension rwork(20+9*maxneq+maxlwm)
c
c...calculate/define some parameters for lsodes routine
      lwm=2*nnz+2*neq+(nnz+9*neq)/lenrat
      lrw=20+9*neq+lwm
      liw=31+neq+nnz
      itol=1
      itask=4
      iopt=1
      rwork(1)=tout
      rwork(5)=zero
      rwork(6)=year
      rwork(7)=zero
      rwork(8)=zero
      iwork(5)=5
      iwork(6)=10000
      iwork(7)=10
      mf=021
c
c...store sparsity structure into arrays used by lsodes

      do 10, ieq=1,neq+1
         iwork(ieq+30)=ia(ieq)
   10 continue

      do 20, inz=1,nnz
```

```
      iwork(inz+neq+31)=ja(inz)
   20 continue

CCC changed lsodes to return a fatal error flag to stop calculations
ccc istate less than zero means an error occurred so stop

      call lsodes(conclc,neq,conc,t,tout,itol,rtol,atol,itask,
     $   istate,iopt,rwork,lrw,iwork,liw,jacobn,mf)

      if (istate .lt. 0) then
          status_msg(1:80) = "lsodes routine failed..."C
          call CMessage(status_msg,c_rc)
      endif

      end
```

F.92 SOPER.FOR

```
c***********************************************************************
c
c  subroutine soper; 5/93
c  C. David Updegraff, GRAM, Inc., Albuquerque, NM 87112.
c  Solves the 's' operator for integrated activity over a time period
c  using equation 2.5, equation B.2, and Table 2.4.
c***********************************************************************
c
      subroutine soper(c,s,lambdr,d,t,nr)

      double precision c
      double precision d,d1mexp
      double precision k
      double precision lambdr
      double precision one
      double precision s
      double precision t
      double precision zero
      integer j
      integer maxr
      integer n,nr
      parameter (maxr=20)
      parameter (one=1.0d+0,zero=0.0d+0)
      dimension c(maxr)
      dimension d(maxr,maxr)
      dimension k(maxr,maxr)
      dimension lambdr(maxr)
      dimension s(maxr)
c
      call kcoef(c,lambdr,d,k,nr,maxr)
c
      do 20, j=1,nr
         s(j)=zero
            do 10, n=1,j
               s(j)=s(j)+k(j,n)*d1mexp(lambdr(n)*t)/lambdr(n)
   10       continue
            s(j)=max(zero,lambdr(j)*s(j))
   20 continue
      end
```

F.93 SPAH.FOR

```
c
c   this subroutine calculates the 'pptf's for the soil-plant-
c   animal-human pathway. See Section 5.3.2
c
      subroutine spah(maxnr,nr,bf,bg,bh,d,fa,lambda,mlf,mlg,mlh,
     $   qd,qf,qg,qh,tca,tff,tgg,tgh,tha,wf,wg,wh,xf,xg,xh,pptf)

      double precision bf,bg,bh
      double precision d
      double precision cs,csad,csaf,csasg,csash,csfc,csgc,cshc
      double precision fa
      double precision lambda
      double precision mlf,mlg,mlh
      double precision one
      double precision pptf
      double precision qd,qf,qg,qh
      double precision tca,tff,tgg,tgh,tha,thous
      double precision year
      double precision wf,wg,wh
      double precision xf,xg,xh
      double precision zero
      integer a
      integer i
      integer j
      integer maxnr,maxpar
      integer nr
      parameter (maxpar=20)
      parameter (one=1.0d+0)
      parameter (thous=1.0d+3)
      parameter (year=365.25d+0)
      parameter (zero=0.0d+0)
      dimension cs(maxpar),csad(maxpar),csaf(maxpar),csasg(maxpar)
      dimension csash(maxpar),csfc(maxpar),csgc(maxpar),cshc(maxpar)
      dimension bf(4,maxnr),bg(4,maxnr),bh(4,maxnr)
      dimension lambda(maxnr)
      dimension fa(4,maxnr)
      dimension d(maxnr,maxnr)
      dimension mlf(4),mlg(4),mlh(4)
      dimension pptf(maxnr,maxnr,4)
      dimension qd(4),qf(4),qg(4),qh(4)
      dimension tca(4),tha(4),tgg(4),tgh(4),tff(4)
      dimension wf(4),wg(4),wh(4)
      dimension xf(4),xg(4),xh(4)
c
      do 90, a=1,4
         do 90, i=1,nr
            do 10, j=1,nr
c...initalize the pptfs
               pptf(j,i,a)=zero
   10       continue
c...calculate concentration factor
            do 20, j=1,nr
               cs(j)=zero
   20       continue
            cs(i)=one
c......eqn. 5.11
            call aoper(cs,cshc,lambda,d,tgh(a),nr)
```

```
            do 30, j=1,nr
               cshc(j)=thous*(mlh(a)+bh(a,j))*wh(a)*cshc(j)/cs(i)
   30       continue
c......eqn. 5.12
            call aoper(cs,csgc,lambda,d,tgg(a),nr)
            do 40, j=1,nr
               csgc(j)=thous*(mlg(a)+bg(a,j))*wg(a)*csgc(j)/cs(i)
   40       continue
c......eqn. 5.15
            call soper(cs,csfc,lambda,d,tff(a),nr)
            do 50, j=1,nr
               csfc(j)=
     $            thous*(mlf(a)+bf(a,j))*wf(a)*csfc(j)/(cs(i)*tff(a))
   50       continue
            do 60, j=1,nr
c......eqn. 5.16
               csash(j)=fa(a,j)*qh(a)*xh(a)*cshc(j)
c......eqn. 5.17
               csasg(j)=fa(a,j)*qg(a)*xg(a)*csgc(j)
c......eqn. 5.18
               csaf(j)=fa(a,j)*qf(a)*xf(a)*csfc(j)
   60       continue
c......eqn. 5.19
            call soper(cs,csad,lambda,d,tff(a),nr)
            do 70, j=1,nr
               csad(j)=thous*fa(a,j)*qd(a)*wf(a)*qf(a)*xf(a)*csad(j)/
     $            (cs(i)*tff(a))
   70       continue
c......eqn. 5.20
            call aoper(csash,csash,lambda,d,tha(a),nr)
            call soper(csash,csash,lambda,d,tca(a),nr)
            call aoper(csasg,csasg,lambda,d,tha(a),nr)
            call soper(csasg,csasg,lambda,d,tca(a),nr)
            call aoper(csaf,csaf,lambda,d,tha(a),nr)
            call aoper(csad,csad,lambda,d,tha(a),nr)
            do 80, j=1,nr
               pptf(j,i,a)=(csash(j)+csasg(j)+
     $            csaf(j)*tca(a)+csad(j)*tca(a))/year
   80       continue
   90 continue
      end
```

F.94 SPAHC.FOR

```
c
c   this subroutine calculates the 'pptf's for the soil-plant-
c   animal-human pathway for carbon-14 only using equation C.2
c   and modified forms of 5.11, 5.12, and 5.15 (modified for
c   long-lived carbon-14, i.e. decay terms drop out)
c
      subroutine spahc(maxnr,bf,bg,bh,fca,fcd,fcf,fcg,fch,mlf,mlg,mlh,
     $   qd,qf,qg,qh,satac,tca,tff,tfg,tfh,wf,wg,wh,xf,xg,xh,pptf)

      double precision bf,bg,bh
      double precision csfc,csgc,cshc,csoil
      double precision fca,fcd,fcf,fcg,fch
      double precision mlf,mlg,mlh
      double precision one
      double precision pptf
      double precision qd,qf,qg,qh
      double precision satac
      double precision tca,tff,tfg,tfh,thous
      double precision wf,wg,wh
      double precision xf,xg,xh
      double precision year
      double precision zero
      integer a
      integer maxnr
      parameter (one=1.0d+0)
      parameter (thous=1.0d+3)
      parameter (year=365.25d+0)
      parameter (zero=0.0d+0)
c
      dimension bf(4),bg(4),bh(4)
      dimension fca(4),fcf(4),fcg(4),fch(4)
      dimension mlf(4),mlg(4),mlh(4)
      dimension pptf(maxnr,maxnr,4)
      dimension qd(4),qf(4),qg(4),qh(4)
      dimension tca(4),tff(4),tfg(4),tfh(4)
      dimension xf(4),xg(4),xh(4)
      dimension wf(4),wg(4),wh(4)
c
      do 10, a=1,4
c...initalize the pptfs
         pptf(1,1,a)=zero
c
c...calculate concentration factor
c......eqn. 5.11 modified for long-lived carbon
         cshc=thous*(mlh(a)+bh(a))*wh(a)
c......eqn. 5.12 modified for long-lived carbon
         csgc=thous*(mlg(a)+bg(a))*wg(a)
c......eqn. 5.15 modified for long-lived carbon
         csfc=thous*(mlf(a)+bf(a))*wf(a)
c...set soil concentration in g/kg
         csoil=thous
c......eqn. C.2
         pptf(1,1,a)=fca(a)*
     $     ((qf(a)*xf(a)*csfc*tff(a)+
     $       qd(a)*wf(a)*qf(a)*xf(a)*csoil*tff(a)+
     $       qg(a)*xg(a)*csgc*tfg(a)+
     $       qh(a)*xh(a)*cshc*tfh(a))*tca(a)*satac)/
```

```
  $     ((fcf(a)*qf(a)*tff(a)+
  $       fcd*qd(a)*wf(a)*qf(a)*tff(a)+
  $       fcg(a)*qg(a)*tfg(a)+
  $       fch(a)*qh(a)*tfh(a))*year)
10 continue
   end
```

F.95 SPAHH.FOR

```
c
c   this subroutine calculates the 'pptf's for the soil-plant-
c   animal-human pathway for trit  n according to
c   Equations D.4 and D.1
c
      subroutine spahh(maxnr,fha,fhd,fhf,fhg,fhh,qd,qf,qg,qh,qw,
     $   sasvh,satah,sh,tca,tff,tfg,tfh,tfw,wf,xf,xg,xh,pptf)

      double precision cs,csfc,csgc,cshc
      double precision fha,fhd,fhf,fhg,fhh
      double precision nine
      double precision one
      double precision pptf
      double precision qd,qf,qg,qh,qw
      double precision sasvh,satah,sh
      double precision tca,tff,tfg,tfh,tfw,thous
      double precision wf
      double precision xf,xg,xh
      double precision year
      double precision zero
      integer a
      integer maxnr
      parameter (nine=9.0d+0)
      parameter (one=1.0d+0)
      parameter (thous=1.0d+3)
      parameter (year=365.25d+0)
      parameter (zero=0.0d+0)
      dimension fha(4),fhf(4),fhg(4),fhh(4)
      dimension pptf(maxnr,maxnr,4)
      dimension qd(4),qf(4),qg(4),qh(4),qw(4)
      dimension tca(4),tff(4),tfg(4),tfh(4),tfw(4)
      dimension xf(4),xg(4),xh(4)
      dimension wf(4)
c
      do 10, a=1,4
c...initalize the pptfs
         pptf(1,1,a)=zero
c...calculate concentration factor
         cs=one
c......eqn. D.1 for forage
         csfc=nine*fhf(a)*cs*sasvh*thous/(sh*cs)
c......eqn. D.1 for grain
         csgc=nine*fhg(a)*cs*sasvh*thous/(sh*cs)
c......eqn. D.1 for hay
         cshc=nine*fhh(a)*cs*sasvh*thous/(sh*cs)
c......eqn. D.4
         pptf(1,1,a)=
     $      fha(a)*((qf(a)*xf(a)*csfc*tff(a)+
     $               qg(a)*xg(a)*csgc*tfg(a)+
     $               qh(a)*xh(a)*cshc*tfh(a)+
     $               thous*qf(a)*xf(a)*qd(a)*wf(a)*tff(a)*cs/cs)*
     $               tca(a)*satah)/
     $      ((fhf(a)*qf(a)*tff(a)+
     $        fhd*qd(a)*wf(a)*qf(a)*tff(a)+
     $        fhg(a)*qg(a)*tfg(a)+
     $        fhh(a)*qh(a)*tfh(a)+qw(a)*tfw(a)/nine)*year)
   10 continue
      end
```

F.96 SPARSE.FOR

```
c
c...this subroutine determines the sparsity structure (arrays
c   'ia' and 'ja') and non-zero elements (array 'ddd') in the
c   matrix of ordinary differential equations governing radionuclide
c   decay and inter-box transfer.  the results of this subroutine are
c   transferred by common blocks to subroutines 'conclc' and 'jacobn.'
c   these two subroutines are called by 'lsodes', which is used to
c   solve the stiff set of ordinary differential equations.
c
      subroutine sparse(d,f,l,lambda,nbox,nnuc,ddd,ia,ja,neq,nnz)

      double precision d,ddd
      double precision f
      double precision l,lambda
      double precision zero
      integer ia,inuc
      integer ja,jbox,jindex,jnuc
      integer maxbox,maxneq,maxnnz,maxnuc
      integer nbox,neq,nnuc,nnz
      parameter (maxbox=12,maxnuc=20)
      parameter (maxneq=maxbox*maxnuc)
      parameter (maxnnz=(maxbox*maxnuc*(maxnuc+3))/2)
      parameter (zero=0.0d+0)
      dimension d(maxnuc,maxnuc)
      dimension ddd(maxnnz)
      dimension ia(maxneq+1)
      dimension ja(maxnnz)
      dimension l(maxbox,maxnuc)
      dimension lambda(maxnuc)
c
c...initialize number of non-zero elements and number of equations
      nnz=0
      neq=0
c     write(*,*) ' from sparse, dd= '
c     write(*,*) ((d(inuc,jbox),jbox=1,nnuc),inuc=1,nnuc)
c
c...loop over each radionuclide
      do 30, jnuc=1,nnuc
         jindex=(jnuc-1)*nbox
c
c......loop over each box within each radionuclide chain
         do 20, jbox=1,nbox
c........start: non-zero coefficients in diagonal boxes
            nnz=nnz+1
            neq=neq+1
            ia(neq)=nnz
c...........non-zero coefficients for all boxes but last (aquifer) box
            if (jbox .lt. nbox) then
               ja(nnz)=jindex+jbox
               ddd(nnz)=-l(jbox,jnuc)-lambda(jnuc)
               nnz=nnz+1
               ja(nnz)=jindex+jbox+1
               ddd(nnz)=l(jbox,jnuc)
c...........coefficients for last (aquifer) box
            else
c..............special case for residential scenario
               if (f .gt. zero) then
```

```
                  ja(nnz)=jindex+1
                  ddd(nnz)=l(nbox,jnuc)*f
                  nnz=nnz+1
               endif
               ja(nnz)=jindex+jbox
               ddd(nnz)=-l(jbox,jnuc)-lambda(jnuc)
            endif
c.........end: coefficients in diagonal boxes
c.........start: coefficients for subdiagonal boxes
            do 10, inuc=jnuc+1,nnuc
               if (d(jnuc,inuc) .gt. zero) then
                  nnz=nnz+1
                  ja(nnz)=jbox+(inuc-1)*nbox
                  ddd(nnz)=d(jnuc,inuc)*lambda(inuc)
               endif
   10       continue
c.........end: coefficients for subdiagonal boxes
   20    continue
   30 continue
      ia(neq+1)=nnz+1
c     write(*,*) ' new nuclide, neq= ',neq,'   nnz= ',nnz
c     write(*,*) ' ia='
c     write(*,*) (ia(inuc),inuc=1,neq+1)
c     write(*,*) ' ja='
c     write(*,*) (ja(inuc),inuc=1,nnz)
c     write(*,*) ' ddd='
c     write(*,*) (ddd(inuc),inuc=1,nnz)
      end
```

F.97 SPH.FOR

```
c
c   this subroutine calculates the 'pptf's for the soil-plant-human
c   pathway as given in Section 5.3.1.
c
      subroutine sph(maxnr,nr,bv,lambda,d,mlv,tcv,tgv,thv,wv,pptf)

      double precision bv
      double precision d
      double precision cs,csvh,csvp
      double precision lambda
      double precision mlv
      double precision one
      double precision pptf
      double precision tcv,tgv,thv,thous
      double precision year
      double precision wv
      double precision zero
      integer i
      integer j
      integer maxpar
      integer nr,maxnr
      integer v
      parameter (maxpar=20)
      parameter (one=1.0d+0)
      parameter (thous=1.0d+3)
      parameter (year=365.25d+0)
      parameter (zero=0.0d+0)
      dimension bv(4,maxnr)
      dimension cs(maxpar),csvh(maxpar),csvp(maxpar)
      dimension d(maxnr,maxnr)
      dimension lambda(maxnr)
      dimension mlv(4)
      dimension pptf(maxnr,maxnr,4)
      dimension tcv(4),tgv(4),thv(4)
      dimension wv(4)
c
      do 60, v=1,4
         do 50, i=1,nr
c...initalize the pptfs
            do 10, j=1,nr
               pptf(j,i,v)=zero
   10       continue
c
c...calculate concentration factor
            do 20, j=1,nr
               cs(j)=zero
   20       continue
            cs(i)=one
c......eqn. 5.4
            call aoper(cs,csvh,lambda,d,tgv(v),nr)
c......eqn. 5.5
            do 30, j=1,nr
               csvh(j)=thous*(mlv(v)+bv(v,j))*wv(v)*csvh(j)/cs(i)
   30       continue
c......eqn. 5.6
            call aoper(csvh,csvp,lambda,d,thv(v),nr)
c......eqn. 5.8
```

```
         call soper(csvp,pptf(1,i,v),lambda,d,tcv(v),nr)
         do 40, j=1,nr
            pptf(j,i,v)=pptf(j,i,v)/year
40       continue
50    continue
60 continue
   end
```

F.98 SPHH.FOR

```
c
c   this subroutine calculates the pptf for the soil-plant-human
c   pathway for tritium only as given in Equations D.1 and D.2
c
      subroutine sphh(maxnr,sasvh,sh,fhv,tcv,pptf)

      double precision cs,csvh
      double precision fhv
      double precision nine
      double precision one
      double precision pptf
      double precision sasvh,sh
      double precision tcv,thous
      double precision year
      double precision zero
      integer v
      integer maxnr
      parameter (nine=9.0d+0,one=1.0d+0,thous=1.0d+3)
      parameter (year=365.25d+0,zero=0.0d+0)
      dimension fhv(4)
      dimension pptf(maxnr,maxnr,4)
      dimension tcv(4)
c
      do 10, v=1,4
c...initalize the pptfs
         pptf(1,1,v)=zero
c...calculate concentration factor
         cs=one
c......eqn. D.1
         csvh=nine*fhv(v)*cs*sasvh*thous/(sh*cs)
c......eqn. D.2
         pptf(1,1,v)=csvh*tcv(v)/year
   10 continue
      end
```

F.99 SROPER.FOR

```
c
c......calculate the ground-water integrated activity over
c       the period of interest. See Section B.7.
c

      subroutine sroper(maxpar,maxprg,nprog,k,root,t,c)

      double complex c
      double complex k
      double complex root
      double complex onec
      integer ibox,ipar,iprog,iroot
      integer maxpar,maxprg
      integer nbox,nprog,nroot
      double precision t
      double precision zero
      dimension c(3,maxprg)
      dimension k(3,3,maxpar,maxprg)
      dimension root(3,maxpar)
      parameter (nbox=3,nroot=3)
      parameter (onec=(1.0d+0,0.0d+0))
      parameter (zero=0.0d+0)

      do 40, iprog=1,nprog
         do 30, ibox=1,nbox
            c(ibox,iprog)=zero
            do 20, ipar=1,iprog
               do 10, iroot=1,nroot
                  c(ibox,iprog)=c(ibox,iprog)+k(ibox,iroot,ipar,iprog)*
c    $               c1mexp(root(iroot,ipar)*t)/root(iroot,ipar)
     $               (cdexp(root(iroot,ipar)*t)-onec)/root(iroot,ipar)
   10          continue
   20       continue

CCC...added a check for small numbers that turn negative;
CCC...zero those numbers out
            if (dble(c(ibox,iprog)) .lt. zero) c(ibox,iprog)=zero

   30    continue
   40 continue
      end
```

F.100 TEDMAX.FOR

```
c
c......this subroutine checks if total tede is maximum so far;
c       if so it is saved, along with its components, and its time
c       of occurrence
c

      subroutine tedmax(mxlist,mxpath,nlist,npath,tcalc,period,persv,
     $   ttede,tedeup,tedemx,pathsv,pathup,nucsv,nucup,dossv,dosup,
     $   sbarsv,sbarup)

      double precision dossv,dosup
      double precision nucsv,nucup
      double precision pathsv,pathup
      double precision tcalc,tedemx,tedeup,ttede
      double precision sbarsv,sbarup
      double precision zero
      parameter (zero=0.0d0)
      integer ilist,ipath
      integer mxlist,mxpath
      integer nlist,npath
      integer period,persv
      dimension dossv(mxlist,mxpath),dosup(mxlist,mxpath)
      dimension nucsv(mxlist),nucup(mxlist)
      dimension pathsv(mxpath),pathup(mxpath)
      dimension sbarsv(mxlist),sbarup(mxlist)
      if (tedeup .gt. tedemx .or.
     &    (tedeup .eq. zero .and. tedemx .eq. zero .and.
     &    period .eq. 1)) then
         persv=period
         ttede=tcalc
         tedemx=tedeup
         do 10, ipath=1,npath
            pathsv(ipath)=pathup(ipath)
10       continue
         do 20, ilist=1,nlist
            nucsv(ilist)=nucup(ilist)
            sbarsv(ilist)=sbarup(ilist)
20       continue
         do 40, ipath=1,npath
            do 30, ilist=1,nlist
               dossv(ilist,ipath)=dosup(ilist,ipath)
30          continue
40       continue
      endif
      end
```

F.101 TEDMXD.FOR

```
c
c......this subroutine checks if total tede is maximum so far;
c       if so it is saved, along with its components, and its time
c       of occurrence
c
      subroutine tedmxd(mxlist,mxorg,mxpath,nlist,norg,npath,tcalc,
     $   period,persv,
     $   ttede,tedeup,tedemx,pathsv,pathup,nucsv,nucup,dossv,dosup,
     $   cwtsv,cwtup,cswtsv,cswtup,hocdsv,hocdup,orgnsv,orgnup)

      double precision cswtsv,cswtup,cwtsv,cwtup
      double precision dossv,dosup
      double precision hocdsv,hocdup
      double precision nucsv,nucup
      double precision pathsv,pathup
      double precision tcalc,tedemx,tedeup,ttede
      double precision orgnsv,orgnup
      integer ilist,iorg,ipath
      integer mxlist,mxorg,mxpath
      integer nlist,norg,npath
      integer period,persv
      dimension cswtsv(mxlist),cswtup(mxlist)
      dimension cwtsv(mxlist),cwtup(mxlist)
      dimension dossv(mxlist,mxpath),dosup(mxlist,mxpath)
      dimension hocdsv(mxlist,mxorg),hocdup(mxlist,mxorg)
      dimension nucsv(mxlist),nucup(mxlist)
      dimension pathsv(mxpath),pathup(mxpath)
      dimension orgnsv(mxorg),orgnup(mxorg)
      if (tedeup .gt. tedemx .or. period .eq. 1) then
         persv=period
         ttede=tcalc
         tedemx=tedeup
      do 10, ilist=1,nlist
         nucsv(ilist)=nucup(ilist)
         cswtsv(ilist)=cswtup(ilist)
         cwtsv(ilist)=cwtup(ilist)
   10 continue
      do 30, ipath=1,npath
         pathsv(ipath)=pathup(ipath)
         do 20, ilist=1,ilist
            dossv(ilist,ipath)=dosup(ilist,ipath)
   20    continue
   30 continue
      do 60, iorg=1,norg
         orgnsv(iorg)=orgnup(iorg)
         do 50, ilist=1,nlist
            hocdsv(ilist,iorg)=hocdup(ilist,iorg)
   50    continue
   60 continue
      endif
      end
```

F.102 TEDMXR.FOR

```
c
c......this subroutine checks if total tede is maximum so far;
c      if so it is saved, along with its components, and its time
c      of occurrence
c
      subroutine tedmxr(mxlist,mxpath,nlist,npath,tcalc,period,persv,
     $   ttede,tedeup,tedemx,pathsv,pathup,nucsv,nucup,dossv,dosup,
     $   astsv,astup,awtsv,awtup)

      double precision astsv,astup,awtsv,awtup
      double precision dossv,dosup
      double precision nucsv,nucup
      double precision pathsv,pathup
      double precision tcalc,tedemx,tedeup,ttede
      integer ilist,ipath
      integer mxlist,mxpath
      integer nlist,npath
      integer period,persv
      dimension astsv(mxlist),astup(mxlist)
      dimension awtsv(mxlist),awtup(mxlist)
      dimension dossv(mxlist,mxpath),dosup(mxlist,mxpath)
      dimension nucsv(mxlist),nucup(mxlist)
      dimension pathsv(mxpath),pathup(mxpath)
      if (tedeup .gt. tedemx .or. period .eq. 1) then
         persv=period
         ttede=tcalc
         tedemx=tedeup
         do 10, ipath=1,npath
            pathsv(ipath)=pathup(ipath)
   10    continue
         do 20, ilist=1,nlist
            nucsv(ilist)=nucup(ilist)
            awtsv(ilist)=awtup(ilist)
            astsv(ilist)=astup(ilist)
   20    continue
         do 40, ipath=1,npath
            do 30, ilist=1,nlist
               dossv(ilist,ipath)=dosup(ilist,ipath)
   30       continue
   40    continue
      endif
      end
```

F.103 TEDSUM.FOR

```
c
c...this subroutine calculates the tede for a chain
c
      subroutine tedsum(mxnr,mxnip,nr,nip,nuc,tede)

      integer ir,iip
      integer mxnr,mxnip
      integer nr,nip
      double precision nuc
      double precision tede
      double precision zero
      dimension nip(mxnr)
      dimension nuc(mxnr,0:mxnip)
      parameter (zero=0.0d+0)
c
c...initialize tede
      tede=zero
c
c...calculate tede by summing over all radionuclides in the chain
      do 20, ir=1,nr
         do 10, iip=0,nip(ir)
            tede=tede+nuc(ir,iip)
   10    continue
   20 continue
      end
```

F.104 TOLER.FOR

```
c
c...this subroutine sets up tolerance factors for lsodes
c
      subroutine toler(conr,neq,atol,rtol)

      double precision atol
      double precision conr
      double precision factor
      double precision rtol
      double precision zero
      parameter (factor=1.0d-12)
      parameter (zero=0.0d+0)
      integer ieq
      integer neq
      dimension conr(neq)
c
      rtol=factor
      atol=zero
      do 10, ieq=1,neq
         atol=max(atol,factor*conr(ieq))
   10 continue
      end
```

F.105 TRNSFR.FOR

```
c
c     this subroutine calculates transfer coefficients between boxes,
c     see eq. (4.7-4.12)
c
      subroutine trnsfr(maxbox,maxnr,nbox,nlay,nr,h1,h2,n1,n2,
     $   f1,f2,rho1,rho2,irr,i,kd,ar,vdr,vtr,fr,wr,l)

      double precision ar
      double precision f1,f2,fr
      double precision h1,h2,hu
      double precision i,irr
      double precision kd
      double precision l
      double precision n1,n2
      double precision one
      double precision rho1,rho2,rt
      double precision theta,thous
      double precision vdr,vir,virr,vtr
      double precision wr
      double precision year
      double precision zero
      integer ibox,ir
      integer maxbox,maxnr
      integer nbox,nlay,nr
      parameter (one=1.0d+0)
      parameter (thous=1.0d+3)
      parameter (year=365.25d+0)
      parameter (zero=0.0d+0)
      dimension kd(maxnr)
      dimension l(maxbox,maxnr)
c
c...calculate aquifer volume and fraction of aquifer
c   returned for surface irrigation
      if (irr .gt. zero) then
         virr=irr*ar*year
      else
         virr=zero
      endif
      vir=i*ar*thous
      vtr=max(vir,virr+vdr)
      wr=one/year
      if (irr .gt. zero) then
         fr=virr/vtr
      else
         fr=zero
      endif
c
c...calculate thickness of layer within unsaturated zone
      hu=h2/dble(nlay)
c
c...calculate block to block transfer coefficients
      do 20, ir=1,nr
c......from soil layer
         theta=n1*f1
         rt=one+kd(ir)*rho1/n1
         l(1,ir)=i/(h1*theta*rt*year)
c......from unsaturated zone layers
```

```
         do 10, ibox=2,nbox-1
            theta=n2*f2
            rt=one+kd(ir)*rho2/n2
            l(ibox,ir)=i/(hu*theta*rt*year)
   10    continue
c......from aquifer
         l(nbox,ir)=wr
   20 continue
      end
```

F.106 UPDOSE.FOR

```
c
c...this subroutine takes the dose calculations for a chain and puts
c   them into the dose array for the list of radionuclides
c
      subroutine updose(mxnr,mxnip,mxpath,mxlist,
     $      nr,nip,npath,
     $      indx,dosnuc,dose)

      double precision dose,dosnuc
      integer i,iip,indx,ipath,ir
      integer nip,npath,nr
      integer mxlist,mxnip,mxnr,mxpath
      dimension dose(mxlist,mxpath)
      dimension dosnuc(mxnr,0:mxnip,mxpath)
      dimension indx(mxnr,0:mxnip)
      dimension nip(mxnr)
      do 30, ipath=1,npath
         do 20, ir=1,nr
            do 10, iip=0,nip(ir)
               i=indx(ir,iip)
               dose(i,ipath)=dose(i,ipath)+dosnuc(ir,iip,ipath)
   10       continue
   20    continue
   30 continue
      end
```

F.107 UPNUC.FOR

```
c
c     This subroutine updates the listed radionuclide dose for all chains
c
      subroutine upnuc(mxnr,mxnip,mxlist,nr,nip,indx,nuc,nucup)

      double precision nuc,nucup
      integer i,iip,indx,ir
      integer mxlist,mxnip,mxnr
      integer nip,nr
      dimension indx(mxnr,0:mxnip)
      dimension nip(mxnr)
      dimension nuc(mxnr,0:mxnip),nucup(mxlist)
      do 20, ir=1,nr
         do 10, iip=0,nip(ir)
            i=indx(ir,iip)
            nucup(i)=nucup(i)+nuc(ir,iip)
   10   continue
   20 continue
      end
```

F.108 UPPATH.FOR

```
c
c     this subroutine updates the pathway dose for all chains
c
      subroutine uppath(mxpath,npath,path,pathup)

      double precision path,pathup
      integer ipath
      integer mxpath
      integer npath
      dimension path(mxpath),pathup(mxpath)
      do 10, ipath=1,npath
         pathup(ipath)=pathup(ipath)+path(ipath)
   10 continue
      end
```


```
c
c     this subroutine updates the pathway dose for all chains
c
      subroutine uppath(mxpath,npath,path,pathup)

      double precision path,pathup
      integer ipath
      integer mxpath
      integer npath
      dimension path(mxpath),pathup(mxpath)
      do 10, ipath=1,npath
         pathup(ipath)=pathup(ipath)+path(ipath)
   10 continue
      end
```

F.109 UPSBAR.FOR

```
c
c     This subroutine updates the listed radionuclide soil activity
c
      subroutine upsbar(mxnr,mxnip,mxlist,nr,nip,
     $   indx,sbar,sbarup)

      double precision sbar,sbarup
      integer i,iip,indx,ir
      integer mxlist,mxnip,mxnr
      integer nip,nr
      dimension indx(mxnr,0:mxnip)
      dimension nip(mxnr)
      dimension sbar(mxnr,0:mxnip),sbarup(mxlist)
      do 20, ir=1,nr
         do 10, iip=0,nip(ir)
            i=indx(ir,iip)
            sbarup(i)=sbarup(i)+sbar(ir,iip)
   10   continue
   20 continue
      end
```

F.110 UPTEDE.FOR

```
c
c     This subroutine updates the total dose for all chains
c
      subroutine uptede(tede,tedeup)
      double precision tede,tedeup
      tedeup=tedeup+tede
      end
```

F.111 UPTIME.FOR

```
c
c     This subroutine updates the time step information
c

      subroutine uptime(con,concen,done,period,tcalc,dt,tstop,endfil)

      double precision dt
      double precision tcalc,tstop
      integer con
      integer endfil
      integer period
      logical concen
      logical done
      done=.false.
      if (concen) then
         read(con,*,iostat=endfil) tcalc
         if (endfil .lt. 0) then
             done=.true.
         elseif (endfil .eq. 0) then
             period=period+1
         else
             call ioreturn(endfil)
             return
         endif
      else
         tcalc=min(tcalc+dt,tstop)
         period=period+1
      endif
      end
```

F.112 VALTST.FOR

```
c
c...this subroutine determines if the new values of an array are
c   decreasing when compared to old values. if they are decreasing
c   the subroutine sets a test value to true to indicate that a maximum
c   value has been reached. the current values of the are
c   saved for the next comparison
c
      subroutine valtst(ndim,valnew,valold,dcreas,zero)

      double precision valnew,valold
      double precision minimum
      integer idim
      integer ndim
      logical dcreas,zero
      dimension valnew(ndim),valold(ndim)
      parameter (minimum=1.0d-03)

c
c...compare new values with integrated with old values. if any of the
c   values are increasing, set flag to false
      do 10, idim=1,ndim
         if (valnew(idim) .gt. valold(idim)) dcreas=.false.
         if (valnew(idim) .gt. minimum) zero = .false.
         valold(idim)=valnew(idim)
   10 continue
c
c...save the new values from this time step as old values
c      do 20, idim=1,ndim
c         valold(idim)=valnew(idim)
c   20 continue
      end
```

F.113 WPAH.FOR

```
c
c   this subroutine calculates the 'pptf's for the
c   irrigation water-plant-animal-human pathway
c
      subroutine wpah(maxnr,nr,bf,bg,bh,d,fa,ir,lambda,lambdw,
     $    mlf,mlg,mlh,ps,qd,qf,qg,qh,qw,rf,rg,rh,tbigf,tbigg,tbigh,
     $   tca,tff,tgg,tgh,tha,wf,wg,wh,yf,yg,yh,xf,xg,xh,xw,pptf)

      double precision bf,bg,bh
      double precision d
      double precision crfd,crfc,crgc,crhc,cw,cwacs,cwahs
      double precision cwahf,cwapf,cwaps,cwaw,cwfc,cwfcs,cwgc,cwgcs
      double precision cwhc,cwhcs
      double precision fa
      double precision ir,irrate
      double precision lambda,lambdw
      double precision mlf,mlg,mlh
      double precision one
      double precision pptf,ps
      double precision qd,qf,qg,qh,qw
      double precision rf,rg,rh,rwff,rwgg,rwhg,rwsf,rwsg
      double precision tbigf,tbigg,tbigh,tca,tff
      double precision tgg,tgh,tha,thous
      double precision wf,wg,wh
      double precision year
      double precision yf,yg,yh
      double precision xf,xg,xh,xw
      double precision zero
      integer a
      integer i
      integer j
      integer maxnr,maxpar
      integer nr
      parameter (maxpar=20)
      parameter (one=1.0d+0)
      parameter (thous=1.0d+3)
      parameter (year=365.25d+0)
      parameter (zero=0.0d+0)
      dimension bf(4,maxnr),bg(4,maxnr),bh(4,maxnr)
      dimension crfc(maxpar),crfd(maxpar),crgc(maxpar),crhc(maxpar)
      dimension cw(maxpar),cwacs(maxpar)
      dimension cwahf(maxpar),cwahs(maxpar),cwapf(maxpar),cwaps(maxpar)
      dimension cwaw(maxpar),cwfc(maxpar),cwfcs(maxpar),cwgc(maxpar)
      dimension cwgcs(maxpar),cwhc(maxpar),cwhcs(maxpar)
      dimension d(maxnr,maxnr)
      dimension fa(4,maxnr)
      dimension lambda(maxnr)
      dimension mlf(4),mlg(4),mlh(4)
      dimension pptf(maxnr,maxnr,4)
      dimension qd(4),qf(4),qg(4),qh(4),qw(4)
      dimension rf(4),rg(4),rh(4)
      dimension rwff(maxpar),rwgg(maxpar),rwhg(maxpar)
      dimension rwsf(maxpar),rwsg(maxpar)
      dimension tbigf(4),tbigg(4),tbigh(4)
      dimension tca(4),tff(4),tgg(4),tgh(4),tha(4)
      dimension wf(4),wg(4),wh(4)
      dimension xf(4),xg(4),xh(4),xw(4)
```

```
      dimension yf(4),yg(4),yh(4)
c
c...initalize the pptfs
      do 200, a=1,4
         do 200, i=1,nr
            do 10, j=1,nr
               pptf(j,i,a)=zero
   10       continue
c
c...calculate concentration factor
            do 20, j=1,nr
               cw(j)=zero
   20       continue
            cw(i)=one
c......forage concentration factors
            irrate=ir*year/tff(a)
c......eqn. 5.37
            do 30, j=1,nr
               rwff(j)=(irrate*rf(a)*tbigf(a)/yf(a))*(cw(j)/cw(i))
   30       continue
c......eqn. 5.38
            call geoper(rwff,cwfc,lambda,lambdw,d,tff(a),nr)
            do 40, j=1,nr
               cwfc(j)=cwfc(j)/tff(a)
   40       continue
c......eqn. 5.43
            do 50, j=1,nr
               rwsf(j)=(irrate/ps)*(cw(j)/cw(i))
   50       continue
c......eqn. 5.44
            call goper(rwsf,cwfcs,lambda,d,tff(a),nr)
            do 60, j=1,nr
               cwfcs(j)=cwfcs(j)/tff(a)
   60       continue
c......eqn. 5.47
            do 70, j=1,nr
               crfc(j)=(mlf(a)+bf(a,j))*wf(a)*cwfcs(j)
   70       continue
c......hay concentration factors
            irrate=ir*year/tgh(a)
c......eqn. 5.48
            do 80, j=1,nr
               rwhg(j)=(irrate*rh(a)*tbigh(a)/yh(a))*(cw(j)/cw(i))
   80       continue
c......eqn. 5.49
            call reoper(rwhg,cwhc,lambda,lambdw,d,tgh(a),nr)
c......eqn. 5.50
            do 90, j=1,nr
               rwsg(j)=(irrate/ps)*(cw(j)/cw(i))
   90       continue
c......eqn. 5.51
            call roper(rwsg,cwhcs,lambda,d,tgh(a),nr)
c......eqn. 5.52
            do 100, j=1,nr
               crhc(j)=(mlh(a)+bh(a,j))*wh(a)*cwhcs(j)
  100       continue
c......grain concentration factors
            irrate=ir*year/tgg(a)
c......eqn. 5.53
```

```
            do 110, j=1,nr
               rwgg(j)=(irrate*rg(a)*tbigg(a)/yg(a))*(cw(j)/cw(i))
  110       continue
c......eqn. 5.54
            call reoper(rwgg,cwgc,lambda,lambdw,d,tgg(a),nr)
c......eqn. 5.55
            do 120, j=1,nr
               rwsg(j)=(irrate/ps)*(cw(j)/cw(i))
  120       continue
c......eqn. 5.56
            call roper(rwsg,cwgcs,lambda,d,tgg(a),nr)
c......eqn. 5.57
            do 130, j=1,nr
               crgc(j)=(mlg(a)+bg(a,j))*wg(a)*cwgcs(j)
  130       continue
c......soil concentration factor while feeding on forage
            irrate=ir*year/tff(a)
c......eqn. 5.58
            do 140, j=1,nr
               rwsf(j)=(irrate/ps)*(cw(j)/cw(i))
  140       continue
c......eqn. 5.59
            call goper(rwsf,crfd,lambda,d,tff(a),nr)
            do 150, j=1,nr
               crfd(j)=crfd(j)/tff(a)
  150       continue
c......eqn. 5.60 and eqn. 5.61
            do 160, j=1,nr
               cwahf(j)=fa(a,j)*qf(a)*xf(a)*
     $            (cwfc(j)+crfc(j)+qd(a)*wf(a)*crfd(j))
               cwacs(j)=fa(a,j)*(qh(a)*xh(a)*(cwhc(j)+crhc(j))+
     $            qg(a)*xg(a)*(cwgc(j)+crgc(j)))
  160       continue
c......eqn. 5.62
            do 170, j=1,nr
               cwahs(j)=cwacs(j)
  170       continue
c......eqn. 5.63
            call aoper(cwahf,cwapf,lambda,d,tha(a),nr)
c......eqn. 5.64
            call aoper(cwahs,cwaps,lambda,d,tha(a),nr)
c......eqn. 5.65
            do 180, j=1,nr
               cwaw(j)=fa(a,j)*qw(a)*xw(a)*(cw(j)/cw(i))
  180       continue
c......eqn. 5.66
            call soper(cwaps,cwaps,lambda,d,tca(a),nr)
            call aoper(cwaw,cwaw,lambda,d,tha(a),nr)
            do 190, j=1,nr
               pptf(j,i,a)=
     $            (tca(a)*cwapf(j)+cwaps(j)+tca(a)*cwaw(j))/year
  190       continue
  200 continue
      end
```

F.114 WPAHC.FOR

```
c
c   this subroutine calculates the 'pptf's for the irrigation
c   water-plant-animal-human pathway for c-14
c
      subroutine wpahc(maxnr,bf,bg,bh,d,fca,fcd,fcf,fcg,fch,
     $   ir,lambda,lambdw,mlf,mlg,mlh,ps,qd,qf,qg,qh,qw,rf,rg,rh,
     $   satac,tbigf,tbigg,tbigh,tca,tff,tfg,tfh,tfw,tgg,tgh,
     $   wf,wg,wh,xf,xg,xh,xw,yf,yg,yh,pptf)

      double precision bf,bg,bh
      double precision d
      double precision crfc,crfd,crgc,crhc
      double precision cwfc,cwfcs,cwgc,cwgcs
      double precision cwhc,cwhcs,cwwc
      double precision fca,fcd,fcf,fcg,fch
      double precision ir,irrate
      double precision lambda,lambdw
      double precision mlf,mlg,mlh
      double precision one
      double precision pptf,ps
      double precision qd,qf,qg,qh,qw
      double precision rf,rg,rh,rwff,rwgg,rwhg,rwsf,rwsg
      double precision satac
      double precision tbigf,tbigg,tbigh,tca,tff,tfg,tfh,tfw
      double precision tgg,tgh,thous
      double precision wf,wg,wh
      double precision xf,xg,xh,xw
      double precision year
      double precision yf,yg,yh
      double precision zero
      integer a
      integer maxnr
      parameter (one=1.0d+0)
      parameter (thous=1.0d+3)
      parameter (year=365.25d+0)
      parameter (zero=0.0d+0)
      dimension bf(4),bg(4),bh(4)
      dimension fca(4),fcf(4),fcg(4),fch(4)
      dimension mlf(4),mlg(4),mlh(4)
      dimension pptf(maxnr,maxnr,4)
      dimension qd(4),qf(4),qg(4),qh(4),qw(4)
      dimension rf(4),rg(4),rh(4)
      dimension tbigf(4),tbigg(4),tbigh(4),tca(4),tff(4),tfg(4),tfh(4)
      dimension tfw(4),tgg(4),tgh(4)
      dimension wf(4),wg(4),wh(4)
      dimension xf(4),xg(4),xh(4),xw(4)
      dimension yf(4),yg(4),yh(4)
c
c...initalize the pptfs
      do 10, a=1,4
         pptf(1,1,a)=zero
c...calculate concentration factor
c...calculations for forage
c......eqn. 5.37
         irrate=ir*year/tff(a)
         rwff=irrate*rf(a)*tbigf(a)/yf(a)
c......eqn. 5.38
```

```
         call geoper(rwff,cwfc,lambda,lambdw,d,tff(a),1)
         cwfc=cwfc/tff(a)
c......eqn. 5.43
         rwsf=irrate/ps
c......eqn. 5.44
         call goper(rwsf,cwfcs,lambda,d,tff(a),1)
         cwfcs=cwfcs/tff(a)
c......eqn. 5.47
         crfc=(mlf(a)+bf(a))*wf(a)*cwfcs
c...calculations for hay
         irrate=ir*year/tgh(a)
c......eqn. 5.48
         rwhg=irrate*rh(a)*tbigh(a)/yh(a)
c......eqn. 5.49
         call reoper(rwhg,cwhc,lambda,lambdw,d,tgh(a),1)
c......eqn. 5.50
         rwsg=irrate/ps
c......eqn. 5.51
         call roper(rwsg,cwhcs,lambda,d,tgh(a),1)
c......eqn. 5.52
         crhc=(mlh(a)+bh(a))*wh(a)*cwhcs
c...calculations for grain
c......eqn. 5.53
         irrate=ir*year/tgg(a)
         rwgg=irrate*rg(a)*tbigg(a)/yg(a)
c......eqn. 5.54
         call reoper(rwgg,cwgc,lambda,lambdw,d,tgg(a),1)
c......eqn. 5.55
         rwsg=irrate/ps
c......eqn. 5.56
         call roper(rwsg,cwgcs,lambda,d,tgg(a),1)
c......eqn. 5.57
         crgc=(mlg(a)+bg(a))*wg(a)*cwgcs
c......eqn. 5.58
c...calculations for soil
         irrate=ir*year/tff(a)
         rwsf=irrate/ps
c......eqn. 5.59
         call goper(rwsf,crfd,lambda,d,tff(a),1)
         crfd=crfd/tff(a)
c...calculations for water
         cwwc=one
c...calculations for pptf (eqn. C.3)
         pptf(1,1,a)=(fca(a)*
     $       (qf(a)*xf(a)*(cwfc+crfc)*tff(a)+
     $        qd(a)*wf(a)*qf(a)*xf(a)*crfd*tff(a)+
     $        qg(a)*xg(a)*(cwgc+crgc)*tfg(a)+
     $        qh(a)*xh(a)*(cwhc+crhc)*tfh(a)+
     $        qw(a)*xw(a)*cwwc*tfw(a))*tca(a)*satac)/
     $      ((fcf(a)*qf(a)*tff(a)+
     $        fcd*qd(a)*wf(a)*qf(a)*tff(a)+
     $        fcg(a)*qg(a)*tfg(a)+
     $        fch(a)*qh(a)*tfh(a))*year)
   10 continue
      end
```

F.115 WPAHH.FOR

```
c
c   this subroutine calculates the pptf for the
c   irrigation water-plant-animal-human pathway for tritium only
c   using Equations D.5, D.6, D.7 and D.8
c
      subroutine wpahh(maxnr,fha,fhd,fhf,fhg,fhh,ir,ps,qd,qf,qg,qh,qw,
     $   satah,sawvh,tca,tff,tfg,tfh,tfw,wf,xf,xg,xh,xw,pptf)

      double precision cw,cwfd,cwfc,cwgc,cwhc,cwwc
      double precision fha,fhd,fhf,fhg,fhh
      double precision ir,irrate
      double precision nine
      double precision one
      double precision pptf,ps
      double precision qd,qf,qg,qh,qw
      double precision rwsf
      double precision satah,sawvh
      double precision tca,tff,tfg,tfh,tfw,thous,two
      double precision wf
      double precision xf,xg,xh,xw
      double precision year
      double precision zero
      integer a
      integer maxnr
      parameter (nine=9.0d+0)
      parameter (one=1.0d+0)
      parameter (thous=1.0d+3)
      parameter (two=2.0d+0)
      parameter (year=365.25d+0)
      parameter (zero=0.0d+0)
      dimension fha(4),fhf(4),fhg(4),fhh(4)
      dimension pptf(maxnr,maxnr,4)
      dimension qd(4),qf(4),qg(4),qh(4),qw(4)
      dimension tca(4),tff(4),tfg(4),tfh(4),tfw(4)
      dimension wf(4)
      dimension xf(4),xg(4),xh(4),xw(4)
c
c...initalize the pptfs
      do 10, a=1,4
         pptf(1,1,a)=zero
c...calculate concentration factor
         cw=one
c......eqn. D.5 for forage
         cwfc=nine*fhf(a)*cw*sawvh/cw
c......eqn. D.5 for grain
         cwgc=nine*fhg(a)*cw*sawvh/cw
c......eqn. D.5 for hay
         cwhc=nine*fhh(a)*cw*sawvh/cw
c......eqn. D.8 and D.9 for soil consumption
         irrate=ir*year/tff(a)
         rwsf=irrate/ps*(cw/cw)
         cwfd=rwsf*tff(a)/two
c......set water concentration
         cwwc=one
c......calculate pptf
         pptf(1,1,a)=
     $      (fha(a)*(qf(a)*xf(a)*cwfc*tff(a)+
```

```
     $                qg(a)*xg(a)*cwgc*tfg(a)+
     $                qh(a)*xh(a)*cwhc*tfh(a)+
     $                qd(a)*wf(a)*qf(a)*xf(a)*cwfd*tff(a)+
     $                qw(a)*xw(a)*cwwc*tfw(a))*tca(a)*satah)/
     $        ((fhf(a)*qf(a)*tff(a)+
     $          fhg(a)*qg(a)*tfg(a)+
     $          fhf(a)*qh(a)*tfh(a)+
     $          fhd*qd(a)*wf(a)*qf(a)*tff(a)+
     $          qw(a)*tfw(a)/nine)*year)
10 continue
   end
```

F.116 WPH.FOR

```
c
c   this subroutine calculates the 'pptf's for the
c   irrigation water-plant-human pathway
c
      subroutine wph(maxnr,nr,bv,d,ir,lambda,lambdw,mlv,ps,rv,
     $   tbigv,tcv,tgv,thv,wv,yv,pptf)

      double precision bv
      double precision d
      double precision crvh,cw,cwvh,cwvhs
      double precision ir,irrate
      double precision lambda,lambdw
      double precision mlv
      double precision one
      double precision pptf,ps
      double precision rv,rwsg,rwvg
      double precision tbigv,tcv,tgv,thv,thous
      double precision wv
      double precision year
      double precision yv
      double precision zero
      integer i
      integer j
      integer maxnr,maxpar
      integer nr
      integer v
      parameter (maxpar=20)
      parameter (one=1.0d+0)
      parameter (thous=1.0d+3)
      parameter (year=365.25d+0)
      parameter (zero=0.0d+0)
      dimension bv(4,maxnr)
      dimension crvh(maxpar)
      dimension cw(maxpar)
      dimension cwvh(maxpar)
      dimension cwvhs(maxpar)
      dimension d(maxnr,maxnr)
      dimension lambda(maxnr)
      dimension mlv(4)
      dimension pptf(maxnr,maxnr,4)
      dimension rv(4)
      dimension rwsg(maxpar),rwvg(maxpar)
      dimension tbigv(4),tcv(4),tgv(4),thv(4)
      dimension wv(4)
      dimension yv(4)
c
c...initalize the pptfs
      do 70, v=1,4
         do 70, i=1,nr
            do 10, j=1,nr
               pptf(j,i,v)=zero
   10       continue
c
c...calculate concentration factor
            do 20, j=1,nr
               cw(j)=zero
   20       continue
```

```
            cw(i)=one
c......eqn. 5.22
            irrate=ir*year/tgv(v)
            do 30, j=1,nr
               rwvg(j)=(irrate*rv(v)*tbigv(v)/yv(v))*(cw(j)/cw(i))
   30       continue
c......eqn. 5.23
            call reoper(rwvg,cwvh,lambda,lambdw,d,tgv(v),nr)
c......eqn. 5.27
            do 40, j=1,nr
               rwsg(j)=(irrate/ps)*(cw(j)/cw(i))
   40       continue
c......eqn. 5.28
            call roper(rwsg,cwvhs,lambda,d,tgv(v),nr)
c......eqn. 5.31
            do 50, j=1,nr
               crvh(j)=(mlv(v)+bv(v,j))*wv(v)*cwvhs(j)
   50       continue
c......eqn. 5.32
            call aoper(cwvh,cwvh,lambda,d,thv(v),nr)
            call soper(cwvh,cwvh,lambda,d,tcv(v),nr)
            call aoper(crvh,crvh,lambda,d,thv(v),nr)
            call soper(crvh,crvh,lambda,d,tcv(v),nr)
            do 60, j=1,nr
               pptf(j,i,v)=(cwvh(j)+crvh(j))/year
   60       continue
   70 continue
      end
```

F.117 WPHH.FOR

```
      subroutine wphh(maxnr,fhv,sawvh,tcv,pptf)
c
c   this subroutine calculates the pptf for the
c   irrigation water-plant-human pathway for tritium only
c   according to Equations D.5 and D.6
c
      double precision cw,cwvh
      double precision fhv
      double precision nine
      double precision one
      double precision pptf
      double precision sawvh
      double precision tcv
      double precision year
      double precision zero
      integer v
      integer maxnr
      parameter (nine=9.0d+0)
      parameter (one=1.0d+0)
      parameter (year=365.25d+0)
      parameter (zero=0.0d+0)
      dimension fhv(4)
      dimension pptf(maxnr,maxnr,4)
      dimension tcv(4)
c
c...initalize the pptfs
      do 10, v=1,4
         pptf(1,1,v)=zero
c...calculate concentration factor
         cw=one
c...eqn. D.5
         cwvh=nine*fhv(v)*cw*sawvh/cw
c...eqn.D.6
         pptf(1,1,v)=cwvh*tcv(v)/year
   10 continue
      end
```

F.118 WRCHN.FOR

```
c
c     This subroutine writes the chain, cede and activity data
c     to the FORTRAN output file
c
      subroutine wrchn(out,mxnr,mxnip,nr,nip,ichn,
     $    chain,echain,nuc,ipos,
     $    ipar,frac,hlif,prfrac,
     $    ci,dfg,dfh,dfes,dfer,type,ioerr)

      character*6 nuc
      character*6 type
      character*8 chain
      double precision ci
      double precision dfer,dfes,dfg,dfh
      double precision factor,frac
      double precision hlif
      double precision prfrac
      double precision zero
      integer ichn,iip,ipar,ioerr,ipos,ir
      integer mxnip,mxnr
      integer nip,nr
      integer out
      integer tnucs
      logical echain
      parameter(zero=0.0d+0)
      dimension ci(mxnr)
      dimension dfer(mxnr,0:mxnip),dfes(mxnr,0:mxnip)
      dimension dfg(mxnr,0:mxnip),dfh(mxnr,0:mxnip)
      dimension frac(2,mxnr)
      dimension hlif(mxnr)
      dimension ipar(2,mxnr)
      dimension ipos(mxnr)
      dimension nip(mxnr)
      dimension nuc(mxnr,0:mxnip)
      dimension prfrac(mxnr,0:mxnip)


      if (echain) then
         write(out,'(/a,i2,3a)',iostat=ioerr) 'Chain No. ',ichn,': ',
     $      chain,'in equilibrium'
          if (ioerr .ne. 0) then
              call ioreturn(ioerr)
              return
          endif

      else
         write(out,'(/a,i2,2a)',iostat=ioerr) 'Chain No. ',ichn,': ',
     $      chain
          if (ioerr .ne. 0) then
              call ioreturn(ioerr)
              return
          endif

      endif

      write(out,'(/3a)',iostat=ioerr)
     $   'Nuclide   Chain   Half Life    Initial ',
```

```
$    '   First  Fractional Second  Fractional',
$    '   Ingestion   Inhalation    Surface      15 cm '
 if (ioerr .ne. 0) then
     call ioreturn(ioerr)
     return
 endif

 write(out,'(3a)',iostat=ioerr)
$    '        Position                Inventory',
$    '  Parent    Yield    Parent    Yield',
$    '     CEDE Factor CEDE Factor   Dose Rate   Dose Rate'
 if (ioerr .ne. 0) then
     call ioreturn(ioerr)
     return
 endif

 write(out,'(3a)',iostat=ioerr)
$    '                                  ',
$    '                                        ',
$    '                                      Factor      Factor'
 if (ioerr .ne. 0) then
     call ioreturn(ioerr)
     return
 endif

 if (type .eq. 'RESIDE' .or. type .eq. 'RENOVA') then
    write(out,'(3a)',iostat=ioerr)
$       '                      (d)        (pCi/g)',
$       '                                        ',
$       '     (Sv/Bq)     (Sv/Bq)     ((Sv/d)/     ((Sv/d)/'
     if (ioerr .ne. 0) then
         call ioreturn(ioerr)
         return
     endif

    write(out,'(3a)',iostat=ioerr)
$       '                                    ',
$       '                                          ',
$       '                                    (Bq/m**2))  (Bq/m**3))'
     if (ioerr .ne. 0) then
         call ioreturn(ioerr)
         return
     endif

 elseif (type .eq. 'OCCUPA') then

    write(out,'(3a)',iostat=ioerr)
$       '                      (d)        (dpm/   ',
$       '                                          ',
$       '     (Sv/Bq)     (Sv/Bq)     ((Sv/d)/-   ((Sv/d)/'
     if (ioerr .ne. 0) then
         call ioreturn(ioerr)
         return
     endif

    write(out,'(3a)',iostat=ioerr)
$       '                              100 cm^2)',
$       '                                          ',
$       '                            (Bq/m**2))  (Bq/m**3))'
```

```
          if (ioerr .ne. 0) then
              call ioreturn(ioerr)
              return
          endif

      elseif (type .eq. 'DRINK') then

         write(out,'(3a)',iostat=ioerr)
     $      '                        (d)         (pCi)',
     $      '                                          ',
     $      '     (Sv/Bq)     (Sv/Bq)     ((Sv/d)/     ((Sv/d)/'
          if (ioerr .ne. 0) then
              call ioreturn(ioerr)
              return
          endif

         write(out,'(3a)',iostat=ioerr)
     $      '                                    ',
     $      '                                          ',
     $      '                                    (Bq/m**2))  (Bq/m**3))'
          if (ioerr .ne. 0) then
              call ioreturn(ioerr)
              return
          endif

      endif
c
      tnucs = nr
      do 30, ir=1,nr
          do 40, iip=1,nip(ir)
              tnucs = tnucs+1
   40     continue
   30 continue
c
      write(out, '(/a,i2,/)',iostat=ioerr) 'Nuclides in chain : ', tnucs
      if (ioerr .ne. 0) then
          call ioreturn(ioerr)
          return
      endif

      do 20, ir=1,nr

         write(out,'(a,i7,2x,2(1p,e12.4),2(0p,i6,f13.5),
     $    2x,4(1p,e12.2))',iostat=ioerr)
     $      nuc(ir,0),ipos(ir),hlif(ir),
     $      ci(ir),ipar(1,ir),frac(1,ir),ipar(2,ir),frac(2,ir),
     $      dfg(ir,0),dfh(ir,0),dfes(ir,0),dfer(ir,0)
          if (ioerr .ne. 0) then
              call ioreturn(ioerr)
              return
          endif

         do 10, iip=1,nip(ir)

            if (echain) then
               factor=prfrac(ir,iip)
            else
               factor=zero
            endif
```

```
        write(out,'(2a,(1p,e12.4),i6,0p,f13.5,a,4(1p,e12.2))',
  $        iostat=ioerr)
  $        nuc(ir,iip),'  Implicit           ',
  $        ci(ir)*factor,ir,prfrac(ir,iip),
  $        '                    ',
  $        dfg(ir,iip),dfh(ir,iip),dfes(ir,iip),dfer(ir,iip)

          if (ioerr .ne. 0) then
              call ioreturn(ioerr)
              return
          endif

10    continue
20 continue
   end
```

F.119 WRDRI.FOR

```
c
c...this subroutine write out the maximum results for
c   the drinking water scenario
c

      subroutine wrdri(out,mxorg,mxpath,mxlist,npath,nlist,
     $   ttede,tt,period,printed,persv,
     $   nuclid,cwtsv,cswtsv,dossv,nucsv,pathsv,tedemx,
     $   dosmx,tdos,tpdos,pathmx,tpath,tppath,nucmx,tnuc,tpnuc,
     $   hocde,orgnsv,ioerr)

      character*6 nuclid
      double precision cwtsv,cswtsv
      double precision dosmx,dossv
      double precision hocde
      double precision nucmx,nucsv
      double precision orgnsv
      double precision pathmx,pathsv
      double precision ttede,tdos,tedemx,tnuc,tpath,tt
      double precision year
      double precision zero
      integer i,ilist,ioerr,iorg
      integer mxlist,mxorg,mxpath
      integer nlist,npath
      integer out
      integer period,printed,persv,tppath,tpdos,tpnuc
      parameter (year=365.25d+0)
      parameter (zero=0.0d+0)
      dimension cwtsv(mxlist),cswtsv(mxlist)
      dimension dosmx(mxlist,mxpath),dossv(mxlist,mxpath)
      dimension hocde(mxlist,mxorg)
      dimension nucmx(mxlist),nucsv(mxlist)
      dimension nuclid(mxlist)
      dimension orgnsv(mxorg)
      dimension pathmx(mxpath),pathsv(mxpath)
      dimension tdos(mxlist,mxpath)
      dimension tnuc(mxlist)
      dimension tpath(mxpath)
      dimension tppath(mxpath)
      dimension tpdos(mxlist,mxpath)
      dimension tpnuc(mxlist)

      write(out, '(/a)',iostat=ioerr) 'TIMESTEP DATA:'
      if (ioerr .ne. 0) then
          call ioreturn(ioerr)
          return
      endif

      write(out,'(/a,i10)',iostat=ioerr)
     $   'Number of nuclides    : ', nlist
      if (ioerr .ne. 0) then
          call ioreturn(ioerr)
          return
      endif

      write(out,'(a,i10)',iostat=ioerr)
     $   'Number of time steps : ', period
```

```
 if (ioerr .ne. 0) then
     call ioreturn(ioerr)
     return
 endif

 write(out,'(a,i10)',iostat=ioerr)
$   'Number of print steps: ', printed
 if (ioerr .ne. 0) then
     call ioreturn(ioerr)
     return
 endif

 write(out,'(a,i10)',iostat=ioerr)
$   'Timestep of maximum  : ', persv
 if (ioerr .ne. 0) then
     call ioreturn(ioerr)
     return
 endif

 write(out,'(a,e16.10)',iostat=ioerr)
$   'Day of maximum       : ', (ttede+tt)
 if (ioerr .ne. 0) then
     call ioreturn(ioerr)
     return
 endif

 write(out,'(a,e16.10)',iostat=ioerr)
$   'Year of maximum      : ', (ttede+tt)/year
 if (ioerr .ne. 0) then
     call ioreturn(ioerr)
     return
 endif

 write(out,'(//a,i10,a,2(1p,e16.10,a)/a,2(1p,e16.10,a))',
$   iostat=ioerr)
$   'For Period ',persv,': ',
$   ttede,' days  to ',ttede+tt,' days',
$   '                          ',
$   ttede/year,' years to ',(ttede+tt)/year,' years'
 if (ioerr .ne. 0) then
     call ioreturn(ioerr)
     return
 endif

 write(out,'(/a)',iostat=ioerr) 'Dose Components of Maximum TEDE'
 if (ioerr .ne. 0) then
     call ioreturn(ioerr)
     return
 endif

 write(out,'(a)',iostat=ioerr) ' '
 if (ioerr .ne. 0) then
     call ioreturn(ioerr)
     return
 endif

 write(out,'(4(a/),a)',iostat=ioerr)
$   'Nuclide     Total       Ground      Ground',
$   '            Dose        Water       Water',
```

```
     $   '                          Activity    Average',
     $   '                                      Activity',
     $   '            (mrem/y)      (pCi/L)     (pCi/L)'
      if (ioerr .ne. 0) then
          call ioreturn(ioerr)
          return
      endif

      write(out,'(/a/)',iostat=ioerr) 'Dose Components'
      if (ioerr .ne. 0) then
          call ioreturn(ioerr)
          return
      endif

      do 10, ilist=1,nlist
CCC...doses and activities should never be negative;
CCC...so zero them out if need be
         if (nucsv(ilist) .lt. 0) nucsv(ilist) = 0
         if (cwtsv(ilist) .lt. 0) cwtsv(ilist) = 0
         if (cswtsv(ilist) .lt. 0) cswtsv(ilist) = 0

         write(out,'(a10,3(1p,e12.5))',iostat=ioerr) nuclid(ilist),
     $      nucsv(ilist), cwtsv(ilist), cswtsv(ilist)

          if (ioerr .ne. 0) then
              call ioreturn(ioerr)
              return
          endif

   10 continue

      write(out,'(54a)',iostat=ioerr) ('-',i=1,54)
      if (ioerr .ne. 0) then
          call ioreturn(ioerr)
          return
      endif

      write(out,'(a10,1p,18e12.5)',iostat=ioerr)
     $   'Totals  ', tedemx
      if (ioerr .ne. 0) then
          call ioreturn(ioerr)
          return
      endif

      write(out, '(//a)',iostat=ioerr) ' '
      if (ioerr .ne. 0) then
          call ioreturn(ioerr)
          return
      endif

c...write out organ doses at maximum TEDE
      write(out,'(/a)',iostat=ioerr)
     $   'Organ Doses at Maximum TEDE (mrem/y)'
      if (ioerr .ne. 0) then
          call ioreturn(ioerr)
          return
      endif

      write(out,'(/2a)',iostat=ioerr)
```

```
     $   'Nuclide   Adrenals    Bld Wall   B_Surface       Brain',
     $   '       Breast       S_Wall      SI_Wall'
      if (ioerr .ne. 0) then
          call ioreturn(ioerr)
          return
      endif

      do 20, ilist=1,nlist
         write(out,'(a,1p,7e12.5)',iostat=ioerr)
     $      nuclid(ilist),(hocde(ilist,iorg),iorg=1,7)

          if (ioerr .ne. 0) then
              call ioreturn(ioerr)
              return
          endif

   20 continue

      write(out,'(90a)',iostat=ioerr) ('-',i=1,90)
      if (ioerr .ne. 0) then
          call ioreturn(ioerr)
          return
      endif

      write(out,'(a,1p,7e12.5)',iostat=ioerr)
     $   'Totals',(orgnsv(iorg),iorg=1,7)

      if (ioerr .ne. 0) then
          call ioreturn(ioerr)
          return
      endif

      write(out,'(/2a)',iostat=ioerr)
     $   'Nuclide   ULI_Wall    LLI_Wall     Kidneys       Liver',
     $   '       Lungs     Ovaries    Pancreas'
      if (ioerr .ne. 0) then
          call ioreturn(ioerr)
          return
      endif

      do 30, ilist=1,nlist
         write(out,'(a,1p,7e12.5)',iostat=ioerr)
     $      nuclid(ilist),(hocde(ilist,iorg),iorg=8,14)

          if (ioerr .ne. 0) then
              call ioreturn(ioerr)
              return
          endif

   30 continue

      write(out,'(90a)',iostat=ioerr) ('-',i=1,90)
      if (ioerr .ne. 0) then
          call ioreturn(ioerr)
          return
      endif

      write(out,'(a,1p,7e12.5)',iostat=ioerr)
     $   'Totals',(orgnsv(iorg),iorg=8,14)
```

```
   if (ioerr .ne. 0) then
       call ioreturn(ioerr)
       return
   endif

   write(out,'(/2a)',iostat=ioerr)
  $   'Nuclide   R_Marrow        Skin       Spleen       Testes',
  $   '       Thymus     Thyroid      Uterus'
   if (ioerr .ne. 0) then
       call ioreturn(ioerr)
       return
   endif

   do 40, ilist=1,nlist
      write(out,'(a,1p,7e12.5)',iostat=ioerr)
  $      nuclid(ilist),(hocde(ilist,iorg),iorg=15,21)

       if (ioerr .ne. 0) then
           call ioreturn(ioerr)
           return
       endif

40 continue

   write(out,'(90a)',iostat=ioerr) ('-',i=1,90)
   if (ioerr .ne. 0) then
       call ioreturn(ioerr)
       return
   endif

   write(out,'(a,1p,7e12.5)',iostat=ioerr)
  $   'Totals',(orgnsv(iorg),iorg=15,21)
   if (ioerr .ne. 0) then
       call ioreturn(ioerr)
       return
   endif

   write(out, '(/a)',iostat=ioerr) 'End of Organ Doses'
   if (ioerr .ne. 0) then
       call ioreturn(ioerr)
       return
   endif

   write(out,'(/a)',iostat=ioerr) ' '
   if (ioerr .ne. 0) then
       call ioreturn(ioerr)
       return
   endif

   write(out,'(a/)',iostat=ioerr)
  $   'Component Maximums and Time of Occurrence'
   if (ioerr .ne. 0) then
       call ioreturn(ioerr)
       return
   endif

   write(out,'(a)',iostat=ioerr)
  $   '   Type    Time   Maximum    Time of Occurrence'
   if (ioerr .ne. 0) then
```

```
        call ioreturn(ioerr)
        return
    endif

    write(out,'(a)',iostat=ioerr)
   $   '            Step   Value       (days)      (years)'
    if (ioerr .ne. 0) then
        call ioreturn(ioerr)
        return
    endif

    write(out,'(a,i5,3(1p,e12.5))',iostat=ioerr)
   $ '   TEDE   ',persv,tedemx,
   $        ttede,ttede/year
    if (ioerr .ne. 0) then
        call ioreturn(ioerr)
        return
    endif

    do 60, ilist=1,nlist
       write(out,'(3a,i5,3(1p,e12.5),a)',iostat=ioerr)
   $       '  ',nuclid(ilist),'  ',tpnuc(ilist), nucmx(ilist),
   $      tnuc(ilist), tnuc(ilist)/year,'  Total Radionuclide Dose'

        if (ioerr .ne. 0) then
            call ioreturn(ioerr)
            return
        endif

 60 continue

    write(out, '(//a)',iostat=ioerr) ' '
    if (ioerr .ne. 0) then
        call ioreturn(ioerr)
        return
    endif
    end
```

F.120 WRHIST.FOR

```
c
c...this subroutine writes to the history file. this file can be read
c   later and used for plotting
c

      subroutine wrhist(his,mxlist,mxpath,nlist,npath,
     $   tcalc,tcalc_end,list,con1,con2,dosup,pathup,nucup,tedeup,
     $   ioerr)

      character*6 list
      double precision con1,con2
      double precision dosup
      double precision nucup
      double precision pathup
      double precision tcalc,tedeup,tcalc_end
      integer his
      integer ilist,ioerr,ipath
      integer mxlist,mxpath
      integer nlist,npath
      dimension con1(mxlist),con2(mxlist)
      dimension dosup(mxlist,mxpath)
      dimension list(mxlist)
      dimension nucup(mxlist)
      dimension pathup(mxpath)
c
c...write number of listed radionuclides and number of pathways
      write(his,*,iostat=ioerr) nlist,npath
      if (ioerr .ne. 0) then
          call ioreturn(ioerr)
          return
      endif

c
c...write time at beginning of the calculation interval
      write(his,*,iostat=ioerr) tcalc,tcalc_end
      if (ioerr .ne. 0) then
          call ioreturn(ioerr)
          return
      endif

c
c...write doses per radionuclide per pathway and radionuclide
      do 10, ilist=1,nlist
         write(his,'(a,1p,10e24.10e3)',iostat=ioerr) list(ilist),
     $    (dosup(ilist,ipath),ipath=1,npath),nucup(ilist),
     $    con1(ilist), con2(ilist)
          if (ioerr .ne. 0) then
              call ioreturn(ioerr)
              return
          endif

   10 continue
c
c...write pathway doses and total doses
      write(his,'(a,1p,8e24.10e3)',iostat=ioerr)
     $   'Total                                          ',
     $   (pathup(ipath),ipath=1,npath),tedeup
```

```
if (ioerr .ne. 0) then
    call ioreturn(ioerr)
    return
endif

end
```

F.121 WROUT.FOR

```
c
c...this subroutine writes out the maximum results
c   for the renovation and occupancy scenarios
c

      subroutine wrout(out,mxpath,mxlist,npath,nlist,short,
     $   ttede,tt,period,printed,persv,
     $   nuclid,s,dossv,nucsv,pathsv,tedemx,
     $   dosmx,tdos,tpdos,pathmx,tpath,tppath,nucmx,tnuc,tpnuc,ioerr)

      character*6 short
      character*6 nuclid
      double precision dosmx,dossv
      double precision nucmx,nucsv
      double precision pathmx,pathsv
      double precision s
      double precision ttede,tdos,tedemx,tnuc,tpath,tt
      double precision year
      double precision zero
      integer i,ilist,ioerr,ipath
      integer mxlist,mxpath
      integer nlist,npath
      integer out
      integer period,printed,persv,tppath,tpdos,tpnuc
      parameter (year=365.25d+0)
      parameter (zero=0.0d+0)
      dimension dosmx(mxlist,mxpath),dossv(mxlist,mxpath)
      dimension nucmx(mxlist),nucsv(mxlist)
      dimension nuclid(mxlist)
      dimension pathmx(mxpath),pathsv(mxpath)
      dimension s(mxlist)
      dimension tdos(mxlist,mxpath)
      dimension tnuc(mxlist)
      dimension tpath(mxpath)
      dimension tppath(mxpath)
      dimension tpdos(mxlist,mxpath)
      dimension tpnuc(mxlist)

      write(out, '(/a)',iostat=ioerr) 'TIMESTEP DATA:'
      if (ioerr .ne. 0) then
          call ioreturn(ioerr)
          return
      endif

      write(out,'(a,i10)',iostat=ioerr)
     $   'Number of nuclides   : ',nlist
      if (ioerr .ne. 0) then
          call ioreturn(ioerr)
          return
      endif

      write(out,'(a,i10)',iostat=ioerr)
     $   'Number of time steps : ', period
      if (ioerr .ne. 0) then
          call ioreturn(ioerr)
          return
      endif
```

```
 write(out,'(a,i10)',iostat=ioerr)
$   'Number of print steps: ', printed
 if (ioerr .ne. 0) then
     call ioreturn(ioerr)
     return
 endif

 write(out,'(a,i10)',iostat=ioerr)
$   'Timestep of maximum  : ', persv
 if (ioerr .ne. 0) then
     call ioreturn(ioerr)
     return
 endif

 write(out,'(a,e16.10)',iostat=ioerr)
$   'Day of maximum       : ', ttede+tt
 if (ioerr .ne. 0) then
     call ioreturn(ioerr)
     return
 endif

 write(out,'(a,e16.10)',iostat=ioerr)
$   'Year of maximum      : ', (ttede+tt)/year
 if (ioerr .ne. 0) then
     call ioreturn(ioerr)
     return
 endif


 write(out,'(//a,i10,a,2(1p,e16.10,a)/a,2(1p,e16.10,a))',
$   iostat=ioerr)
$   'For Period ',persv,': ',
$   ttede,' days  to ',ttede+tt,' days',
$   '                          ',
$   ttede/year,' years to ',(ttede+tt)/year,' years'
 if (ioerr .ne. 0) then
     call ioreturn(ioerr)
     return
 endif

 write(out,'(/a)',iostat=ioerr) 'Dose Components of Maximum TEDE'
 if (ioerr .ne. 0) then
     call ioreturn(ioerr)
     return
 endif

 write(out,'(a)',iostat=ioerr) ' '
 if (ioerr .ne. 0) then
     call ioreturn(ioerr)
     return
 endif


 if (short .eq. 'RENOVA') then
    write(out,'(3(2a/),2a/)',iostat=ioerr)
$       'Nuclide     External    Inhalation  Ingestion   Total',
$       '       Average',
$       '            Dose        Dose        Dose        Dose  ',
$       '       Activity',
```

```
     $       '              (mrem/y)     (mrem/y)     (mrem/y)     (mrem/y)',
     $       '        (pCi/g)',
     $       ' ',
     $       ' '
          if (ioerr .ne. 0) then
              call ioreturn(ioerr)
              return
          endif

      elseif (short .eq. 'OCCUPA') then
         write(out,'(3(2a/),2a/)',iostat=ioerr)
     $       'Nuclide     External    Inhalation  Ingestion   Total',
     $       '        Average',
     $       '              Dose        Dose        Dose        Dose  ',
     $       '       Activity',
     $       '              (mrem/y)     (mrem/y)     (mrem/y)     (mrem/y)',
     $       '     (dpm/100 cm^2)',
     $       '                                                          ',
     $       ' '



          if (ioerr .ne. 0) then
              call ioreturn(ioerr)
              return
          endif

      endif

      write(out,'(a/)',iostat=ioerr) 'Dose Components'
      if (ioerr .ne. 0) then
          call ioreturn(ioerr)
          return
      endif


      do 10, ilist=1,nlist
CCC...doses and activities should never be negative;
CCC...so zero them out if need be
         do 30, ipath=1,npath
              if (dossv(ilist,ipath) .lt. 0) dossv(ilist,ipath) = 0
   30    continue
         if (nucsv(ilist) .lt. 0) nucsv(ilist) = 0
         if (s(ilist) .lt. 0) s(ilist) = 0

         write(out,'(a10,5(1p,e12.5))',iostat=ioerr) nuclid(ilist),
     $       (dossv(ilist,ipath),ipath=1,npath),nucsv(ilist),s(ilist)
          if (ioerr .ne. 0) then
              call ioreturn(ioerr)
              return
          endif

   10 continue

      write(out,'(67a)',iostat=ioerr) ('-',i=1,67)
      if (ioerr .ne. 0) then
          call ioreturn(ioerr)
          return
```

```
      endif

      write(out,'(a10,5(1p,e12.5))',iostat=ioerr)
     $   'Totals  ',(pathsv(ipath),ipath=1,npath),tedemx
      if (ioerr .ne. 0) then
          call ioreturn(ioerr)
          return
      endif

c
      write(out,'(/a)',iostat=ioerr) ' '
      if (ioerr .ne. 0) then
          call ioreturn(ioerr)
          return
      endif

      write(out,'(a/)',iostat=ioerr)
     $   'Component Maximums and Time of Occurrence'
      if (ioerr .ne. 0) then
          call ioreturn(ioerr)
          return
      endif

      write(out,'(a)',iostat=ioerr)
     $   '   Type    Time   Maximum    Time of Occurrence'
      if (ioerr .ne. 0) then
          call ioreturn(ioerr)
          return
      endif

      write(out,'(a)',iostat=ioerr)
     $   '           Step   Value      (days)      (years)'
      if (ioerr .ne. 0) then
          call ioreturn(ioerr)
          return
      endif

      write(out,'(a,i5,3(1p,e12.5))',iostat=ioerr)
     $   ' External ',tppath(1),pathmx(1),tpath(1),tpath(1)/year
      if (ioerr .ne. 0) then
          call ioreturn(ioerr)
          return
      endif

      write(out,'(a,i5,3(1p,e12.5))',iostat=ioerr)
     $   'Inhalation',tppath(2),pathmx(2),tpath(2),tpath(2)/year
      if (ioerr .ne. 0) then
          call ioreturn(ioerr)
          return
      endif

      write(out,'(a,i5,3(1p,e12.5))',iostat=ioerr)
     $   'Ingestion ',tppath(3),pathmx(3),tpath(3),tpath(3)/year
      if (ioerr .ne. 0) then
          call ioreturn(ioerr)
          return
      endif

      write(out,'(a,i5,3(1p,e12.5))',iostat=ioerr)
```

```
$    '    TEDE    ',persv,tedemx,ttede,
$     ttede/year
 if (ioerr .ne. 0) then
     call ioreturn(ioerr)
     return
 endif

 do 20, ilist=1,nlist
    write(out,'(3a,i5,3(1p,e12.5),a)',iostat=ioerr)
$       '  ',nuclid(ilist),'  ',tpdos(ilist,1),dosmx(ilist,1),
$       tdos(ilist,1),
$       tdos(ilist,1)/year,'  Maximum External'
      if (ioerr .ne. 0) then
          call ioreturn(ioerr)
          return
      endif

    write(out,'(3a,i5,3(1p,e12.5),a)',iostat=ioerr)
$       '  ',nuclid(ilist),'  ',tpdos(ilist,2),dosmx(ilist,2),
$       tdos(ilist,2),
$       tdos(ilist,2)/year,'  Maximum Inhalation'
      if (ioerr .ne. 0) then
          call ioreturn(ioerr)
          return
      endif

    write(out,'(3a,i5,3(1p,e12.5),a)',iostat=ioerr)
$       '  ',nuclid(ilist),'  ',tpdos(ilist,3),dosmx(ilist,3),
$       tdos(ilist,3),
$       tdos(ilist,3)/year,'  Maximum Ingestion'
      if (ioerr .ne. 0) then
          call ioreturn(ioerr)
          return
      endif

    write(out,'(3a,i5,3(1p,e12.5),a)',iostat=ioerr)
$       '  ',nuclid(ilist),'  ',tpnuc(ilist),nucmx(ilist),
$       tnuc(ilist),tnuc(ilist)/year,'  Maximum Nuclide'
      if (ioerr .ne. 0) then
          call ioreturn(ioerr)
          return
      endif

20 continue

 write(out, '(//a)',iostat=ioerr) ' '
 if (ioerr .ne. 0) then
     call ioreturn(ioerr)
     return
 endif

 end
```

F.122 WRRES.FOR

```
c
c...this subroutine write out the maximum results
c   for the residential scenario
c

      subroutine wrres(out,mxpath,mxlist,npath,nlist,
     $   ttede,tt,period,printed,persv,
     $   nuclid,astsv,awtsv,dossv,nucsv,pathsv,tedemx,
     $   dosmx,tdos,tpdos,pathmx,tpath,tppath,nucmx,tnuc,tpnuc,ioerr)

      character*6 nuclid
      double precision astsv,awtsv
      double precision dosmx,dossv
      double precision nucmx,nucsv
      double precision pathmx,pathsv
      double precision ttede,tdos,tedemx,tnuc,tpath,tt
      double precision year
      double precision zero
      integer i,ilist,ioerr,ipath
      integer mxlist,mxpath
      integer nlist,npath
      integer out
      integer period,printed,persv,tppath,tpdos,tpnuc
      parameter (year=365.25d+0)
      parameter (zero=0.0d+0)
      dimension astsv(mxlist),awtsv(mxlist)
      dimension dosmx(mxlist,mxpath),dossv(mxlist,mxpath)
      dimension nucmx(mxlist),nucsv(mxlist)
      dimension nuclid(mxlist)
      dimension pathmx(mxpath),pathsv(mxpath)
      dimension tdos(mxlist,mxpath)
      dimension tnuc(mxlist)
      dimension tpath(mxpath)
      dimension tppath(mxpath)
      dimension tpdos(mxlist,mxpath)
      dimension tpnuc(mxlist)


      write(out, '(/a)',iostat=ioerr) 'TIMESTEP DATA:'
      if (ioerr .ne. 0) then
        call ioreturn(ioerr)
        return
      endif

      write(out,'(/a,i10)',iostat=ioerr)
     $   'Number of nuclides    : ', nlist
      if (ioerr .ne. 0) then
        call ioreturn(ioerr)
        return
      endif

      write(out,'(a,i10)',iostat=ioerr)
     $   'Number of time steps : ', period
      if (ioerr .ne. 0) then
        call ioreturn(ioerr)
        return
      endif
```

```
 write(out,'(a,i10)',iostat=ioerr)
$   'Number of print steps: ', printed
 if (ioerr .ne. 0) then
   call ioreturn(ioerr)
   return
 endif

 write(out,'(a,i10)',iostat=ioerr)
$   'Timestep of maximum  : ', persv
 if (ioerr .ne. 0) then
   call ioreturn(ioerr)
   return
 endif

 write(out,'(a,e16.10)',iostat=ioerr)
$   'Day of maximum       : ', ttede+tt
 if (ioerr .ne. 0) then
   call ioreturn(ioerr)
   return
 endif

 write(out,'(a,e16.10)',iostat=ioerr)
$   'Year of maximum      : ', (ttede+tt)/year
 if (ioerr .ne. 0) then
   call ioreturn(ioerr)
   return
 endif



 write(out,'(//a,i10,a,2(1p,e16.10,a)/a,2(1p,e16.10,a))',
$   iostat=ioerr)
$   'For Period ',persv,': ',
$   ttede,' days  to ',ttede+tt,' days',
$   '                          ',
$   ttede/year,' years to ',(ttede+tt)/year,' years'
 if (ioerr .ne. 0) then
   call ioreturn(ioerr)
   return
 endif


 write(out,'(/a)',iostat=ioerr) 'Dose Components of Maximum TEDE'
 if (ioerr .ne. 0) then
   call ioreturn(ioerr)
   return
 endif

 write(out,'(a)',iostat=ioerr) ' '
 if (ioerr .ne. 0) then
   call ioreturn(ioerr)
   return
 endif


 write(out,'(4(3a/))',iostat=ioerr)
$   'Nuclide     External    Inhalation  Agricult.   Soil      ',
$   '   Drinking    Irrigated   Aquatic     Total  ',
$   '   Soil        Water Ave. ',
```

```
     $   '              Dose         Dose         Food         Ingestion',
     $   '   Water       Food         Food         Nuclide',
     $   '   Activity    Activity',
     $   '                                       Ingestion   Dose      ',
     $   '   Ingestion   Ingestion   Ingestion   Dose                  ',
     $   '              ',
     $   '                                       Dose                  ',
     $   '   Dose        Dose        Dose                  ',
     $   '                                 ',
     $   '              (mrem/y)     (mrem/y)     (mrem/y)     (mrem/y)',
     $   '   (mrem/y)     (mrem/y)     (mrem/y)     (mrem/y)',
     $   '   (pCi/g)      (pCi/l)   '
      if (ioerr .ne. 0) then
        call ioreturn(ioerr)
        return
      endif


      write(out,'(/a/)',iostat=ioerr) 'Dose Components'
      if (ioerr .ne. 0) then
        call ioreturn(ioerr)
        return
      endif

      do 10, ilist=1,nlist
CCC...doses and activities should never be negative;
CCC...so zero them out if need be

         do 30, ipath=1,npath
              if (dossv(ilist,ipath) .lt. 0) dossv(ilist,ipath) = 0
   30    continue

         if (nucsv(ilist) .lt. 0) nucsv(ilist) = 0
         if (astsv(ilist) .lt. 0) astsv(ilist) = 0
         if (awtsv(ilist) .lt. 0) awtsv(ilist) = 0


      write(out,'(a10,10(1p,e12.5))',iostat=ioerr) nuclid(ilist),
     $      (dossv(ilist,ipath),ipath=1,npath),nucsv(ilist),
     $      astsv(ilist),awtsv(ilist)
          if (ioerr .ne. 0) then
              call ioreturn(ioerr)
              return
          endif

   10 continue

      write(out,'(126a)',iostat=ioerr) ('-',i=1,126)
      if (ioerr .ne. 0) then
        call ioreturn(ioerr)
        return
      endif

      write(out,'(a10,8(1p,e12.5))',iostat=ioerr)
     $   'Totals  ',
     $   (pathsv(ipath),ipath=1,npath),tedemx
      if (ioerr .ne. 0) then
        call ioreturn(ioerr)
        return
```

```
 endif

 write(out,'(/a)',iostat=ioerr) ' '
 if (ioerr .ne. 0) then
   call ioreturn(ioerr)
   return
 endif

 write(out,'(a/)',iostat=ioerr)
$   'Component Maximums and Time of Occurrence'
 if (ioerr .ne. 0) then
   call ioreturn(ioerr)
   return
 endif

 write(out,'(a)',iostat=ioerr)
$   '   Type    Time   Maximum    Time of Occurrence'
 if (ioerr .ne. 0) then
   call ioreturn(ioerr)
   return
 endif

 write(out,'(a)',iostat=ioerr)
$   '            Step   Value      (days)      (years)'
 if (ioerr .ne. 0) then
   call ioreturn(ioerr)
   return
 endif

 write(out,'(a,i5,3(1p,e12.5))',iostat=ioerr)
$   ' External ',tppath(1),pathmx(1),tpath(1),tpath(1)/year
 if (ioerr .ne. 0) then
   call ioreturn(ioerr)
   return
 endif


 write(out,'(a,i5,3(1p,e12.5))',iostat=ioerr)
$   'Inhalation',tppath(2),pathmx(2),tpath(2),tpath(2)/year
 if (ioerr .ne. 0) then
   call ioreturn(ioerr)
   return
 endif


 write(out,'(a,i5,3(1p,e12.5))',iostat=ioerr)
$   'Agricultur',tppath(3),pathmx(3),tpath(3),tpath(3)/year
 if (ioerr .ne. 0) then
   call ioreturn(ioerr)
   return
 endif


 write(out,'(a,i5,3(1p,e12.5))',iostat=ioerr)
$   '   Soil    ',tppath(4),pathmx(4),tpath(4),tpath(4)/year
 if (ioerr .ne. 0) then
   call ioreturn(ioerr)
   return
```

```
 endif

 write(out,'(a,i5,3(1p,e12.5))',iostat=ioerr)
$    ' Drinking ',tppath(5),pathmx(5),tpath(5),tpath(5)/year
 if (ioerr .ne. 0) then
   call ioreturn(ioerr)
   return
 endif

 write(out,'(a,i5,3(1p,e12.5))',iostat=ioerr)
$    'Irrigated ',tppath(6),pathmx(6),tpath(6),tpath(6)/year
 if (ioerr .ne. 0) then
   call ioreturn(ioerr)
   return
 endif

 write(out,'(a,i5,3(1p,e12.5))',iostat=ioerr)
$    '  Aquatic ',tppath(7),pathmx(7),tpath(7),tpath(7)/year
 if (ioerr .ne. 0) then
   call ioreturn(ioerr)
   return
 endif

 write(out,'(a,i5,3(1p,e12.5))',iostat=ioerr)
$    '    TEDE    ',persv,tedemx,ttede,ttede/year
 if (ioerr .ne. 0) then
   call ioreturn(ioerr)
   return
 endif

 do 20, ilist=1,nlist

 write(out,'(3a,i5,3(1p,e12.5),a)',iostat=ioerr)
$       '  ',nuclid(ilist),'  ',tpdos(ilist,1),dosmx(ilist,1),
$       tdos(ilist,1),tdos(ilist,1)/year,'  Maximum External'
     if (ioerr .ne. 0) then
         call ioreturn(ioerr)
         return
     endif

 write(out,'(3a,i5,3(1p,e12.5),a)',iostat=ioerr)
$       '  ',nuclid(ilist),'  ',tpdos(ilist,2),dosmx(ilist,2),
$       tdos(ilist,2),
$       tdos(ilist,2)/year,'  Maximum Inhalation'
     if (ioerr .ne. 0) then
         call ioreturn(ioerr)
         return
     endif

 write(out,'(3a,i5,3(1p,e12.5),a)',iostat=ioerr)
$       '  ',nuclid(ilist),'  ',tpdos(ilist,3),dosmx(ilist,3),
$       tdos(ilist,3),
$       tdos(ilist,3)/year,'  Maximum Agricultural Food Ingestion'
```

```
      if (ioerr .ne. 0) then
          call ioreturn(ioerr)
          return
      endif


   write(out,'(3a,i5,3(1p,e12.5),a)',iostat=ioerr)
  $       '  ',nuclid(ilist),'  ',tpdos(ilist,4),dosmx(ilist,4),
  $       tdos(ilist,4),
  $       tdos(ilist,4)/year,'  Maximum Soil Ingestion'
      if (ioerr .ne. 0) then
          call ioreturn(ioerr)
          return
      endif


   write(out,'(3a,i5,3(1p,e12.5),a)',iostat=ioerr)
  $       '  ',nuclid(ilist),'  ',tpdos(ilist,5),dosmx(ilist,5),
  $       tdos(ilist,5),
  $       tdos(ilist,5)/year,'  Maximum Drinking Water Ingestion'
      if (ioerr .ne. 0) then
          call ioreturn(ioerr)
          return
      endif


   write(out,'(3a,i5,3(1p,e12.5),a)',iostat=ioerr)
  $       '  ',nuclid(ilist),'  ',tpdos(ilist,6),dosmx(ilist,6),
  $       tdos(ilist,6),
  $       tdos(ilist,6)/year,'  Maximum Irrigated Food Ingestion'
      if (ioerr .ne. 0) then
          call ioreturn(ioerr)
          return
      endif


   write(out,'(3a,i5,3(1p,e12.5),a)',iostat=ioerr)
  $       '  ',nuclid(ilist),'  ',tpdos(ilist,7),dosmx(ilist,7),
  $       tdos(ilist,7),
  $       tdos(ilist,7)/year,'  Maximum Aquatic Food Ingestion'
      if (ioerr .ne. 0) then
          call ioreturn(ioerr)
          return
      endif


   write(out,'(3a,i5,3(1p,e12.5),a)',iostat=ioerr)
  $       '  ',nuclid(ilist),'  ',tpnuc(ilist),nucmx(ilist),
  $       tnuc(ilist),tnuc(ilist)/year,'  Maximum Nuclide'
      if (ioerr .ne. 0) then
          call ioreturn(ioerr)
          return
      endif
20 continue


   write(out, '(//a)',iostat=ioerr) ' '
   if (ioerr .ne. 0) then
     call ioreturn(ioerr)
```

```
  return
endif
end
```

Appendix F FORTRAN Source Code F-1
F.1 ACTTST.FOR F-1
F.2 ADJ.FOR F-2
F.3 ADOPER.FOR F-3
F.4 AGRICL.FOR F-4
F.5 AOPER.FOR F-5
F.6 AQUATC.FOR F-6
F.7 AROPER.FOR F-7
F.8 BLDDEF.FOR F-8
F.9 BOXSET.FOR F-9
F.10 C1MEXP F-11
F.11 CDECL.INC F-12
F.12 CLOSE.FOR F-13
F.13 COEFF.FOR F-14
F.14 CONCEN.FOR F-17
F.15 CONCLC.FOR F-19
F.16 CONCRD.FOR F-20
F.17 CONNCT.FOR F-24
F.18 CONRES.FOR F-25
F.19 CONTST.FOR F-26
F.20 CONVRT.FOR F-28
F.21 CORH.FOR F-29
F.22 CUBIC.FOR F-30
F.23 D1MEXP.FOR F-31
F.24 DDARAY.FOR F-32
F.25 DFOCON.FOR F-33
F.26 DLSODES.FOR F-34
F.27 DOSMAX.FOR F-128
F.28 DRIDEF.FOR F-129
F.29 DRIDOS.FOR F-130
F.30 DRIMAX.FOR F-134
F.31 DRINK.FOR F-144
F.32 DRIOUT.FOR F-145
F.33 DRMAIN.FOR F-153
F.34 DTMEXP.FOR F-167
F.35 EQUIL.FOR F-168
F.36 EXTERN.FOR F-169
F.37 FRACT1.FOR F-170
F.38 FRCTNL.FOR F-171
F.39 GEOPER.FOR F-172
F.40 GOPER.FOR F-173
F.41 INHALE.FOR F-174
F.42 INIT0.FOR F-175
F.43 INITLG.FOR F-176
F.44 INITZC.FOR F-177
F.45 INITZD.FOR F-178
F.46 INPUT.FOR F-179
F.47 IORETINT.FOR F-181
F.48 IORETURN.FOR F-182
F.49 IRIGAT.FOR F-183
F.50 ISANUC.FOR F-184
F.51 JACOBN.FOR F-185
F.52 KCOEF.FOR F-186
F.53 KDCOEF.FOR F-187
F.54 MVMLTC.FOR F-188
F.55 NATURL.FOR F-189
F.56 NEWLAM.FOR F-190
F.57 NUCIT.FOR F-191
F.58 NUCLIS.FOR F-192
F.59 NUCMAX.FOR F-193

F.60 NUCSUM.FOR F-194
F.61 OCCDOS.FOR F-195
F.62 OCCMAIN.FOR F-197
F.63 OCCMAX.FOR F-206
F.64 OCCOUT.FOR F-213
F.65 PF.FOR F-219
F.66 PTHMAX.FOR F-222
F.67 PTHSUM.FOR F-223
F.68 RDBIO.FOR F-224
F.69 RDCEDE.FOR F-226
F.70 RDCHN.FOR F-230
F.71 RDCONC.FOR F-234
F.72 RDDFO.FOR F-236
F.73 RDKD.FOR F-239
F.74 RDLIST.FOR F-241
F.75 RDTRAN.FOR F-243
F.76 RENDEF.FOR F-245
F.77 RENDOS.FOR F-246
F.78 RENMAIN.FOR F-248
F.79 RENMAX.FOR F-257
F.80 RENOUT.FOR F-263
F.81 REOPER.FOR F-269
F.82 RESDEF.FOR F-270
F.83 RESDOS.FOR F-272
F.84 RESIDE.FOR F-277
F.85 RESMAX.FOR F-292
F.86 RESOUT.FOR F-304
F.87 ROPER.FOR F-341
F.88 RPTTIM.FOR F-342
F.89 SDOPER.FOR F-343
F.90 SOIL.FOR F-344
F.91 SOLVDE.FOR F-345
F.92 SOPER.FOR F-347
F.93 SPAH.FOR F-348
F.94 SPAHC.FOR F-350
F.95 SPAHH.FOR F-352
F.96 SPARSE.FOR F-353
F.97 SPH.FOR F-355
F.98 SPHH.FOR F-357
F.99 SROPER.FOR F-358
F.100 TEDMAX.FOR F-359
F.101 TEDMXD.FOR F-360
F.102 TEDMXR.FOR F-361
F.103 TEDSUM.FOR F-362
F.104 TOLER.FOR F-363
F.105 TRNSFR.FOR F-364
F.106 UPDOSE.FOR F-366
F.107 UPNUC.FOR F-367
F.108 UPPATH.FOR F-368
F.109 UPSBAR.FOR F-369
F.110 UPTEDE.FOR F-370
F.111 UPTIME.FOR F-371
F.112 VALTST.FOR F-372
F.113 WPAH.FOR F-373
F.114 WPAHC.FOR F-376
F.115 WPAHH.FOR F-378
F.116 WPH.FOR F-380
F.117 WPHH.FOR F-382
F.118 WRCHN.FOR F-383
F.119 WRDRI.FOR F-387
F.120 WRHIST.FOR F-393

F.121 WROUT.FOR F-395
F.122 WRRES.FOR F-400

C FORM 335
9)
CM 1102,
I, 3202

U.S. NUCLEAR REGULATORY COMMISSION

BIBLIOGRAPHIC DATA SHEET

(See instructions on the reverse)

1. REPORT NUMBER
(Assigned by NRC, Add Vol., Supp., Rev., and Addendum Numbers, if any.)

NUREG/CR-5512
Vol. 2
Draft

TITLE AND SUBTITLE

Residual Radioactive Contamination From Decommissioning

User's Manual
Draft Report

3. DATE REPORT PUBLISHED

MONTH	YEAR
May	1999

4. FIN OR GRANT NUMBER

W6227

AUTHOR(S)

M.A. Wernig, Interna Environmental Services, Inc.
A.M. Tomasi, F.A. Duran, Sandia National Laboratories
C.D. Updegraff, GRAM, Inc.

6. TYPE OF REPORT

Technical

7. PERIOD COVERED *(Inclusive Dates)*

PERFORMING ORGANIZATION - NAME AND ADDRESS *(If NRC, provide Division, Office or Region, U.S. Nuclear Regulatory Commission, and mailing address; if contractor, provide name and mailing address.)*

Sandia National Laboratories
Albuquerque, NM 87185

Interna Environmental Services, Inc.
1650 University Blvd. NE Ste 300
Albuquerque, NM 87112

GRAM, Inc.
8500 Menual Blvd. NE Ste B-335
Albuquerque, NM 87112

SPONSORING ORGANIZATION - NAME AND ADDRESS *(If NRC, type "Same as above"; if contractor, provide NRC Division, Office or Region, U.S. Nuclear Regulatory Commission, and mailing address.)*

Division of Risk Analysis and Application
Office of Nuclear Reactor Research
U.S. Nuclear Regulatory Commission
Washington, DC 20555-0001

. SUPPLEMENTARY NOTES

C. Dailey, NRC Project Manager

. ABSTRACT *(200 words or less)*

The Decontamination and Decommissioning (DandD) software package developed by Sandia National Laboratories for the Nuclear Regulatory Commission (NRC) provides a user-friendly analytical tool to address the technical dose criteria contained in NRC's Radiological Criteria for License Termination rule (10 CFR 20 Subpart E; NRC, 1997). Specifically, DandD embodies the NRC's screening methodology to allow licensees to convert residual radioactivity contamination levels at their site to annual dose, in a way consistent with both 10 CFR Part 20 and the corresponding implementation guidance currently under development by NRC. The screening methodology employs reasonably conservative scenarios, fate and transport models, and default parameter values that have been developed to allow the NRC to qualitatively estimate the risk of terminating a license given only information about the level of contamination. A licensee has the option of specifying only the level of contamination and running the code with the default parameter values or, in the case where site specific information is available, applying site-specific parameter values and calculating dose. DandD can evaluate dose for four different scenarios: residential, building occupancy, building renovation, or drinking water. Note that only two scenarios, residential and building occupancy, can currently be used for screening using the default parameter values in the software. The screening methodology and DandD are part of a larger decision framework that allows and encourages licensees to optimize decisions regarding alternative actions at their site, including collection of additional data and information [NRC, 1998]. This document provides instructions for the application and use of the DandD software.

. KEY WORDS/DESCRIPTORS *(List words or phrases that will assist researchers in locating the report.)*

decommission
decontamination
license termination
software
DandD
screening
dose assessment

13. AVAILABILITY STATEMENT

unlimited

14. SECURITY CLASSIFICATION

(This Page)
unclassified

(This Report)
unclassified

15. NUMBER OF PAGES

16. PRICE

RC FORM 335 (2-89)

UNITED STATES
NUCLEAR REGULATORY COMMISSION
WASHINGTON, DC 20555-0001

OFFICIAL BUSINESS
PENALTY FOR PRIVATE USE, $300

FIRST CLASS MAIL
POSTAGE AND FEES PAID
USNRC
PERMIT NO. G-67