LA-CP-03-0284

Los Alamos Controlled Publication;
distribution is limited

Title:

MCNP — A General Monte Carlo N-Particle Transport Code, Version 5

Volume III: Developer's Guide

Authors:

X-5 Monte Carlo Team

April 24, 2003

FOREWORD

This manual is a practical guide for the use of the general-purpose Monte Carlo code MCNP. The previous version of the manual (LA-13709-M, March 2000) has been corrected and updated to include the new features found in MCNP Version 5 (MCNP5). The manual has also been split into 3 volumes:

Volume I:	MCNP Overview and Theory	Chapters 1, 2 and Appendices G, H
Volume II:	MCNP User's Guide	Chapters 1, 3, 4, 5 and Appendices A, B, I, J, K
Volume III:	MCNP Developer's Guide	Appendices C, D, E, F

Volume I (LA-UR-03-1987) provides an overview of the capabilities of MCNP5 and a detailed discussion of the theoretical basis for the code. The first chapter provides introductory information about MCNP5. The second chapter describes the mathematics, data, physics, and Monte Carlo simulation techniques which form the basis for MCNP5. This discussion is not meant to be exhaustive — details of some techniques and of the Monte Carlo method itself are covered by references to the literature.

Volume II (LA-CP-03-0245) provides detailed specifications for MCNP5 input and options, numerous example problems, and a discussion of the output generated by MCNP5. The first chapter is a primer on basic MCNP5 use. The third chapter shows the user how to prepare input for the code. The fourth chapter contains several examples, and the fifth chapter explains the output. The appendices provide information on the available data libraries for MCNP, the format for several input/output files, and plotting the geometry, tallies, and cross-sections.

Volume III (LA-CP-03-0284) provides details on how to install MCNP on various computer systems, how to modify the code, the meaning of some of the code variables, and data layouts for certain arrays.

The Monte Carlo method for solving transport problems emerged from work done at Los Alamos during World War II. The method is generally attributed to Fermi, von Neumann, Ulam, Metropolis, and Richtmyer. MCNP, first released in 1977, is the successor to their work and has been under continuous development for the past 25 years. Neither the code nor the manual is static. The code is changed as needs arise, and the manual is changed to reflect the latest version of the code. This particular manual refers to Version 5.

MCNP5 and this manual are the product of the combined effort of many people in the Diagnostics Applications Group (X-5) in the Applied Physics Division (X Division) at the Los Alamos National Laboratory:

X-5 Monte Carlo Team

Thomas E. Booth	John T. Goorley	Avneet Sood
Forrest B. Brown	H. Grady Hughes	Jeremy E. Sweezy
Jeffrey S. Bull	Russell D. Mosteller	Richard F. Barrett (X-3)
Lawrence J. Cox	Richard E. Prael	Susan E. Post (CCN-8)
R. Arthur Forster	Elizabeth C. Selcow	Teresa L. Roberts (IM-8)

X-5 Data Team

Joann M. Campbell	Robert C. Little	Morgan C. White
Stephanie C. Frankle		

Technical Editors

Sheila M. Girard	Judith B. Shinn

The code and manual can be obtained from the Radiation Safety Information Computational Center (RSICC), P. O. Box 2008, Oak Ridge, TN, 37831-6362.

Forrest B. Brown
MCNP Team Leader
<fbrown@lanl.gov>

MCNP – A General Monte Carlo N-Particle Transport Code Version 5

X-5 Monte Carlo Team
Diagnostics Applications Group
Los Alamos National Laboratory

ABSTRACT

MCNP is a general-purpose Monte Carlo N–Particle code that can be used for neutron, photon, electron, or coupled neutron/photon/electron transport, including the capability to calculate eigenvalues for critical systems. The code treats an arbitrary three-dimensional configuration of materials in geometric cells bounded by first- and second-degree surfaces and fourth-degree elliptical tori.

Pointwise cross-section data are used. For neutrons, all reactions given in a particular cross-section evaluation (such as ENDF/B-VI) are accounted for. Thermal neutrons are described by both the free gas and $S(\alpha,\beta)$ models. For photons, the code accounts for incoherent and coherent scattering, the possibility of fluorescent emission after photoelectric absorption, absorption in pair production with local emission of annihilation radiation, and bremsstrahlung. A continuous-slowing-down model is used for electron transport that includes positrons, k x-rays, and bremsstrahlung, but does not include external or self-induced fields.

Important standard features that make MCNP very versatile and easy to use include a powerful general source, criticality source, and surface source; both geometry and output tally plotters; a rich collection of variance reduction techniques; a flexible tally structure; and an extensive collection of cross-section data.

Volume I: Overview and Theory

Volume II: User's Guide

Volume III: Developer's Guide

APPENDIX C - INSTALLING AND RUNNING MCNP ON VARIOUS SYSTEMS

Some of Appendix C is adapted from project development research notes.[1] The first half of the appendix (Sections I through VIII) addresses the mechanics of configuring, building, and installing the MCNP executable program(s) on both UNIX and PC platforms. The second half (Sections IX through XII) is adapted from a previous version of the MCNP manual and addresses the following topics: MCNP testing, modification, verification, and cross-section file conversion. Section XIII is a list of references.

I. A NEW BUILD SYSTEM FOR MCNP FORTRAN 90 ON UNIX

A new build system has been developed for use with the Fortran 90 version of the MCNP code. This build procedure is based upon the utilization of GNU *make*[2] and the burst file format (i.e., separate files for each subprogram) of the MCNP code.[3] Features of the new build system include a custom configuration utility which can be utilized in several modes to tailor the installation and execution of the code. The new build system is described in this appendix.

The build system previously used for MCNP was based upon a large "ID file" and a combination of scripts and Fortran 77 code.[4] The modernization of the code to Fortran 90, together with the use of burst files rather than large ID files, presents the need to upgrade and modernize the build system. GNU *make* is widely used on many different UNIX platforms to build and execute code.[5,6] GNU *make* may be installed on a system with the name *gmake* or *make*, depending on choices of the system administrator. In this appendix, use of the name *make* is assumed to refer to *gmake*. A version of GNU *make* can also be obtained for Personal Computers (PCs).[7]

Make is a tool for automating the compilation of large amounts of source code. Proper use of *make* reduces time spent compiling programs and guarantees that programs are compiled with appropriate options and linked with appropriate versions of modules and libraries. The *make* facility uses a special *Makefile* to define and describe targets, dependencies, abbreviations and macros, directory searches, and rules to automate the build process. For descriptions of the *make* facility, see References 2, 10, and 11.

With the help of the *make* facility, building MCNP for a variety of hardware platforms becomes easier for the end user. The end user simply types a *make* command, optionally specifying the desired target names and configuration features. As a prelude to issuing the *make* command, an installation script queries users about the relevant characteristics of their environment, then assigns values to special variables that are used in the special *Makefile* files that appear throughout the hierarchical levels of the source distribution.

Using *Makefiles* the new MCNP system can build much faster than the old system for any given platform using a single processor.[12,13] A detailed overview of the build system, with specific focus on the different modes of operation, and a discussion of the testing performed to date on different computer platforms with the MCNP Version 5 (MCNP5) code are included in this appendix.

The work described herein is specific to UNIX systems or UNIX-like environments, such as the Cygwin system for Windows PCs.

II. NEW UNIX BUILD SYSTEM DESCRIPTION

In the MCNP5 distribution, the user will find the two directories, *Source* and *Testing,* under the current working directory. The user should change directories to the *Source* directory, where all functions of the build system can be invoked. Table C.1 summarizes the directory structure and the files included in the MCNP5 distribution. The build system can be invoked simply by typing *./install*. This *install* utility will set up the configuration and invoke *gmake* to build and test the code. If the *install* utility described in the following pages does not meet the user's needs, the build can also be controlled directly with *gmake* as described in Section V.

The following files are provided with the MCNP5 distribution:

Table C.1
Description of MCNP Distribution Directories and Files

Directory	Relevant Files	Description
Source	install	Configuration setup utility, invokes gmake
	Makefile	Top level Makefile, invokes build and test
	answer.${sys}	Answer file generated by running *install* utility
	install.log	Install log file generated by running *install* utility
Source/config	Makefile	Makefile for generating custom configurations
	${OS}.gcf	Operating system specific GNU configuration file where ${OS} represents an operating system name, e.g., SunOS, AIX, IRIX64, OSF1, Linux
	${OS}-modes.gcf	Modes file for building and testing many configurations in series
	${OS}_aux.gcf	Rules for expected Fortran compiler and operating system combination
	Unix_options.gcf	Configuration options that are common to UNIX platforms
	VC_info.gcf	A generated file containing Version Control information from the configuration management repository

Table C.1
Description of MCNP Distribution Directories and Files

Directory	Relevant Files	Description
	custom_${OS}.gcf	User-specific custom configuration file generated by running the *install* utility with appropriate options given for a specific operating system
Source/src	Makefile	Makefile for building MCNP executable program
	*.F90	MCNP Fortran source code files
	mc.c	MCNP C source code file
	FILE.list	A list of all MCNP source code files
	Depends	A dependencies file
Source/datasrc	Makefile	Makefile for building MAKXSF executable program (cross-section file)
	makxsf.F90	MAKXSF Fortran source code file
Source/dotcomm/include	dotcomm.h	Include file for building libdotcomm.a
Source/dotcomm/src	Makefile	Makefile used to build libdotcomm.a
	*.F90	Fortran source for libdotcomm.a
Source/dotcomm/src/ internals/mpi	Makefile	Makefile used to compile C source for libdotcomm.a
	*.c, *.h	C source for libdotcomm.a interface to MPI libraries
Source/dotcomm/src/ internals/pvm	Makefile	Makefile used to compile C source for libdotcomm.a
	*.c, *.h	C source for libdotcomm.a interface to PVM libraries
Testing/Regression	Makefile	Makefile for building the regression tests for MCNP
Testing/Regression/Inputs	testinp.tar	Archive of test set input files in tar format (do not untar)
	testlib1	Type 1 cross-section (XS) for regression testing
	testdir1	XSDIR for regression testing
	specs.1-2	Specification file for type 1 to 2 XS generation
	specs.2-1a	Specification file for type 2 to 1 XS generation (not used by build system, present in Eolus Razor Repository)

Table C.1
Description of MCNP Distribution Directories and Files

Directory	Relevant Files	Description
Testing/Regression/ Templates	testoutp.${OS}	Operating system specific expected output for test problems
	testmctl.${OS}	Operating system specific expected tally output for test problems
Testing/config	test_options.mk	Make macros and options for testing

III. THE UNIX INSTALL UTILITY

The *install* utility, which is the top level component of the build system, can be utilized in the different modes described in Table C.2.

Table C.2
Ways of Invoking the Install Utility

Mode of Operation	System	Code	Version
install	Operating system is detected via uname command in user's current login session	MCNP	5 or 5.mpi or 5.pvm
install <code>	Operating system is detected via uname command in user's current login session	User input	5
install <code> <version>	Operating system is detected via uname command in user's current login session	User input	User input
install <sys>	User input (allowable values for <sys> include sgi, alpha, sun, aix, linux)	MCNP	5 or 5.mpi or 5.pvm
install <sys> <code>	User input (allowable values for <sys> include sgi, alpha, sun, aix, linux)	User input	5 or 5.mpi or 5.pvm
install <sys> <code> <version>	User input (allowable values for <sys> include sgi, alpha, sun, aix, linux)	User input	User input

The install script stores the options chosen by the user in the answers.${sys} file. When the install script is executed, the answers.${sys} file is read to restore the user's options.

Note that the order of arguments on the install command line IS IMPORTANT and must conform to the specifications in Table C.2

The last three options depicted in Table C.2 are useful for generating answer files and custom configuration files for UNIX systems that are different from the user's current login system. For example, a user working at an IRIX64 system might generate configuration files for Sun (OS-SunOS, sys=sun). In this case, the build system would generate the GNU configuration to include file and then exit, allowing the user to examine and edit the configuration files.

Please note that the answer files and custom configuration files perform the same function by setting up a custom configuration file for future reuse. The rationale for this redundancy is to accommodate a wide spectrum of users with varying preferences. This new build system will serve those individuals who prefer to install the code in the more traditional MCNP mode of menu options and answer files, as well as those who prefer to directly control the building and testing of the code through the widely used *gmake* utility.

Since this system is *gmake* centric, it is important to ensure that the user's environment variable for PATH is set correctly. The existence of a correct path can be confirmed by a positive response to typing "*which gmake.*" NOTE: *make* is aliased to *gmake* on some systems.

IV. UNIX CONFIGURATION WITH INSTALL UTILITY

Before starting construction it is wise to check the values of the PATH and DATAPATH environment variables with the UNIX echo command (`echo $PATH, echo $DATAPATH`). All compilers (which f90, which cc) you intend to use must be locatable with the $PATH value. The PATH environment variable must include the current working directory (.) to assure that all items referenced in scripts and Makefiles can be found. The UNIX `whoami` command must be locatable with the $PATH value. Any cross-section directories and libraries you intend to use must be locatable with the $DATAPATH value. Compilers and cross-section files and libraries are referenced in the config/$(OS).gcf, config/$(OS)-modes.gcf, and config/$(OS)_aux.gcf files. Rules associated with file name extensions for the specific compiler used on each operating system platform appear in the config/$(OS)_aux.gcf files.

Table C.3 shows the current key default configuration options for different systems that are included in the current *install* utility.

Table C.3
Default Configuration Options for UNIX Install Utility

Platform	OS	CONFIG=options	Datapath
sgi	IRIX64	seq plot	/usr/projects/data/nuclear/mc/type1
alpha	OSF1	seq plot	/usr/projects/data/nuclear/mc/type1
ibm	AIX	seq plot	/usr/local/udata/mcnpxs
sun	SunOS	seq plot cheap	/usr/local/udata/mcnpxs
intel	Linux	seq plot cheap	/usr/local/udata/mcnpxs
Notes: The HP system has not been configured or tested yet. Cray and Vax are no longer supported.			

In the figures that follow (C-1 through C-6), items tagged with a (toggle) label alternate between the values on/off or the alternate sequentially through a set of values that is listed near the (toggle) label. Items that are tagged with a (menu) label present a sub-menu of choices. The sub-menu works in a similar way to the top level setup menu, i.e., items are tagged. Items labeled (entry) require the user to type in the desired value or path. The examples given in the figures are the result of running the install script on a Sun platform.

Figure C-1. Example Top Level Setup Menu for UNIX Install Script

```
*************************** MCNP SETUP MENU ***************************
****** Type item number and <Enter> to toggle or change an item *******
********** Or type letter and <Enter> to execute the action ***********
***********************************************************************

COMPUTER SYSTEM DESCRIPTION                            sun, SunOS

Configurable Numbered Items                       Current Value
-----------------------------------------------------------------
1. 64-bit Consts & Vars (toggle)                     off
   32 bit (off) is recommended for Sun, Linux and Windows
   64-bit (on) is recommended for all others

2. Plotting (menu driven)                            on
   Graphics Option                                   XLIB
   Graphics Library Path                             /usr/openwin/lib
   Graphics Library Name                             libX11.a
   Graphics Include Path                             /usr/openwin/include

3. CROSS-SECTION DATAPATH (menu driven)              /usr/local/udata/mcnpxs

4. MULTIPROCESSING OPTIONS (menu driven)             seq

5. COMPILER OPTIONS (menu driven)
   FORTRAN 90 Compiler                               f90
   C Compiler                                        cc

6. Generate a Debuggable Version? (toggle)           no
```

```
7. Generate a Post-processed Version? (toggle)   no

8. Comparison with other source files? (menu)    no

9. Test Type 2 Cross Section Data? (toggle)      no

Permitted Letter Actions
------------------------
Z. Compile MCNP and run test problems
C. Only compile MCNP
T. Only run test problems
M. Only generate custom Makefiles
D. Run an automated script using default values and actions
X. Exit without doing anything

NOTE: 1 simultaneous gmake execution(s) will be allowed.
      Set the GNUJ environment variable to the desired number
      of parallel  executions before running this script.
```

Figure C-2. Example Plotting Sub-Menu (Item 2 from Top Level Menu)

```
**************************** GRAPHICS OPTIONS *************************

Configurable Numbered Items                    Current Value
------------------------------------------------------------
1. Plotting (toggle)                           on

2. Graphics Option (toggle)                    XLIB
                          XLIB
                          LAHEY
                          QWIN

3. Graphics Library Path (entry)               /usr/openwin/lib

4. Graphics Library Name (entry)               libX11.a

5. Graphics Include Path (entry)               /usr/openwin/include

Confirmed:  Graphic Library Path /usr/openwin/lib is valid.
Confirmed:  Graphics Library Name libX11.a is valid.
Confirmed:  Graphics Include Path /usr/openwin/include is valid.

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!Warning: Changes to the Graphics Library Name or Graphics Library Path!
!         here will not actually change the Name or Path.  You must    !
!         make the changes by hand to PLOTLIBS in config/SunOS.gcf!    !
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

Permitted Letter Actions
------------------------
X. Exit Graphics Options


***************** Type your choice and hit <Enter> ********************
***********************************************************************
```

Figure C-3. Example Cross-Section DATAPATH Sub-Menu (Item 3 from Top Level Menu)

```
**************************** Cross Section Data ***********************

Configurable Numbered Items                       Current Value
-----------------------------------------------------------
1. Cross Section DataPath (entry)           /usr/local/udata/mcnpxs

Permitted Letter Actions
------------------------
X. Exit Cross Section Data Options


       Confirmed: Cross Section Datapath verified.

***************** Type your choice and hit <Enter> ********************
***********************************************************************
```

Figure C-4. Example Multiprocessing Options Sub-Menu (Item 4 from Top Level Menu)

```
**************************** Multiprocessing OPTIONS *************************

Configurable Numbered Items                       Current Value
-----------------------------------------------------------
   MULTIPROCESSING OPTIONS                        seq

1. Distributed Memory Model (toggle)              seq
      Sequential                     seq
      Distributed Memory   MPI       mpi
      Distributed Memory   PVM       pvm

2. Shared Memory Model (toggle)                   none
      No Shared Memory               none
      Shared Memory        OpenMP omp


Permitted Letter Actions
------------------------
X. Exit Multiprocessing Options


***************** Type your choice and hit <Enter> ********************
***********************************************************************
```

Figure C-5. Example Compiler Options Sub-Menu (Item 5 from Top Level Menu)

```
**************************** COMPILER OPTIONS *************************

Configurable Numbered Items                       Current Value
-----------------------------------------------------------
2. Fortran 90 Path (entry)                        f90

3. C Path (entry)                                 cc

Permitted Letter Actions
------------------------
X. Exit Compiler Options

***************** Type your choice and hit <Enter> ********************
***********************************************************************
```

Figure C-6. Example Comparison Sub-Menu (Item 8 from Top Level Menu)

```
******************** Source Comparison Options ************************

Configurable Numbered Items                    Current Value
-----------------------------------------------------------
1. Comparison with other source files? (toggle)   no

Permitted Letter Actions
------------------------
X. Exit Source Comparison Options


***************** Type your choice and hit <Enter> ********************
***********************************************************************
```

V. *UNIX CONFIGURATION WITHOUT INSTALL UTILITY*

Before starting construction, it is wise to check the values of the PATH and DATAPATH environment variables with the UNIX `echo` command (`echo $PATH, echo $DATAPATH`). All compilers (which f90, which cc) you intend to use must be locatable with the $PATH value. The PATH environment variable must include the current working directory (.) to assure that all items referenced in scripts and Makefiles can be found. The UNIX `whoami` command must be locatable with the $PATH value. Any cross-section directories and libraries you intend to use must be locatable with the $DATAPATH value. Compilers and cross-section files and libraries are referenced in the config/$(OS).gcf, config/$(OS)-modes.gcf, and config/$(OS)_aux.gcf files (if they exist for the platform). Rules associated with file name extensions for the compiler used on each platform also appear in these files.

We recommend you work in the Source directory of the distribution. Verify the results of the build with the tests located in the Testing directory. The Testing directory contains cross-section files and libraries used with the set of tests in the same directory. Use of cross-sections other than the ones included in the Testing subdirectory (testdir1, testlib1) requires that the environment variable DATAPATH be set to an appropriate cross-section directory and the xsdir=value parameter must be used to specify the cross-section file that is appropriate for the $DATAPATH value. Content of input files must match the contents of the XSDIR that is specified.

Build the executable by issuing the following command:

```
    gmake build
```

This will build a sequential version of MCNP according to the settings found in the $(OS).gcf (see Table C.5) and the corresponding Makefile that includes it. See the commands documented in Table C.6 for more complex examples that specify the CONFIG=values parameter.

The Source and Testing directories should be at the same directory level. From within the Source directory, run the set of tests in the Testing/Regression directory by issuing the following command:

```
gmake test
```

The behavior of the test target depends upon whether or not the environment variable TESTDATA is set. If TESTDATA is not set, only type 1 cross-sections will be used with the tests. If TESTDATA is set, then the tests will be run first with type 1 cross-section data then with type 2 cross-section data.

If you wish to run the tests with only type 1 cross-section data, use the command

```
gmake test1
```

If you wish to run the tests with only type 2 cross-section data, use the command

```
gmake test2
```

Although this runs a variety of differently contrived tests, it does not test the plotting utilities. To run a few simple tests of the plotting utilities, work from the Testing/Regression directory. It is easy to access the cross-section files and directories from there. Create a symbolic link in the Testing/Regression directory that points to the mcnp executable (`ln -s ../../Source/src/mcnp5 mcnp`). Your path to the mcnp5 executable (the argument after the -s in the ln command) may vary, depending upon how you unpackage the source distribution.

Testing the 3 types of plotting will be done in 3 separate executions of mcnp. In these tests, the type 1 cross-section data is used.These tests assume that you have created the symbolic link, mcnp, as described above.

GEOMETRY PLOTTING [PLOT UTILITY]

Run the executable (mcnp) and specify the input file (i=inp01), the cross-section file (xsdir=testdir1), and options to process input and enable geometric plotting (ip) with the command shown below:

```
mcnp i=inp01 xsdir=testdir1 ip
```

Interact with the plot window that appears according to the instructions it displays.

To exit, click on the item in the plot window labeled *End.*

TALLY PLOTTING [MCPLOT UTILITY]

Run the executable (mcnp) and specify the input file (i=inp01), the output file (o=out01), the tally file (r=rtpe01), and the cross-section file (xsdir=testdir1) with the command shown below:

```
mcnp i=inp01 o=out01 r=rtpe01 xsdir=testdir1
```

Run the executable (mcnp) and specify the tally file where results of the prior run were collected (r=rtpe01), and enable tally plotting (z) with the command shown below:

```
mcnp r=rtpe01 z
```

When the mcplot> prompt shows up, specify the tally (tally 1) and a tally fluctuation chart (tfc=m) with the command shown below:

```
tally 1 tfc=m
```

To exit, give the exit command shown below:

```
e
```

CROSS-SECTION PLOTTING [MCPLOT UTILITY]

In the example that follows, a simple neutron problem is used and the default particle type in use is neutrons (par=n).

Run the executable (mcnp) and specify the input file (i=inp01), the cross section file (xsdir=testdir1), and the options to process input, process cross-section, and enable cross-section plotting (ixz) with the command shown below:

```
mcnp i=inp01 xsdir=testdir1 ixz
```

When the mcplot> prompt shows up, specify appropriate commands and values from the input file inp01 (e.g., material 2 (xs=m1) and reaction identifier 1 (mt=1)) with the command shown below:

```
xs=m1 mt=1
```

Try another plot at the mcplot> prompt (e.g., material 2 (xs=m2) and reaction identifier 2 (mt=2)) with the command shown below

```
xs=m2 mt=2
```

To exit, give the exit command as shown below:

```
e
```

VI. UNIX MODES OF OPERATION

The different modes of operation of the build system and the constituent Makefiles are described below and illustrated in Table C.6.

As mentioned earlier, the *install* utility defines and exports environment variables that are utilized in the Makefile system to build and test the code. In addition to invoking the execution of *gmake* from the *install* utility, it is also possible to control the execution of *gmake* from individual directories and Makefiles to perform specific functions. In particular, the *Source* Makefile controls the overall building and testing of the code, the *Source/src* Makefile controls the building of the MCNP executable, the *Source/datasrc* Makefile controls the building of the MAKXSF executable, the *Source/dotcomm/src* Makefile controls the building of the dotcomm library, and the *Testing/ Regression* Makefile controls the regression testing of the MCNP code.

A. Source Directory

This directory contains the top level Makefile, which, as stated, controls the main functions to build and test the code. These functions include the cleaning of all directories (removing files from previous builds and testing), conditional generation of a custom configuration file, conditional generation of a post-processed version of the MCNP code, building of the MCNP executable, building of the MAKXSF executable, and regression testing of the code. The conditional generation of a post-processed version of the code is set by the *install* utility.

A custom configuration file is only generated when the *install* utility is executed; otherwise, a custom configuration file, from a previous execution of *install,* is used to set the desired environment variables for a user-specific installation. The user is given the option to delete this custom configuration file when using the install utility. If a user desires to use the system defaults, then *gmake* may be invoked at this level without the presence of a custom configuration file. The defaults are included in the standard operating specific configuration file, ${OS}.gcf, which is also included as a Make include file. These defaults are the same as those invoked by the *install* utility, with the exception of the default configuration variable for the IBM AIX system.

In addition, there is a global export of environment variables from this top level Makefile to all the sub-makes that are subsequently called, but only when the *install* utility is executed. An option also exists to build the MCNP executable in parallel, and it is controlled through the environment variable GNUJ. If the user sets a value for GNUJ, then that value will be used to attempt to execute multiple make commands concurrently. If the user has not set a value for GNUJ, the default value of GNUJ is 1. Some installations are limited by single-user compiler licenses.

B. Source/config Directory

CUSTOM CONFIGURATION FILE

The configuration Makefile in this directory generates the custom configuration file based upon the variables that are exported from the *install* utility. As stated, this Makefile is only executed if the *install* utility is exercised; otherwise the custom configuration file is not created, but is included as a Make include file if present. The custom configuration file is called custom_${OS}.gcf, where ${OS} is the name of the operating system the user is logged into and gcf is a file suffix denoting "GNU configuration file". The contents of the custom configuration file are described below in Table C.4 and illustrated in Figure C-7.

Table C.4
Description of Variables Included in the UNIX Custom Configuration File

Variable	Significance
MCUSER	User designation, in UNIX **whoami**
OS	Operating system designation, in UNIX **uname** (e.g., IRIX64)
sys	System designation (e.g., sgi)
CODE	Code name (e.g., MCNP)
VERSION	Code version (e.g., 5.0)
menugra	Graphics option
menugpath	Graphics library path
menuglib	Graphics library name
menugincl	Include path
DATAPATH	Data path
NPVM	Number of PVM processes used for running the test suite
NTRD	Number of threads used for running the test suite
NMPI	Number of MPI processors used for running the test suite
menupre	Option to generate a post-processed version of the code
menucomp	Option for file comparison of post-processed code
compdir	Target directory used in file comparison option
TESTDATA	Option to generate and test type 2 data

Table C.4
Description of Variables Included in the UNIX Custom Configuration File

Variable	Significance
CONFIG	General configuration variable, includes the following options: [plotting multiprocessing cheap debugging]. Multiprocessing keywords are listed in Table C.6. For example, CONFIG=plot seq cheap debug. (On the AIX platform, IEEE is included as an additional option to specify various floating point options, and is used to compare with results from other systems.) (On the LINUX platform, portland, lahey, and absoft are included as additional options to specify the Fortran compiler to be used.) (On the Windows platform, compaq, lahey, and absoft are included as additional options to specify the Fortran compiler to be used. The options gcc, fcc, acc, and cl are included as additional options to specify the C compiler to be used.)
The variables CODE, VERSION, DATAPATH are then incorporated in zc_mod.F90 via C-preprocessor directives.	
premake	Logical switch to tell the Makefile that the install script or the custom_${OS}.gcf file is being used.
FC	Fortran 90 compiler
CC	C compiler
PVM_ROOT	Path to PVM distribution
PVM_ARCH	PVM machine type
MPI	MPI distribution (mpich or other)
MPIFC	MPI Fortran 90 compiler
MPICC	MPI C Compiler

STANDARD SYSTEM-DEPENDENT CONFIGURATION FILE

The standard system-dependent configuration file is called ${OS}.gcf, and contains system-dependent defaults, values of compiler and CPP flags, and specific files which are not used for certain options such as plotting. It is necessary to filter out unused files because some systems will give compiler warnings or errors if a file, after preprocessing, contains no Fortran statements. The key system defaults for the systems tested to date include:

Table C.5
UNIX System Defaults via ${OS}.gcf

System	Default Values
sgi	CONFIG=seq plot DATAPATH: /usr/projects/data/nuclear/mc/type1
alpha	CONFIG=seq plot DATAPATH: /usr/projects/data/nuclear/mc/type1
aix	CONFIG=seq plot DATAPATH: /usr/local/udata/mcnpxs
sun	CONFIG=seq plot cheap DATAPATH: /usr/local/udata/mcnpxs
linux	CONFIG=seq plot cheap DATAPATH: /usr/local/udata/mcnpxs

This standard system-dependent configuration file is included in the following Makefiles, along with the custom configuration files (if present): *Source*/Makefile, *Source/src*/Makefile, *Source/datasrc*/Makefile, and *Testing/Regression*/Makefile.

C. Source/src Directory

The Makefile in this directory generates the executable for the MCNP code. The name of the executable is defined by the CODE and VERSION variables previously referenced, for example MCNP5. This Makefile also invokes the generation of a post-processed version of the code that is conditionally set by the *install* utility, as mentioned in Section III. These intermediate files are traditionally given the .i suffix. These .i files are edited with a couple of simple lines of *Perl* to remove trailing white space and to delete the lines containing #ident. The purpose for this editing is to facilitate comparison with other similar source files located in another directory. The results of such a comparison are summarized in a listing of dif_.f files which is automatically generated when the post-processed file comparison option is enabled.

D. Source/datasrc Directory

The Makefile in this directory generates the MAKXSF executable. This executable translates type 1 cross-section data (ASCII text format) into type 2 cross-section data (binary format dependent upon the platform OS).

E. Source/dotcomm/src Directory

The Makefile in this directory governs the creation of the dotcomm library, which is only needed when the MPI or PVM versions of MCNP are built. The Makefile in this directory compiles all the .F files, also in this directory, and then passes control to the internals/mpi or internals/pvm directory Makefile, depending on the value of DOTCOMM_INTERNAL, which is set in the ${OS}.gcf config file. The appropriate .h files for MPI or PVM are also included, based on paths given in the ${OS}.gcf config files. After the .c source files in the appropriate subdirectory are built, control is passed back to the place where the *gmake* was invoked.

F. Testing/Regression Directory

The Makefile in this directory will execute the regression test suite and present a summary of the comparisons with standard *testoutp* and *testmctl* templates. The summary also includes the number of threads and/or processors used for multiprocessing runs. An additional feature, which is set by the *install* utility, is the option to generate and test type 2 data. In this mode, the regression testing of type 1 data is first performed, and then MAKXSF is executed with an appropriate SPECS file, and the regression testing is repeated with the type 2 data. The summary table indicates when type 2 data results are being shown.   This could be further generalized in the future to generate and test type 2 cross sections for data other than those included in the libraries developed for the testing regression suite. This is necessary due to an incompatibility of direct access binary cross-section files.

It is important to note that any option that is either a system default or included in the custom configuration file can be overridden on the *gmake* command line. For example, on the SGI IRIX system, the configuration default is ‘plot seq.’ If a user desired to modify this to run an MPI problem (build and test the code) with the number of processors equal to 3, the following options are available to implement this change in configuration. It is necessary to choose only one of these options to effect this desired change.

1. Rerunning the installation utility and modifying the response to the multiprocessing option selections
2. Modifying the custom configuration file, if present
3. Typing on the command line in the *Source* directory: gmake CONFIG=’plot omp’ NTRD=3

An additional method of operation involves the utilization of the file ${OS}-modes.gcf as shown in Table C.1 to define different targets and modes, as shown in Table C.3. This is useful for building and testing a spectrum of distributed and shared memory multiprocessing options, as well as incorporating combinations of other configuration variables such as cheap.

Table C.6
UNIX Commands

Directory/ Sub-Directory	**User Commands**	**Mode of Operation**	**Comments**
Source	`install`	General configure setup; creates answer files, cleans all directories, generates post-processed files (conditional), builds the code executable, builds the makxsf executable, and executes the regression test suite.	Create configuration setup through MCNP setup menu: Enable all *or* individual default options *or* enter user specific selections for setup.
Source	`install <sys>`	General configuration setup	Create configure files for UNIX systems *other* than current OS
Source	`gmake` *see note below on gmake	Cleans directories, generates post-processed files (conditional), builds the code, creates the makxsf executable, and runs the tests	
		• Using a custom configure file previously generated by *install* *or* • Without running *install* to generate custom configure files	• Uses information from custom configuration file *if present* *or* • Uses system-dependent default options if custom configuration file is *not* present
Source	`gmake reinstall`	Builds the executables, and runs the tests, omitting the initial file cleaning of all directories	Used for development

Table C.6
UNIX Commands

Directory/ Sub-Directory	User Commands	Mode of Operation	Comments
Source	`gmake clean`	Cleans	Cleans all directories
Source	`gmake preproc`	Generates post-processed files	Creates files
Source	`gmake build`	Builds the MCNP executable	Generates executable
Source	`gmake build CONFIG=mpi`	Builds the MPI version MCNP executable	**see note below on CONFIG
Source	`gmake build CONFIG=pvm`	Builds the PVM version MCNP executable	**see note below on CONFIG
Source	`gmake build CONFIG='mpi omp'`	Builds the MPI and threads version of MCNP	**see note below on CONFIG
Source	`gmake build CONFIG='pvm omp'`	Builds the PVM and threads version of MCNP	**see note below on CONFIG
Source	`gmake makxsf`	Builds the MAKXSF executable	Generates executable
Source	`gmake test`	Runs the tests	Executes the regression tests
Source	`gmake mode` *or* `gmake cheap-modes`	Builds the code and executes the regression test suite for many configurations consecutively	Uses the modes definitions, such as multiprocessing options and cheap, indicated in the ${OS}-modes.gcf files
Source/src	`gmake` *or* `gmake EXEC=<exec-name>`	Builds the MCNP executable *or* Changes the name of the executable from the default to <exec-name>	Generates object files, links, and creates the executable. If the object files and/or executable exist and are up to date, they are not regenerated
Source/src	`gmake clean`	Cleans the current directory	Removes files
Source/src	`gmake preproc`	Generates post-processed files	Creates files
Source/src	`gmake filename(s).o`	Generates all the specific object files requested	Useful if only one or a few files have changed

Table C.6
UNIX Commands

Directory/ Sub-Directory	User Commands	Mode of Operation	Comments
Source/datasrc	`gmake` *or* `gmake makxsf`	Generates the MAKXSF executable	Cross-section file handling
Source/ dotcomm/src	`gmake clean`	Cleans all dotcomm directories	Support for multi-processing options
Source/ dotcomm/src	`gmake libdotcomm.a DOTCOMM_INTERNAL=mpi`	Builds MPI version of libdotcomm.a	Support for multi-processing options
Source/ dotcomm/src	`gmake libdotcomm.a DOTCOMM_INTERNAL=pvm`	Builds PVM version of libdotcomm.a	Support for multi-processing option
Testing/ Regression	`gmake` *or* `gmake tests` *or* `gmake test1` *or* `gmake test2`	Cleans the current directory and executes the regression test suite	The number of threads or processors can be changed on the command line. This overrides the setting in the custom configuration file.
Testing/ Regression	`gmake ELOC=<path> EXEC=<exec-name>`	Changes the path and file name of the executable to the values given, cleans the current directory and runs the tests	
Testing/ Regression	`gmake clean`	Cleans the current directory only	
*Type `gmake | &tee file.log` to create a log file.			

Table C.6
UNIX Commands

Directory/ Sub-Directory	User Commands	Mode of Operation	Comments

**The CONFIG gmake command line parameter may contain several different values to control which version of MCNP is built. If more than one value is given, all should be contained in single quotation marks. Possible values include one from each of the following categories:
PARALLEL: seq, mpi, pvm, ‘mpi omp’, or ‘pvm, omp’
GRAPHICS: plot
DEBUG: debug
CHEAP: cheap
 	
Example 1:
To build a graphics threaded MPI version on an IRIX64 operating system, type the following command:
`gmake build CONFIG=’mpi omp plot’`
Example 2:
To build a graphics MPI version using the Portland Group PGI compiler on a LINUX operating system, type the following command:
`gmake build CONFIG='mpi plot cheap portland'`
 	
If CONFIG is not set on the command line, the default is used, which is sequential mode (seq).			

Figure C-7. Example of a UNIX Custom Configuration File with Default Options Set for a Sun System (Letter M from Top Level Menu)

```
# --- User identification ---
MCUSER=eolus
# --- Operating system identification ---
OS=SunOS
# --- System identification ---
sys=sun:
# --- Code name ---
CODE=mcnp
# --- Code version ---
VERSION=5
# --- Graphics option ---
menugra=XLIB
# --- Graphics library path ---
menugpath=/usr/openwin/lib
# --- Graphics library name ---
menuglib=libX11.a
# --- Include path ---
menugincl=/usr/openwin/include
# --- Datapath ---
DATAPATH=/usr/local/udata/mcnpxs
# --- Number of PVM processors ---
NPVM=1
# --- Number of threads ---
```

```
NTRD=1
# --- Number of MPI processors ---
NMPI=1
# --- Option to generate post-processed code ---
menupre=no
# --- Option for file comparison of post-processed code ---
menucomp=no
# --- Target directory used in file comparison option ---
compdir=/home/eolus
# --- Option to generate and test type 2 data ---
TESTDATA=
# --- General configuration options: Plotting Multiprocessing CHEAP Debugging
---
CONFIG=plot seq cheap
# --- Switch to say that this file has been read ---
premake=premake2
# --- Fortran 90 Compiler ---
FC=f90
# --- C Compiler ---
CC=cc
# --- MPI Implementation ---
MPI=other
# --- MPI Fortran 90 Compiler ---
MPIFC=f90
# --- MPI C Compiler ---
MPICC=cc
```

VII. INSTALLING AND BUILDING MCNP5 ON WINDOWS PCs

There are two different ways to install MCNP5 on a PC running a Windows operating system (95/98/NT/2000/XP/ME). The simplest method is to use the InstallShield® setup programs, similar to that of other Windows programs. The first setup program copies the MCNP executables, source code, and test problems to a user-selected directory and then sets two environmental variables. The MCNP Visual Editor and MCNP documentation are also installed. The second setup program installs the data libraries, MAKXSF (a cross-section library compression program), the files XSDIR and SPECS, and sets an environmental variable. The user is then asked to log out, log back in, and then run the test problems to verify that MCNP has been installed correctly. The main advantage of using this method is that no compilers are needed and no source code needs to be built. This option meets the needs of most users.

Alternatively the user can copy the MCNP directory tree to the desired location and use the supplied install script to build MCNP5, MAKXSF, and/or run the test problem suite. The install script can be used to build MCNP only if the appropriate compilers are already installed. The advantage to this method is that the executables can be rebuilt to apply patches or modifications to the source.

After installing or building the MCNP executable, additional software may need to be installed and appropriate environmental variables may need to be set or changed to take advantage of X11 graphics or parallel communications capabilities in MCNP. X-windows client software, not provided with MCNP5, needs to be running to display geometry, tally, or cross-section plots. X-Windows software is discussed on page C-24 of this appendix. To use the parallel versions of the MCNP executables, parallel communications software will need to be installed prior to running with this capability, and prior to building a parallel version of MCNP, if the install-shield executables are not used. Specific instructions on how to install MPI and PVM are given on page C-24. The environmental variables PATH and DATAPATH can be modified to make file management easier. DISPLAY may need to be set to use the plotting capabilities of MCNP. MCNP environmental variables are discussed on page C-25.

A. Installing MCNP5 on Windows PCs

THE INSTALLSHIELD® SETUP PROGRAM

The InstallShield® programs for MCNP5 are similar to that of other windows applications. Double clicking on the setup file will start the InstallShield® program. After starting the MCNP5 Executables installer, the initial setup window is displayed, then the next two windows present the Copyright notice and Software License Agreement, and request for user information. The following window asks where MCNP should be installed. It does not need to be in the default directory of /Program Files/LANL/MCNP5/. The installation package will then copy the plotting-sequential and parallel executables, MCNP source code, documentation (including the MCNP Manual), Visual Editor, and problem test suite into the chosen directory. The final screen queries the user if it can change the appropriate environmental variables: PATH and DISPLAY. Since PATH is already present, the MCNP5 directory path is appended to this variable. If these environmental variables are changed, the user must log out and then log back in (or reboot for some operating systems) before they will take effect. The user should be aware that if another executable with the name MCNP5 is already present in the path, the first executable in the path, i.e. the previously existing MCNP5, may be unintentionally used. The InstallShield® setup program cannot be used to build MCNP executables.

The second InstallShield® setup program installs the data libraries, the MAKXSF program and the XSDIR and SPECS files. The installer queries the user for a directory to place these files, which may or may not be a subdirectory where MCNP5 was installed. After this directory is specified, the environmental variable DATAPATH is set. Administrative privileges are also required, as well as write permission to at least 2.5 Gigabytes of hard disk space. This large amount of space is mostly used by the ASCII format (type 1) updated ENDF/B-VI cross-section libraries, which can be compressed with MAKXSF and the SPECS file to ~800 Megabytes. For more information about using MAKXSF, see section XII. beginning on page C-39 of this appendix.

After the InstallShield® programs are completed, the user should run the test suite to verify that the executable has been installed and operates correctly on the user's specific operating system and hardware. This testing procedure can be started by double clicking on the runprob.bat icon located in the directory Installation, where MCNP5 was installed. After the test problems are run, files that list the differences between the tally or output files generated and the expected results are displayed. These difference files should be reviewed by the user to determine if the differences are simple round-off errors or something more substantial, indicating incompatible hardware or software and that MCNP may give incorrect results.

To uninstall either the MCNP5 Executables or MCNP Data install packages, the user should remove them via the Windows Control Panel, with the Add/Remove Programs function. This will delete any files that were installed (files created while running the test problems will not be deleted) and will modify the registry appropriately. The environmental variables will not be removed, however, but these can be removed manually.

UNINSTALLING MCNP5

When MCNP5 is installed with the InstallShield® setup program, it can and should be uninstalled like any other Windows application. The user simply goes to the Add/Remove Programs feature inside the Control Panel within My Computer, selects MCNP5, and clicks on Change/Remove. Windows will then uninstall MCNP5, along with any files that it placed on the computer as part of the installation. Any files that were written subsequently, such as files that were written during the testing process, will not be removed, and folders that are not empty will not be removed. However, in such cases, the user has two simple alternatives. If the only additional files are ones that were written during the testing process, the user can run the cleanup.bat script to remove them by double-clicking on its icon in the Installation subfolder within the MCNP5 folder prior to uninstalling MCNP5. Otherwise, the user can manually remove the remaining folders and the files they contain.

It should be noted that uninstalling MCNP5 does not reset the environmental variables that were set as part of the installation. The user can manually remove or reset those variables, although in most cases they are ignored by other Windows applications.

If the user anticipates reinstalling MCNP5 at a later date, it is imperative that the uninstaller be used, because it cleans up the Windows Registry as part of the process. Simply deleting the folders created by the InstallShield® setup program leaves the Registry unaltered, which may cause problems during a subsequent reinstallation.

THE INSTALL SCRIPT

A second method to install MCNP5 uses an install script, which interactively queries the user for various build options and then executes the make utility to build MCNP, MAKSXF and/or run the test problem suite. The various build options include which Fortran and C compilers should be used, the location of appropriate X11 files, and the path to the XSDIR file. The script also gives the user the opportunity to only generate custom makefiles. These custom makefiles contain the build options selected, and will be automatically used whenever the make utility is used. An answer file, which lists the options chosen in the install script, is also created. The answer file can only be used to set options in the install script, and only if it is specified on the install command line. If MCNP is to be built, or the test suite is to be run, then install script will then execute *gmake* (the GNU version of *make*) and write most of the output to the file install.log.[5,6,7] This script, which uses the *gmake* utility to build the code, can only be used if a UNIX-based shell is installed.

INSTALLING A UNIX SHELL - CYGWIN

If the install script or *gmake* utility is used, a UNIX command shell must be installed. UNIX command shells are not standard on Windows operating systems and must be installed. Cygwin, a freeware port of a UNIX command shell for Windows PCs, can be obtained at www.cygwin.com or http://www.redhat.com/apps/download/. The setup program can be downloaded or run from the

Redhat website. This setup program will step users through the Cygwin installation process, allowing them to select web installation or download installation files to a local drive. The location where the Cygwin software should be installed (a path without spaces is recommended) and a temporary directory where files can be downloaded are specified in the next two windows.

After selecting a website to download or install from, the user selects Cygwin packages to install. In addition to the Cygwin packages that are selected by default, the *gcc* and *make* packages (located under the Devel directory) will also need to be selected to build MCNP with the *make* utility. The *perl* package (located under the Interpreters directory) is also recommended, and is required if the Absoft or Lahey Fortran compilers are going to be used to compile MCNP.

INSTALLING PLOTTING SOFTWARE - X WINDOWS CLIENT

In order to display MCNP plots, an X11 windows client software package is needed. Several commercial X-windows clients are available: Reflection X (http://www.wrq.com/products/), Hummingbird's Exceed_NT (http://www.hummingbird.com/products/nc/exceed/index.html), and Starnet's X-win32 (http://www.starnet.com/). No single commercial product is recommended. A freeware X client is also available with Cygwin, X-Free86. It has also been tested with MCNP5. These client software packages do not need to be the developer or professional versions, since the X11 header and library files are included with the MCNP5 distribution.

INSTALLING PARALLEL SOFTWARE - MPICH.NT OR PVM

In order to use or build MCNP5 with parallel capabilities, appropriate parallel communications software must be installed. Either MPI or PVM communications protocols are supported. To build a parallel version of MCNP5 for Windows, see page C-29.

The MPI port for Windows is MPICH.NT, developed at Argonne National Laboratory, and can be downloaded from http://www-unix.mcs.anl.gov/~ashton/mpich.nt/. This website also offers the helpful references MPICH.NT FAQ and MPICH Users Manual. If there is no need to rebuild parallel MCNP5, only the runtime dlls and MPIRun package will be needed (mpich.nt.1.2.4.exe). This package uses an InstallShield® setup program which requires installation from an administrative account on all PCs in the cluster. If MCNP5 needs to be built, then the source code (package mpich.nt.1.2.4.src.exe) should be downloaded and unzipped as well. The program MPIConfig must be run on each computer after MPICH.NT installation. The local host name must be added and the settings applied.

Once MPICH.NT is installed, a few additional steps are required. MPI enabled MCNP must be copied to the same directory on all hosts. MCNP can be executed through either the Windows MPIRun GUI or command line MPIRun. For the MPIRun GUI, hosts must be added by selecting or typing their names in the hosts section. The DATAPATH may need to be set under the advanced options. For the command line MPIRun, the hosts must be specified by the `-hosts` option, which can be used to specify the number of processes started on each host. Typically the number of processes is equal to the number of CPUs utilized plus one. The first process listed is the master process and does not run any histories. For example, the command to start three MCNP MPI processes on ComputerA (a dual CPU machine) and one process on ComputerB is:

```
mpirun -hosts 2 ComputerA 3 ComputerB 1 mcnp5mpi inp=test
```

Alternatively, MCNP5 can be built and run with PVM, developed at Oak Ridge National Laboratory. The PVM port for Windows can be downloaded from http://www.csm.ornl.gov/~sscott/PVM/Software/. The file ParallelVistualMachine3.4.3.zip contains the source and binaries needed to install and run PVM on a single computer. This InstallShield® program must be run on all Windows PCs in the cluster. PVM requires additional communications software, a remote shell (RSH) client/server package, before it will run across a cluster of Windows PCs. Two commercial RSH packages can be obtained from http://www.winrshd.com/ and http://www.ataman.com/. The RSH package must be installed on each computer in the cluster, and the permissions must be set to allow RSH or REXEC connections for the desired user accounts. The Ataman RSH package was successfully tested.

Similar post-installation steps are required to run the PVM version of MCNP5 on Windows PCs. The MCNP executable must be copied into the %PVM_ROOT%/pvm3/bin/WIN32 directory on all hosts. PVM must be started on a single computer before a PVM enabled MCNP can be executed. After PVM is started, additional hosts can be added to the PVM cluster with the "add host" command from the PVM console prompt. PVM operability can be tested and verified with the PVM example programs, such as hello. MCNP can then be started from a separate command shell with the following command:

```
mcnp5pvm inp=test tasks n
```

where `n` is the number of slave processes. The number of tasks is usually the number of CPUs in the cluster. A negative number entered for `n` causes MCNP to skip the initial load-balancing feature, and is recommended for a homogeneous cluster. Additional information on how to install MPI or PVM can be obtained in their respective user manuals. Additional information on running MCNP in parallel can be found in Appendix C section VIII. beginning on page C-30. For more information about running MCNP in parallel on Windows clusters, see Reference 16 on page C-42.

SETTING ENVIRONMENTAL VARIABLES

After installation, it may be necessary to change or add three Windows environmental variables. Variables are set differently for Windows 95/NT/2000/XP/ME, but for each of these operating systems the variables can be viewed the same way. The value of an environmental variable PATH, for example, can be printed in a command shell window (i.e. a "DOS prompt") with the command

```
echo %PATH%.
```

The first environmental variable that may need to be changed is PATH, a semicolon-separated list of directories used to find executables and dynamic link libraries. The directory where MCNP is installed should be included, so that MCNP can be executed from any directory, making file management more convenient. If several programs with the same name exist within PATH, then the executable in the first occurring directory will be executed. Appending the directory of the newly installed version of MCNP to the PATH may not change which MCNP is executed if an older version with the same name is given earlier in the directory listing. Failure to change PATH will mean that the MCNP executable must be located in the directory where the input file is located. To use the plotting features of MCNP, the location of the dynamic link library X11.dll or Xlib.dll may need to be added to PATH as well. Failure to make sure that X11.dll or Xlib.dll is in your path may mean that a plotting-enabled executable cannot run, even if the plotting features of MCNP are not used.

The second environmental variable that may need to be added or changed is DATAPATH. It is the directory path to the file XSDIR, which is used by MCNP to locate the cross-section data libraries. MCNP also searches in the local directory and a directory specified at compile time. If XSDIR does not exist in any of these three locations, MCNP will issue a FATAL error. Additionally, the command line option XSDIR=name may be used to specify an XSDIR formatted file with a different name located in the directory given in DATAPATH. Failure to set DATAPATH means that the file XSDIR must be located in the directory where the input deck is located.

The third environmental variable is DISPLAY, which is only needed for a plotting version of MCNP. This environmental variable is used by the X windows client to route X windows. It is usually set to display windows on the same computer where MCNP is executed. In this case, the value of DISPLAY should be localhost:0.0. Failure to set DISPLAY may mean that the plotting executable will not be able to open a window and plot. A graphics version of MCNP will still be able to create a postscript file of the plots, even without the X-windows client software or the environmental variable DISPLAY.

B. Building MCNP on Windows PCs

MCNP5 must be built if the install script is used or the source needs to be modified or patched. The install script uses the *make* utility to direct one of three supported Fortran 90/95 compilers, and one of three supported C compilers, to compile and link MCNP. Alternatively, the interactive graphical interface, Compaq Developer Studio, uses Compaq Fortran 90 and Microsoft C to build MCNP. Neither the *make* utility nor a UNIX shell is needed to build MCNP with Compaq Developer Studio. These two methods will be discussed in the following paragraphs.

THE *MAKE* UTILITY

Make is a utility which understands user defined relationships between different files used to build programs. In an attempt to supervise the building of a target program, *make* controls preprocessors, compilers, linkers, archive utilities, or other programs to manipulate these files. MCNP5 can be built on all supported platforms, including Windows PCs, with the *make* utility.

Unlike most UNIX or LINUX based operating systems, the *make* utility is not standard on Windows. The recommended version of *make* for a Windows PC is the GNU *make* utility, which is an optional addition with Cygwin, described above. The *make* utility does not include any compilers, which also must be installed to build MCNP. The *make* utility can be used to build MCNP by typing "make build" in a Cygwin command prompt in the MCNP5/Source directory.

One of the first things *make* does is read an operating-system dependent file in the MCNP5/Source/config directory. For all Windows installations, this file is Windows_NT.gcf. It contains all the default paths to the supported compilers, header files, and libraries, and specifies the compiler and linker options. This file should be modified if the compilers and other necessary files are not in the default locations. The *make* utility will also read the custom_Windows_NT.gcf file created by the install script if present. Additional information about these files and *make* commands is given in Appendix C section II. beginning on page C-2.

THREE FORTRAN COMPILERS

The *make* utility uses one of three Fortran 90/95 compilers and one of three C compilers to build MCNP5 on Windows PCs. The supported Fortran 90 compilers are: Compaq Visual Fortran (CVF version 6.6B) [formerly known as Digital Visual Fortran], Lahey Fortran 95 Pro (LF95 version 5.70c), and Absoft Pro Fortran (F95 version 8.0).[8,9] The supported C compilers are the Microsoft C/C++ compiler (MSC version 12.00.8168) and GNU gcc compiler (version 2.95.2-5 [Cygwin special]), either of which may be used with any of the three Fortran compilers. The Fujitsu C compiler (FCC version 3.0) is also supported, but only in conjunction with Lahey Fortran 95. Table C.7 shows the versions of MCNP that can be compiled with the *make* utility. For example, the command "make build CONFIG='plot cheap compaq cl' " at the Cygwin command prompt in the MCNP5/Source directory will build a plotting version of MCNP5 with the Compaq Fortran and Microsoft C compilers.

Table C.7
Supported Versions and Compilers with the Cygwin *Make* Utility

MCNP Version	Supported Compiler Sets*	Make Command Line	Notes

Sequential	Compaq (v6.6B)
	Lahey (v5.70c)
	Absoft (v8.0)
		CONFIG='seq compaq'
		CONFIG='seq lahey'
		CONFIG='seq absoft'
			GNU gcc is used***
			Absoft and Lahey need preprocessor.
			Absoft does not support control-c interrupts.

Plotting	Compaq + MSC**
	Compaq + gcc
	Lahey + MSC
	Lahey + gcc
	Lahey + fcc
	Absoft + MSC
	Absoft + bcc
		CONFIG='plot compaq cl'
		CONFIG='plot compaq gcc'
		CONFIG='plot lahey cl'
		CONFIG='plot lahey gcc'
		CONFIG='plot lahey fcc'
		CONFIG='plot absoft cl'
		CONFIG='plot absoft gcc'
			X11 library and headers required to compile.
			X client running required to display.
			Absoft and Lahey need preprocessor.
			Absoft does not support control-c interrupts.

*Only the professional version of the three Fortran compilers is supported.
**The default. The command "make build" will build a plotting executable with CVF and MSC.
***The *make* utility expects there to be an object file from the c source, so gcc is used to build a nearly empty object file which the Fortran compilers link.

All of the compilers use the environmental variables LIB and INCLUDE to locate the compiler's libraries and include files, respectively. These should be set when the compilers were installed. Additionally, the directory path to the compiler executables (f90, lf95, f95) should also be located in the PATH environmental variable. If more than one compiler with the same name is installed, the explicit path and compiler can be specified in the Windows_NT.gcf file.

A PERL PREPROCESSOR

Unlike the Compaq Fortran compilers, the Absoft and Lahey Fortran compilers do not have the capability to preprocess #ifdef statements that are used in the MCNP Fortran and C source to specify version (sequential, plotting, etc) and platform (UNIX, LINUX, DEC, etc.) specific code. The MCNP Fortran source must be preprocessed before the Absoft and Lahey compilers can be used. This can be done with the *perl* script fpp.pl, which is provided with the MCNP distribution. This script processes out the #ifdef statements and substitutes the appropriate values for the MCNP version, compile date, and version number. If the environmental variable DATAPATH is set, it will also be substituted directly into the source code before MCNP is compiled. The processed files are saved with .F95 extensions for Absoft or .i extensions for Lahey, which are then compiled. The intermediate files are deleted after linking.

COMPAQ DEVELOPER STUDIO

As an alternative to the *make* utility, the Compaq Developer Studio build utility is also supported. It is an independent GUI which requires neither Cygwin nor the *make* utility. Using the Developer Studio, it is possible to build a sequential, plotting, MPI or PVM version of MCNP5. Included in the MCNP5 distribution are Developer Studio projects and project workspaces (.dsp and .dsw files, respectively) for each of these versions. To build one of these four versions, open the corresponding .dsw file located in the /Source/CVF directory. If CVF is installed on the computer, this can be done simply by double clicking on the appropriate file in the Windows Explorer.

The only settings that may need to be changed are the paths to the X11, MPICH.NT or PVM files, if they were not installed in the default directory. The paths to these files are specified in the C preprocessor and link input fields of the project settings. Unlike *make*, the CVF Developer Studio has no problems when paths to these files contain spaces. An additional path to the file XSDIR can also be specified at compile time. The Fortran preprocessor keyword DPATH and its value can be set on the project settings' Fortran tab. While the directory path must not contain spaces, the DOS style format will work. For example, instead of C:\Program Files\LANL\MCNP5\data, C:\Progra~1\LANL\MCNP5\data should be specified.

The option to build is the second item under the build menu on the topmost menu bar. Alternatively, the F7 button can be pressed to start the build. In some cases Developer Studio cannot determine which files need to be built prior to others, and the code may need to be rebuilt after the first failed build completes. Additionally, the user is encouraged to build the release version of MCNP by setting the active configuration (under the build menu item). The versions of MCNP5 that have been built and tested with the Compaq Developer Studio are shown in Table C.8.

Table C.8
Supported Versions with Compaq Developer Studio

MCNP Version	Supported Compiler Sets	Notes
Sequential	Compaq	No C compiler is needed.
Plotting	Compaq + MSC	X11 library and headers required to compile. X client required to display.
Parallel (MPI or PVM)	Compaq + MSC	PVM or MPICH.NT also required.

BUILDING PLOTTING VERSIONS

To display MCNP geometry, tally, or cross-section plots, it is necessary to have a plotting enabled version of MCNP and X window client software. A plotting version of MCNP can be built with either the *make* utility or Developer Studio. With *make*, a plotting executable is the default. To force *make* to build a plotting enabled version of MCNP with *make*, it is necessary to specify the plot on the CONFIG keyword.

One of the enhancements for Windows PCs is that all of the appropriate source files (including the X11.lib, X11.dll, and header files) are included in the MCNP distribution and are located in the Source/X11r6 directory. The original source can be downloaded from www.X.org. The X11 library may be downloaded from ftp://ftp.cc.utexas.edu/microlib/nt/x11r6/ or built from the X.org source. Proprietary X11 header files and library files may also be included with a commercial X windows client (if the developer/professional version was purchased), but using the X11 files with the MCNP distribution is recommended.

BUILDING PARALLEL VERSIONS

To use the parallel features of MCNP, it is necessary to have a parallel enabled version of MCNP and the appropriate parallel communications software installed and running. Parallel enabled versions of MCNP can only be built with Compaq Developer Studio after the header files and appropriate libraries have been installed.

The Developer Studio project files mcnp5mpi.dsw and mcnp5pvm.dsw already have all the appropriate settings configured. These changes include the addition of the preprocessor definition MULTP (and MPI for the MPI version), additional Fortran and C source, MCNP include files, and the additional paths to these directories and the MPI or PVM directories. The only changes needed are the paths to the MPI or PVM include files and libraries if these are not installed in the default directories. The include file path will need to be listed in the C++ tab (preprocessor category). The MPI or PVM library should be listed in the Object/library modules line on the link tab (the general category). The necessary libraries for MPI are the ws2_32.lib and mpich.lib libraries. The PVM libraries are ws2_32.lib, libpvm3.lib, and libgpvm3.lib. The path to this library should be listed in the input category.

VIII. PARALLEL CONFIGURATION INFORMATION

BUILDING THE DOTCOMM LIBRARY

The dotcomm library, libdotcomm.a, is only needed when the MPI or PVM version of MCNP is built. The dotcomm.a library will be automatically built if the *gmake* command-line parameter definition of CONFIG contains either MPI or PVM, or the MPI or PVM option is selected in the install script. To build the dotcomm library only, the command `gmake libdotcomm.a DOTCOMM_INTERNAL=x` should be typed in the Source/dotcomm/src directory, where x is either MPI or PVM. If either parallel option was selected, the dotcomm library and either the MPI or PVM library is linked when the MCNP executable is linked. If the MPI or PVM libraries are not found while linking, then the appropriate library include path should be set in the ${OS}.gcf file by adding the path to the definition of DMMP. While the dotcomm library cannot be built with both MPI and PVM, either option can be combined with the OpenMP threads (CONFIG=omp) option, which does not use the dotcomm library.

EXECUTION IN PARALLEL MODES

MCNP may be built for either MPI or PVM and then executed in the same fashion as other parallel programs on your system. In general, for MPI the execution line will look like:

```
mpirun -np <m> mcnp5.mpi i=inp01
```

where <m> is the total number of processes including master, and m-1 slave processes will be available to transport particles. To provide load balancing with MPI, add the keyword "balance" to the command line. For load balancing with MPI, the execution line will look like:

```
mpirun -np <m> mcnp5.mpi i=inp01 balance
```

If the MPICH implementation of MPI is used, then the keyword "eol" must be added after all other MCNP keywords to distinguish MCNP keywords from directives added by MPICH. In this case your execution line will be:

```
mpirun -np <m> mcnp5.mpi i=inp01 eol
```

If you combined the build with OpenMP for a multiprocessor per node SMP machine, then each slave may be optionally threaded by setting the tasks option:

```
mpirun -np <m>   mcnp5.mpi   i=inp01 tasks <n>
```

The syntax required to allocate enough resources for the threading varies by system. For instance, on an alpha Tru64 System 5.1, use:

```
prun -n <m>  -c <n> mcnp5.mpi    i=inp01 tasks <n>
```

to reserve n processors for each <m> MPI process.

The combined-option executable can be executed in any of the following ways: sequential, all-MPI, all threads, or hybrid. Note that the minimum number of MPI slaves accepted by MCNP is 2, so you need at least three MPI processes.

For a PVM execution, you first need to assure there is a copy of (or link to) the MCNP5.pvm executable in subdirectory pvm3/bin/<pvm arch> of the home directory. Launch PVM. At the `pvm>` prompt, add machines to the "Parallel Virtual Machine" using the `'add'` command. The syntax of the `'add'` command is `'add <hostname>'`, where `<hostname>` is the name of the machine to be added to the "Parallel Virtual Machine." Next, give the command `'quit'` to exit the console, but leave the pvm daemon running.

Execute the MCNP job(s), and then at the end of the session again enter PVM to return to the console. Give command `'halt'` to kill the pvm daemon.

The MCNP command line will be of the form:

```
mcnp5 i=inp01 tasks <j>
```

where the absolute value of <j> is the number of slaves that will be spawned to track particles. If the user built with PVM OpenMP combined, the command line will be

```
mcnp5  i=inp01 tasks <j>x<n>
```

where each of the j spawned slaves runs particles on n threads.

For either PVM option, if j is positive, a load balancing operation is enabled early in the run to account for a heterogeneous environment. To omit that step, as for a homogeneous cluster, enter a negative value for j.

The simplest parallel run would be to build for one shared memory node and to just run openMP threading (No PVM or MPI). Build with CONFIG='omp' and execute with just tasks option on the following command line:

```
mcnp5  i=inp01  tasks <n>
```

Running the automated test suite in any of these parallel configurations may require editing the Testing/Regression/Inputs/ runprob.mpi to satisfy your environment. Script runprob.mpi is used for MPI testing; runprobmt for all other configurations (seq, OpenMP only, PVM).

IX. TESTING PERFORMED TO DATE

The testing performed to date includes the following platforms: sgi, alpha, aix, sun, and PC. There were several limitations and problems encountered, which are mentioned in this section.

On some platforms, such as the Alpha OSF1 Q machine, it is necessary to load modules upon login in order to access compilers or debugging tools. The compiler versions corresponding to the tested platforms include:

Table C.9
UNIX Platforms Tested to Date

Platform Name	Compiler Version	Mode	Comments
SGI IRIX64	MIPSpro 7.3.0 MIPSpro 7.3.1.2m	Sequential Seq and Parallel	host kaji hosts theta and bluemountain
Alpha OSF1	Fortran 5.3-915 Fortran 5.4.1.a	Sequential Seq and Parallel	host ratbert hosts QSC and Q
AIX 4.3.3 AIX 4.3 AIX 5.1	xlf90 7.1 xlf90 7.1 xlf90 7.1	Sequential Seq and Parallel Seq and Parallel	host toji host blue at LLNL host frost at LLNL
SunOS	f90 version 6.2 f90 version 6.1	Sequential Seq/PVM	host glitter host glitter
Linux	PGF90 4.0-1	Seq/MPI/PVM/ OMP	host lambda
Linux	LAHEYPro_6.1e	Seq/MPI/PVM/ OMP	host lambda
Linux	Absoft Pro-8.0 QF3	Seq/MPI/PVM	host lambda

In Table C.10, Table C.11, and Table C.12, the software and settings used to test the PC version of MCNP5 are documented.

Table C.10
PC Platforms Tested to Date

Software	Version	Used in MCNP Mode	Notes
Compaq Visual Fortran	6.6B	seq, plot, mpi, pvm	
Compaq Developer Studio		seq, plot, mpi, pvm	
Microsoft Visual C/C++	6.0	plot, mpi, pvm	
Lahey Pro Fortran 95	5.70c	seq, plot	
Fujitsu C/C++	3.0	bplot	Only links with Lahey
Absoft Pro Fortran 95	7.5, 8.0	seq, plot	
GNU gcc	2.95.3-5 (cygwin special)	seq, plot	
Cygwin	Setup version 2.249.2.5	seq, plot	
GNU Make	3.79.1 (i686-pc-cygwin)	seq, plot	
GNU Perl	5.6.1 (cygwin multi)	seq, plot	Only required with build using Absoft or Lahey
fpp (Perl Script)		seq, plot	Only required with build using Absoft or Lahey

Table C.11
Files, Libraries, and Client Software for Plotting on PCs

Software	Version	Used in MCNP Mode	Notes
X11 .h include files	X11R6.6	plot	
X11.lib	X11R6.6	plot	
Reflection X	9.0.3	plot	X windows client

Table C.12
Software for Parallel Jobs on PCs

Software	Version	Used in MCNP Mode	Notes
PVM	3.4.3	PVM	
ATRLS (*TCP Remote Logon Services*)	3.1	PVM	Not needed for single multiprocessor PC. Needed for cluster of PCs. Only tested with Windows 2000 PCs.
MPICH_NT	1.2.4	MPI	

X. MODIFYING MCNP WITH PATCHES

This section describes the process for modifying MCNP. The method is based on the GNU tools *diff* and *patch*.[5]

NOTE: *Before making any changes to the MCNP distribution, backup the original files for recovery and for comparison. It is necessary to have an unmodified copy of the entire distribution to generate and/or apply patches.*

The method for creating and applying patches has changed completely from how it was done with earlier versions of MCNP. Patches are not written; they are generated. Also patches can be applied to the entire MCNP directory tree.

The commands described here to build or rebuild the code assume a UNIX operating system or a similar shell running on Windows (Cygwin).

Why Use Patches?

Why might you need to use a patch?

- A patch may be issued by the MCNP team to fix bugs or add new features.
- You may want to distribute your changes to others.
- You may want to adapt your changes to a new MCNP version.

In each of the above scenarios, the ability to summarize the changes in a short file makes the work easier. For example, if the MCNP team issues a patch and you have also made local changes, you can use two patches to merge the two together.

Modifying the Source Code

To make local changes to a copy of MCNP, just edit the appropriate source files. The directory structure for MCNP5 is:

```
Source/
     Makefile
     install
     config/
     CVF/
     datasrc/
     dotcomm/
          src/
     X11R6/
Testing/
     Regression/
          Makefile
          Inputs/
          Templates/
     config/
          test_options.mk
```

The radiation transport source code is in *Source/src/.* If you need to make changes to existing files, edit them and leave them in their current locations.

If you add new files, they will be automatically included in builds if you follow these steps.

1. Create the new file conforming to the Fortran ANSI Standard[14] in the free-style format. The name of new Fortran files should be of the form `<name>.F90`. See Reference 15 for the definition of the MCNP coding style guide.
2. Edit the file *Source/src/FILE.list* to add your new Fortran files to the *F_SRC* macro. A backslash is necessary to continue to the next line. So if you add a whole new line at the end, you must append a backslash on the line before it. If you insert a line in the middle, put a backslash at the end of the new line.
   Note: *There can be no white space after the backslash.*
3. Add a dependency line for each of your new files. If your files do not USE any MCNP modules (existing or new), this step can be skipped. If you make modifications to an existing file that adds a module dependency, add the new dependency condition to the appropriate line. See the *Source/src/Depends* file for the format.
4. In *Source/,* type `'make clean'` to prepare for a full rebuild.

When your changes are ready, change into the *Source/* directory and type `'make build'`.

GENERATING A PATCH

To generate a patch from a modified version of MCNP, you need to have access to an unmodified version (or other reference version) for comparison. Comparisons and patches are generated with the GNU utility *diff*.

To see the extent of the modifications, you can generate a list of modified, added and deleted files with the command

```
> diff -qr <base_dir> <modified_dir>
```

Be sure to examine the output of this command to look for unintended differences. Object files, editor-backup files or executables may be in the list. Clean up any extraneous files before proceeding with patch generation.

For example, comparison of two MCNP directories *MCNP/Source/* and *MCNP/Modified/* will generate output like the following. In this example, there are no added/deleted files or extraneous files, only modified ones.

```
> cd MCNP
> diff -qr Source Modified
Files Source/src/Makefile and Modified/src/Makefile differ
Files Source/src/dmmp.F90 and Modified/src/dmmp.F90 differ
Files Source/src/dynamic_arrays.F90 and Modified/src/dynamic_arrays.F90 differ
.
.
.
```

The patch itself is generated with the command

```
>diff -Naur <base_dir> <modified_dir> > <patch_file>
```

where <patch_file> is the name you want to give the file containing the patch.

To generate the patch, you should position the two directory trees adjacent to each other. In the above example, the unmodified MCNP distribution, `<base_dir>`, is in *MCNP/Source*, the modified version, `<modified_dir>` is in *MCNP/Modified.* Let the desired output file, `<patch_file>` be *MCNP/my_mods.txt.*

For this example, the commands to generate the patch are

```
> cd MCNP
> diff -Naur Source Modified > my_mods.txt
```

The first few lines of `my_mods.txt` would look something like the following patch if the file *Source/src/Makefile* was modified.

```
diff -Naur Source/src/Makefile Modified/src/Makefile
--- Source/src/Makefile 2002-10-17 12:32:40.000000000-0600
+++ Modified/src/Makefile2002-10-21 21:07:40.000000000-0600
.
.
.
```

After generating a patch, you can add commentary to the patch-file to describe its function. Start all comment lines with `'#'` to mark them as comments. See the GNU documentation on *patch* (type `'man patch'` on UNIX systems) for more information on the flexibility of this excellent tool.

APPLYING A PATCH

Applying a patch is as simple as generating one. It is especially easy if it is a patch to the base MCNP distribution and you want to apply it to your unmodified copy of the same version of MCNP.

Note: *In all cases, make backups of the patches, the original distribution, and your modified directory before proceeding.*

To see the files that a particular patch will affect, you can use the GNU utility *grep* as follows. This can be useful to determine if two patches are independent or may be in conflict (modifying the same files). For the example used above, partial results are given below.

```
>grep '^diff' <patch_file>
diff -Naur Source/src/Makefile Modified/src/Makefile
diff -Naur Source/src/dmmp.F90 Modified/src/dmmp.F90
diff -Naur Source/src/dynamic_arrays.F90 Modified/src/dynamic_arrays.F90
.
.
.
```

If an unmodified copy of the MCNP distribution (*not your only copy!*) is stored in *MCNP/Source*, the commands to apply a patch named *mcnp_patch.20021102* is

```
>cd MCNP
>patch -p1 -d Source < mcnp_patch.20021102
```

The output will look something like the following with one line per patched file:

```
patching file `src/Makefile'
.
.
.
```

If all the output is of this form, the application of the patch is fully successful.

Complications can arise if you need to apply more than one patch to the same version. Patches from the MCNP team will be documented for compatibility with other such patches. If a specific order of application is required, that will be made clear.

If you have made local modifications to the base distribution, follow these steps:

1. Generate a patch with your changes from the base MCNP distribution or a version patched only with MCNP-team patches. See the discussion above for instructions. Back up a copy of your patch.
2. Apply any new patches from the MCNP team in the documented order before applying your patch. You should also back up the MCNP-team patches for future use.
3. Back up the newly-patched MCNP distribution for future reference.
4. Apply your patch to the newly-patched MCNP.

If your patch modifies sections of MCNP that are also modified by other patches, portions of your patch may be rejected by the *patch* utility as being in conflict with the newly-patched version. In these cases, a file will be created with the rejected portions. For example, if the patch to *Source/src/Makefile* fails, a file named *Source/src/Makefile.rej* will be created with the problem section. You will have to resolve any such rejections manually. In this case, a file named *Source/src/Makefile.orig* will also be created and represents a copy of the original.

If you receive an MCNP-team patch, you *can* try to apply it directly to a locally modified version. The format of the generated patches makes the patch utility able to find the most likely place to insert modifications. If the patching fails, fall back to the procedure given above.

Note: It is possible that either method of applying your own patches on top of MCNP-team patches will succeed, but still generate incorrect code. *The MCNP team cannot and does not give any warranty that your modifications will be compatible with its own. It is the responsibility of the author of the non-MCNP-team modifications to make them compatible with team releases.*

XI. MCNP VERIFICATION

MCNP5 comes with a set of test problems that appear in a directory called Testing. This directory should be placed at the same level as the Source directory. In other words, the parent directory of both Testing and Source should be the same. This allows the install and test script to find the scripts that run the set of test problems. Input files and Templates of expected test results for each supported platform are included in the content of the Testing directory. Table C.13 below documents the names and contents of the scripts.

Table C.13
Scripts for Running Test Problems

File	Description
Makefile	Makefile for MCNP verification.
testinp.tar	Compressed input files for MCNP verification.
testmctl.sys	Compressed tally output files for MCNP verification.
testoutp.sys	Compressed MCNP output files for MCNP verification.
testdir1	Cross-section directory for MCNP verification.
testlib1	Cross-section data (type 1) for MCNP verification.

Upon completion of the regression test set, there should be a set of inp??m and inp??o files (?? = 01, 02, etc.). If the appropriate files to be compared exist, they are compared; if they do not exist, an error message is produced for that problem number. Differences between these runs and the standard show up in the dif?? files. Exact tracking is required for MCNP5 verification. Significant differences, that is, other than round-off in the last digit, may prove to be serious (e.g., compiler bugs). In such cases, the cause of the difference should be fully understood.

The test problems are neither good nor typical examples of MCNP problems. Rather, they are bizarre test configurations designed to exercise as many features as possible. The test set is constantly changed as new capabilities are added to MCNP and as bugs are corrected. The INPnn files are the same for all systems, but the answers, mctl??, differ slightly from system to system because of differences in arithmetic processors. The test set works on the basis of "particle tracking" in which the random walks must be identical. The test problem data library testlib1 is also only for testing purposes because it contains bad data used to test the code. The testlib1 data should not be used for real transport problems.

XII. CONVERTING CROSS-SECTION FILES WITH MAKXSF

The auxiliary code MAKXSF can be used to convert cross-section libraries from one format to another and to construct custom-designed cross-section libraries.

MCNP can read cross-section data from two types of files. Type 1 files are formatted and have sequential access. Type 2 files are unformatted, binary files and have direct access. RSICC distributes type 1 files because they are portable across platforms, but in that format the files are slow. The auxiliary program MAKXSF is provided for translating type 1 files into the faster access type 2 files. In the same manner, type 2 cross-sections can also be converted into type 1 cross-sections. You can also use MAKXSF to delete cross-section tables that you do not need and to reorganize the cross-section tables into custom-designed cross-section libraries. Please note that it is necessary to rebuild your type 2 cross-section files for MCNP5, but the procedure has been simplified. The program MAKXSF is compiled appropriately for your platform using the makefile system at the same time as MCNP is built. It is located in directory Source/datasrc.

The input files to MAKXSF are one or more existing cross-section libraries; a directory file, XSDIR, which describes the input cross-section libraries; and a file called SPECS that tells MAKXSF what it is supposed to do. The output files are one or more new cross-section libraries, a new directory file that describes the new cross-section libraries, and a file called TPRINT that contains any error messages generated during the run. The input and output cross-section libraries can be any combination of type 1 and type 2 files. The various types of cross-section libraries and the form and contents of the cross-section directory file are described in detail in Appendix F. The directory file XSDIR in the MCNP data package contains complete descriptions of all the cross-section files in that package. Printing it provides a useful reference. The sample SPECS file in the MCNP data package can be used with MAKXSF to create a complete set of type 2 files from the type 1 files provided. It should be noted that when re-executing MAKXSF, it is necessary to first delete a pre-existing TPRINT output file.

The SPECS file is a formatted sequential file with records not exceeding 80 characters. The data items in each record may start in any column and are delimited by blanks. The structure of the SPECS file is given in Table C.14 and an example is given in Table C.15.

Table C.14
SPECS File Record Structure

Record	**Contents**		
1	Name of old dir file	Name of new dir file	
2	Name of old xs lib*	Name of new xs lib	Type Recl* Epr*
3	Access route* entered into new directory file (or blank line)		
4 +	Nuclide list, if old xs lib is absent		
Blank record			
Where	* = optional		

The default for Epr (entries per record) is 512. Recl (record length) will be set appropriately by MAKXSF, depending on whether your platform specifies records in terms of words (OSF and PC) or bytes (all others), resulting in 8 bytes per entry. All data is stored double precision. You should NOT need to use these options.

Records 2 through 4+ can be repeated any number of times with data for additional new cross-section libraries. The SPECS file ends with a blank record. If “name of old cross–section library” exists on record 2, all nuclides from that library will be converted.

Table C.15
Example SPECS File

Record	Contents		
1	xsdir1	xsdir2	
2	el1	el2	2
3	home/scratch/el2		
4	rmccsab2	2	
5	datalib/rmccsab2		
6	7015.55c		
7	1001.50c		
8			

In Table C.15, the SPECS file starts with type 1 cross-section directory file XSDIR1, electron library EL1, and neutron libraries RMCCSA1 and RMCCS1. All nuclides on the electron data file EL1 are to be converted to a type 2 file called EL2. Records 4–7 tell MAKXSF to search all libraries listed in XSDIR1 until it finds nuclides 7015.55c and 1001.50c (which happen to be on RMCCSA1 and RMCCS1, respectively) and construct a new type 2 library RMCCSAB2 consisting of only these nuclides. The new directory file XSDIR2 will tell MCNP to look for the electron cross sections in /home/scratch/el2 and for the neutron cross sections in /datalib/rmccsab2.

If the type of the new cross-section file is specified to be 1 in record 2, only the name of the new cross-section file and the 1 for the type are read in that record. If the type in record 2 is 2, the record length and the number of entries per record can be specified, but it should not be necessary. If there is any difficulty, be sure that you are compiling MAKXSF with the option

```
-DDIRACCESS_RECL_WORD
```

for platforms specifying records in terms of words (OSF and PC) and without that flag for platforms specifying record lengths in bytes. Running the type 2 tests of the test suite from the install script is a good way to be sure the settings are correct.

The optional access route on record 3 of the SPECS file is a concatenation of a UNIX data path with the library name and becomes the fourth entry for each nuclide in the library in the XSDIR file.

It is not necessary to generate all the cross-section files that you will ever need in one MAKXSF run. You can combine and edit directory files at any time with a text editor or with another MAKXSF run. The only requirement is that you must give MCNP a directory file that points to all the cross-section files that are needed by the current problem. If you plan to run a long series of

MCNP problems that all use the same small set of cross-section tables, it might be convenient to generate with MAKXSF a small special-purpose cross-section file and directory file just for your project.

XIII. REFERENCES

1. E. C. Selcow, “New Fortran 90 Build System for MCNP (U),” Los Alamos National Laboratory research notes X-5:RN(U)01-11 (Rev. 1), May 8, 2001.
2. R.M. Stallman and R. McGrath, *GNU MAKE*, Free Software Foundation, Boston, MA (available URL: http//www.gnu.org) July 1998.
3. J. F. Briesmeister, Ed., “MCNP - A General Monte Carlo N-Particle Transport Code, Version 4C,” Los Alamos National Laboratory report LA-13709-M, March 2000.
4. G. W. McKinney, “Enhancement of the MCNP Install Package,” Los Alamos National Laboratory memorandum X-6:GWM-93-611, September 1993.
5. Richard Stallman, Paul Visscher, Bret Smith, and Luis M. Arteaga, "GNU’s not Unix!," (available URL: http://www.gnu.org/software) October 22, 2002.
6. Richard Stallman, Bob Chassell, Melissa Weishaus, Roland McGrath, and Paul D. Smith, *GNU make*, Version 3.80 (stable), Free Software Foundation, Inc. Boston, MA, October 2002.
7. D. Hagerty, M. Weisshauss, and E. Zaretskii, *GNU Software for MS-Windows and MS-DOS*, Free Software Foundation, Boston, MA (available URL: http//www.gnu.org/order/windows.html) June 2001.
8. "Fortran Compilers," (available URL: http:\\www5.compaq.com/fortran) 2002.
9. "Fortran" Lahey Computer Systems, Inc., Incline Village, NV (available URL: http:\\www.lahey.com) August 2002.
10. Mike Loukides and Andy Oram, "Programming with GNU Software," *Thompson Learning,* Chapter 7 (O’Reilly & Associates, Inc., Sebastopol, CA) April 1995.
11. Andy Oram and Steve Talbott, *Managing Projects with make*, 2nd Edition (O’Reilly & Associates, Inc., Sebastopol, CA) October 1991.
12. L. J. Cox, “Scripts for Manipulating MCNP Source Code,” Los Alamos National Laboratory memorandum X-5:99-13(U), April 2000.
13. L. J. Cox, “DMMP Upgrade for MCNP4C,” Los Alamos National Laboratory memorandum X-5:00-44(U), November 2000.
14. "American National Standard for Programming Language - Fortran - Extended," American National Standards Institute, ANSI X3.198-1992, New York, NY, September 1992.
15. L. J. Cox, "Standards For Writing and Documenting Fortran 90 Code," Los Alamos National Laboratory research notes X-5-RN(U)-00-28, March 28, 2000.
16. J. T. Goorley, F. B. Brown, and L.J. Cox, “MCNP5 Improvements for Windows PCs,” Nuclear Mathematical and Computational Sciences: A Century in Review, A Century Anew, Gatlinburg, Tennessee, April 6-11, 2003, on CD-ROM, American Nuclear Society, LaGrange Park, IL (2003).

APPENDIX D - MODIFYING MCNP

Appendix D contains information that users will need when they write modifications to MCNP. Other sections of the MCNP5 manual are also applicable, especially Chapter 2 for theory, Appendix E for variables and arrays in common, and Appendix F for the details of the cross-section tables.

Users sometimes have to modify MCNP for particular applications. In the past, most user modifications were for special sources or special tallies. The need for tally modifications has been greatly reduced by the generalization of the standard tallies in MCNP Versions 2 and 2B. The generalization of the standard sources in Version 3A has done the same for source modifications. However, users continue to find new applications for MCNP and will find new reasons to modify it.

I. PREPROCESSORS

When MCNP is compiled, it must be preprocessed to delete inappropriate system-dependent sections of code. Most Fortran-90 compilers have built-in capabilities for performing the preprocessing. For compilers that do not have these capabilities (e.g., Absoft compiler on PC systems), a *perl* script *fpp* can be used to perform the preprocessing prior to compilation. This *fpp* script is provided with the MCNP distribution files. See Appendix C for information on the CONFIG keyword required for selecting the appropriate system-dependent code and how to load MCNP on the various systems.

II. PROGRAMMING LANGUAGE

MCNP is written in standard Fortran 90.[1] Deviations from the standard are avoided because they make it more difficult to maintain portability. MCNP programming currently deviates from the standard in the following areas: system-dependent features and X-window graphics. The X-window graphics are implemented using C routines found in the distribution file *mc.c*.

Some dynamically allocated storage array in MCNP has an offset that is added to the first subscript expression in every reference to the array. This causes the value of the subscript expression to exceed the corresponding upper dimension bound for the array, which violates a Fortran rule. So far this has not caused trouble because the systems that MCNP currently runs on do not enforce the rule dynamically. The rule cannot be enforced at compile time because the offset is a variable.

While the Fortran standard is not specific as to the case of the source characters, MCNP source files are distributed primarily in lower case. The few exceptions to this are predicated by the following comment : “!*** must be upper case.” Changing the case of the source files should be avoided. Input to MCNP (via input files or the terminal) is now case insensitive. Case conversion is provided in subroutine NXTSYM.

III. SYMBOLIC NAMES

In MCNP, the name of every entity in COMMON and the name of every function subprogram is at least three characters long. The name of every local entity, including statement functions, is less than three characters long. Thus, the local or global status of a symbolic name can be determined at a glance.

The default implicit typing of Fortran is used for all integer and real entities in MCNP. When MCNP is compiled on any 32-bit computer, the statement "REAL(DKND) (A-H,O-Z)" is included in all program units, where "DKND" is a real type corresponding to double-precision. There is no usage of complex data types in MCNP. Logical entities are rare and are always local. The names of most, but not all, character entities begin with the letter H.

IV. SYSTEM DEPENDENCE

The use of standard Fortran goes a long way toward making MCNP run on many different computer systems. However, differences between the systems still have to be allowed to some extent.

The most important difference between hardware systems is that some have 64-bit words, whereas others, such as IBM and SUN machines, have 32-bit words. MCNP assumes that no more than 32 bits are available for most integer quantities. Floating-point data are always stored in 64-bit variables, including geometry specifications and cross-section data. The only use of 32-bit floating point variables is for a few plotting routine interfaces to X-Windows graphics routines on 32-bit systems. Geometry tracking in MCNP uses floating-point quantities without any special allowance for the fact that floating-point quantities are only approximations to the mathematical real numbers that they represent. This turns out to be a safe practice if the floating-point numbers have at least 48 bits of precision (but not with much less than 48).

The magnitude of a floating-point number cannot exceed about 10^{38} in most 32-bit machines; therefore, intermediate values do not exceed that limit. Sections of MCNP can still fail when the user attempts to generate numbers greater than 10^{38}.

The Fortran standard allows I/O units to be preconnected, which means that MCNP must avoid using certain unit numbers. Fortunately the preconnected unit numbers in all systems on which MCNP currently runs are numbers less than 10 or greater than 99. To avoid them, MCNP uses unit numbers in the thirties, forties, and fifties.

The Fortran standard does not specify the units for the length of the records of a direct-access file. Some systems define the length in bytes, some in words. This inconsistency does not affect the portability of MCNP. Direct access is used only for Type 2 cross-section files. The record length is read from the cross-section directory file and is entered explicitly in the input file to the auxiliary program MAKXSF, which writes the Type 2 files and the cross-section directory file. The question of the units occurs at the same time that the user chooses the size of the records, all in the context of the local system.

Some features of MCNP cannot be provided within the Fortran language. They are implemented by calling subroutines in local system libraries. Not all system-dependent features are available in all systems. The geometry-plotting feature is a special case. Its availability depends more on the local availability of GKS or of one of the other plotting packages – CGS, X-window, Lahey Winteractor, or DVF Quickwin – than on the nature of the computer system.

We have encountered bugs in compilers. Some of the comments in MCNP identify places where unusual programming has been done to get around compiler bugs.

System-dependent sections of code are set off by the preprocessor directives

#ifdef *name* ... #endif

See Appendix C for the names that are used and for how to use the preprocessors. As much as possible, we have tried to gather the system-dependent code in MCNP into only a few places, away from heavily mathematical parts of the program. One technique, exemplified by subroutine SETIDT, is to write a subroutine to do just one or several closely related system-dependent tasks. A subroutine of this sort consists of several alternative sections of code, one for each of the different systems. When that technique is impractical, we have tried to concentrate system-dependent code into the main program and into the top subroutines of the main sections. However, some system-dependent code may be found almost anywhere. Finally, coding practices forced on us by the limitations of certain systems, such as keeping all integer values within 32 bits, affect the entire program.

V. COMMON BLOCKS

Most of the common storage is in the Fortran module *mcnp_global*, which is used by all MCNP program units except some short-mathematical or system-oriented subprograms. This common storage is divided into nine separate common blocks. Dynamically allocated storage is in common block /DAC/, separate from statically allocated storage. Fixed, variable, and ephemeral data are separated to simplify maintenance of subroutine TPEFIL, which writes and reads the RUNTPE file. Fixed data are defined in setting up the problem, are written to RUNTPE only once, and are not changed during transport. Variable data are changed during transport and have to be written to RUNTPE for each restart dump. Ephemeral data, in common blocks /EPHCOM/ and /TSKCOM, are needed only during problem setup or only during the current history and are not written to RUNTPE. The particle description variables that have to be saved when a detector tally is made, when a DXTRAN particle is generated, and when a particle is banked are in common block /PBLCOM/, which is separate from the rest of the ephemeral data. Character data are in a common block /CHARCM/ separate from the numerical data in accordance with the rules of Fortran. Tables of hard-wired data are in a separate block called /TABLES/.

If any one of the following common blocks is changed, the length parameters associated with the block may need to be changed. The values of the length parameters are the numbers of numeric storage units in the floating point and integer portions of the common block.

.

common block	length parameters
/FIXCOM/	NFIXCM, LFIXCM
/VARCOM/	NVARCM, LVARCM
/EPHCOM/	NEPHCM, LEPHCM
/PBLCOM/	NPBLCM, LPBLCM
/TSKCOM/	NTSKCM, LTSKCM

The expressions for some of the length parameters include the parameter NDP2, which is the number of numeric storage units needed for a floating-point quantity. It has the value 1 on 64-bit machines and 2 on 32-bit machines. If any changes are made to /PBLCOM/ before the real variable ZPBLCM or between the integer variables NPA and MPBLCM, those changes must be echoed in the section of duplicate variables ending in “9” (XXX9, YYY9, etc.). The last two small common blocks, /GKSSIM/ and /MSGCOM/, are used in graphics routines and message passing routines, respectively.

VI. DYNAMICALLY ALLOCATED STORAGE

MCNP uses standard Fortran-90 dynamically allocated storage. The lengths and locations of all dynamically allocated arrays are defined during problem setup and are not changed during transport and output. Most of the arrays are included in three sets of arrays, one each for fixed, variable, and ephemeral data. The arrays used for statistics (SHSD, STT, NHSD), tallying (TAL), and for nuclear data tables (XSS, EXS) follow at the end. The lengths of most of the arrays are determined during the course of a preliminary reading of the INP file by subroutine PASS1. The INP file is then rewound and is read again by subroutine RDPROB. This time the data from INP are actually stored. The length of TAL is calculated in subroutine ITALLY. The length of XSS is calculated in subroutines under XACT. The parameter NDP2 is used to make the appropriate adjustments to the offsets where an integer array follows a floating-point array or vice versa.

VII. THE RUNTPE FILE

The RUNTPE file contains all the information needed to restart a problem in the continue-run mode. It can be used either to run more histories or to postprocess and plot tallies (see Appendix B).

The RUNTPE file is sequential and unformatted. It is written and read by subroutine TPEFIL in conjunction with subroutines RUNTPR and RUNTPW. The first part of RUNTPE is a sequence of records containing fixed data for the problem. The rest of RUNTPE is a sequence of restart dumps, each consisting of a sequence of records containing variable data. The first dump is written immediately after the records of fixed data are written, but before any transport calculations are done. Subsequent dumps are written from time to time during the initial run and during any continue-runs. If a continue-run is done with execute message item C, its dumps are written after the dump from which it started. If a continue-run is done with execute message item CN, its dumps are written after the fixed-data records. In either case, the number of dumps on the RUNTPE file can be limited by the fourth entry on the PRDMP card (see page 3-136).

Records in the Fixed-Data Part of the RUNTPE File

Identification Record

KOD*8	name of the code
VER*5	version identification
LODDAT*8	load date of the code
IDTM*19	machine designator, date and time
CHCD*10	charge code
PROBID*19	problem identification
PROBS*19	problem identification of surface source
AID*80	problem title
UFIL(3,6)*11	characteristics of user files
MXE	number of cross–section tables in the problem

Cross-section tables, MXE of them, one per record.

The contents of /FIXCOM/.

The part of /DAC/ that contains fixed data.Records in a Restart Dump

Records in a Restart Dump

Dump Identification Record

Current values of KOD, VER, LODDAT, IDTM, CHCD, and PROBID. PROBID is always the same as in the initial identification record.

The contents of /VARCOM/.

The part of /DAC/ that contains variable data.

The part of /DAC/ that contains tally information, if any.

Endfile record, which is overwritten by the next dump.

VIII. C FUNCTIONS

The MCNP source includes a file *mc.c* of C functions for X-window graphics that are implemented on most UNIX and PC systems. Use of these features requires an ANSI C compiler. The terse style of these C routines is historical, and is not typical of code written by experienced C programmers. Note also the use of 6 characters or less in those C function names referenced from Fortran. Other function names and variables reflect standard C programming.

IX. SUBROUTINE USAGE IN MCNP5

A. MCNP Structure

The general internal structure of MCNP is as follows:

Initiation (IMCN):

- Read input file (INP) to get dimensions (PASS1);
- Set up variable dimensions or dynamically allocated storage (SETDAS);
- Re-read input file (INP) to load input (RDPROB);
- Process source (ISOURC);
- Process tallies (ITALLY);
- Process materials specifications (STUFF) including masses without loading the data files;
- Calculate cell volumes and surface areas (VOLUME).

Interactive Geometry Plot (PLOT).

Cross-section Processing (XACT):

- Load libraries (GETXST);
- Eliminate excess neutron data outside problem energy range (EXPUNG);
- Doppler broaden elastic and total cross sections to the proper temperature if the problem temperature is higher than the library temperature (BROADN);
- Process multigroup libraries (MGXSPT);
- Process electron libraries (XSGEN) including calculation of range tables, straggling tables, scattering angle distributions, and bremsstrahlung.

MCRUN sets up multitasking and multiprocessing, runs histories (by calling TRNSPT, which calls HSTORY), and returns to OUTPUT to print, write RUNTPE dumps, or process another criticality (KCODE) cycle.

Under MCRUN, MCNP runs neutron, photon, or electron histories (HSTORY), calling ELECTR for electron tracks:

- Start a source particle (STARTP);
- Find the distance to the next boundary (TRACK), cross the surface (SURFAC) and enter the next cell (NEWCEL);
- Find the total neutron cross section (ACETOT) and process neutron collisions (COLIDN) producing photons as appropriate (ACEGAM);
- Find the total photon cross section (PHOTOT) and process photon collisions (COLIDP) producing electrons as appropriate (EMAKER);
- Use the optional thick–target bremsstrahlung approximation if no electron transport (TTBR);
- Follow electron tracks (ELECTR);
- Process optional multigroup collisions (MGCOLN, MGCOLP, MGACOL);
- Process detector tallies (TALLYD) or DXTRAN;
- Process surface, cell, and pulse height tallies (TALLY).

Periodically write output file, restart dumps, update to next criticality (KCODE) cycle, rendezvous for multitasking, and update detector and DXTRAN Russian roulette criteria, etc. (OUTPUT):

- Go to the next criticality cycle (KCALC);
- Print output file summary tables (SUMARY, ACTION);

- Print tallies (TALLYP);
- Generate weight windows (OUTWWG).

Plot tallies, cross sections, and other data (MCPLOT).
GKS graphics simulation routines.
PVM and MPI distributed processor multiprocessing routines.
Random number generator and control (MCNP_RANDOM).
Mathematics, character manipulation, and other slave routines.

B. History Flow

The basic flow of a particle history for a coupled neutron/photon/electron problem is handled in subroutine HSTORY. HSTORY is called from TRNSPT after the random number sequence is set up and the number of the history, NPS, is incremented. The flow of HSTORY is then as follows.

First, STARTP is called. The flag IPT is set for the type of particle being run: 1 for a neutron, 2 for a photon, and 3 for an electron. Some arrays and variables (such as NBNK, the number of particles in the bank) are initialized to zero. The starting random number is determined, and the branch of the history, NODE, is set to 1.

Next, the appropriate source routine is called. Source options are the standard fixed sources (SOURCB), the surface source (SURSRC), the KCODE criticality source (SOURCK), or a user-provided source (SOURCE). All of the parameters describing the particle are set in these source routines, including position, direction of flight, energy, weight, time, and starting cell (and possibly surface), by sampling the various distributions described on the source input control cards. Several checks are made at this time to verify that the particle is in the correct cell or on the correct surface, and directed toward the correct cell; then control is returned to STARTP.

Next in STARTP, the initial parameters of the first fifty particle histories are printed. Then some of the summary information is incremented (see Appendix E for an explanation of these arrays). Energy, time, and weight are checked against cutoffs. A number of error checks are made. TALLYD is called to score any detector contributions, and then DXTRAN is called (if used in the problem) to create particles on the spheres. The particles are saved with BANKIT for later tracking. TALPH is called to start the bookkeeping for the pulse height cell tally energy balance. The weight window game is played, with any additional particles from splitting put into the bank and any losses to Russian roulette terminated. Control is returned to HSTORY.

Back in HSTORY, the actual particle transport is started. For an electron source, ELECTR is called and electrons are run separately. For a neutron or photon source, TRACK is called to calculate the intersection of the particle trajectory with each bounding surface of the cell. The minimum positive distance DLS to the cell boundary indicates the next surface JSU the particle is heading toward. The distance to the nearest DXTRAN sphere DXL is calculated, as is the distance to time cutoff DTC, and energy boundary for multigroup charged particles DEB. The cross sections for cell ICL are calculated using a binary table lookup in ACETOT for neutrons and in PHOTOT for photons. (New to MCNP5, the total photon cross section returned by PHOTOT may include the photonuclear portion of the cross section if photonuclear physics is in use.) The total cross section is modified in EXTRAN by the exponential transformation if necessary. The distance PMF to the next collision is determined (if a forced collision is required, FORCOL is called and the uncollided part is banked). The track length D of the particle in the cell is found as the minimum of the distance PMF to collision, the distance DLS to the surface JSU, one mean free path DW (in the case of a mesh-based weight window), the distance DXL to a DXTRAN sphere, the distance DTC to time cutoff, or the distance DEB to energy boundary. TALLY then is called to increment any track length cell tallies. Some summary information is incremented. The particle's parameters (time, position, and energy) are then updated. If the particle's distance DXL to a DXTRAN sphere (of the same type as the current particle) is equal to the minimum track length D, the particle is terminated because particles reaching the DXTRAN sphere are already accounted for by the DXTRAN particles from each collision. If the particle exceeds the time cutoff, the track is terminated. If the particle was detected leaving a DXTRAN sphere, the DXTRAN flag IDX is set to zero and the weight cutoff game is played. The particle is either terminated to weight cutoff or survives with an increased weight. Weight adjustments then are made for the exponential transformation.

If the minimum track length D is equal to the distance-to-surface crossing DLS, the particle is transported distance D to surface JSU, and SURFAC is called to cross the surface and do any surface tallies (by calling TALLY) and to process the particle across the surface into the next cell by calling NEWCEL. It is in SURFAC that reflecting surfaces, periodic boundaries, geometry splitting, Russian roulette from importance sampling, and loss to escape are treated. For splitting, one bank entry of NPA particle tracks is made in BANKIT for an (NPA+1)-for-1 split. The bank is the IBNK array, and entries or retrievals are made with the GPBLCM and JPBLCM arrays (the bank operates strictly on a last-in, first-out basis). The history is continued by going back to HSTORY and calling TRACK.

If the distance to collision PMF is less than the distance to surface DLS, or if a multigroup charged particle reaches the distance to energy boundary DEB, the particle undergoes a collision. Everything about the collision is determined in COLIDN for neutrons and COLIDP for photons. COLIDN determines which nuclide is involved in the collision, samples the target velocity of the collision nuclide by calling TGTVEL for the free gas thermal treatment, generates and banks any photons (ACEGAM), handles analog capture or capture by weight reduction, plays the weight cutoff game, handles $S(\alpha, \beta)$ thermal collisions (SABCOL) and elastic or inelastic scattering (ACECOL). For criticality problems, COLIDK is called to store fission sites for subsequent generations. Any additional tracks generated in the collision are put in the bank. ACECAS and ACECOS determine the energies and directions of particles exiting the collision. Multigroup and multigroup/adjoint collisions are treated separately in MGCOLN and MGACOL that are called from COLIDN. The collision process and thermal treatments are described in more detail in Chapter 2 (see page 2-28).

COLIDP for photons is similar to COLIDN, and it covers the simple or the detailed physics treatments. The simple physics treatment is valid only for free electrons, i.e. it does not account for electron binding effects when sampling emission distributions; the detailed treatment is the default and includes form factors and Compton profiles for electron binding effects, coherent (Thomson) scatter, and fluorescence from photoelectric capture (see page 2-54). New as of MCNP5, COLIDP may also include photonuclear physics (if photonuclear physics is in use). Additionally, photonuclear biasing is available (similar to forced collisions) to split the photon (updating the weight by the interaction probabilities) and force one part to undergo a photoatomic collision and the second part to undergo a photonuclear collision. COLIDP samples for the collision nuclide, treats photonuclear collisions (in COLLPN), treats photoelectric absorption, or capture (with fluorescence in the detailed physics treatment), incoherent (Compton) scatter (with Compton profiles and incoherent scattering factors in the detailed physics treatment to account for electron binding), coherent (Thomson) scatter for the detailed physics treatment only (again with form factors), and pair production. Secondary particles from photonuclear collisions (either photons or neutrons) are sampled in COLLPN using the same routines (ACECAS and ACECOS) as for inelastic neutron collisions (see Elastic and Inelastic Scattering on page 2-35). Electrons are generated (EMAKER) for incoherent scatter, pair production, and photoelectric absorption. These electrons may be assumed to deposit all their energy instantly if IDES=1 on the PHYS:P card, or they may produce electrons with the thick–target bremsstrahlung approximation (default for MODE P problems, IDES=0 on the PHYS:P card), or they may undergo full electron transport (default for MODE P E problems, IDES=0 on the PHYS:P card). Multigroup or multigroup/adjoint photons are treated separately in MGCOLP or MGACOL.

After the surface crossing or collision is processed, control returns to HSTORY and transport continues by calling TRACK, where the distance to cell boundary is calculated. Or if the particle involved in the collision was killed by capture or variance reduction, the bank is checked for any remaining progeny, and if none exists, the history is terminated. Appropriate summary information is incremented, the tallies of this particular history are added to the total tally data by TALSHF, and a return is made to TRNSPT.

In TRNSPT, checks are made to see if output is required or if the job should be terminated because enough histories have been run or too little time remains to continue. For continuation, HSTORY is called again. Otherwise a return is made to MCRUN. MCRUN calls OUTPUT, which calls SUMARY to print the summary information. Then SUMARY calls TALLYP to print the tally data. Appendix E defines all of the MCNP variables as well as detailed descriptions of some important arrays.

X. *REFERENCES*

1. International Standards Organization, *Information Technology–Programming Languages–Fortran (Fortran 90)*, ISO/IEC 1539:1991, Geneva (1991).

APPENDIX E - GLOBAL CONSTANTS, VARIABLES, AND ARRAYS

This appendix contains information for users who need to modify MCNP. The first section is a dictionary of the symbolic names of the global entities in MCNP. The second section contains descriptions of some complicated arrays.

I. DICTIONARY OF SYMBOLIC NAMES

The global variables and arrays in MCNP are declared in Fortran 90 modules. The modules are referenced as needed by the MCNP program units.

COPYRIGHT_INFO	Copyright notice
MCNP_PARAMS	Double precision declaration and named constants
MCNP_DATA	Tables of constant data and character variables

Fortran 90 modules for all program units:

FIXCOM	Fixed variables that are unchanged after problem initiation
VARCOM	Variables that change throughout random walk and are needed for continue run
EPHCOM	Ephemeral variables that are not used in continue run
PBLCOM	Particle description variables required for banking particles
TSKCOM	Variables repeated on each multitasking processor
DYNAMIC ARRAYS	Dynamically allocated constructs (DAC) for variably dimensioned arrays
GKSSIM	Low-level graphics routines for GKS simulation subroutines
MSGCON	Multiprocessing message passing subroutines
MSGTSK	Turns multitasking lock on and off
MCNP_INPUT	Variables and constants for the IMCN program unit
MCNP_PLOT	Variables and constants for the PLOT geometry plotting section
MCNP_LANDAU	Landau electron data
MCNP_MODULE	Variables and constants for the MCPLOT tally and cross section

The symbolic names of the global constants, variables, and arrays are listed alphabetically below. The dimension bounds (for arrays), the value (for parameters), the module in which the variable is located, and a brief description are given for each entry. The adjustable dimension bound of each dynamically allocated array is indicated by a colon (:). The names of the entities in PBLCOM that end in 9 are not included in the dictionary. They are used only for temporarily saving the other entities in PBLCOM.

Variable	Contained in Module	Type	Description
abhi(2)	mcplot_module	R8	Upper x-axis limit of plot data.
ablo(2)	mcplot_module	R8	Lower x-axis limit of plot data.
aid	mcnp_data	character(len=80)	Title card of the initial run.
aid1	mcnp_data	character(len=80)	Title card of the current run.
aids	mcnp_data	character(len=80)	Title card of the surface source write run.
ajsh	mcnp_input	R8	Coefficient for surface area of a torus.
als	mcnp_plot	real	Current distance along a polyline.
amfp	tskcom	R8	Mean free paths to detector or DXTRAN sphere.
amx(:,:,:)	mcnp_global	R8, ALLOCATABLE	Matrices of surface coefficients from SCF.
aneut = 1.008664967d+0	mcnp_params	R8, parameter	Neutron mass in a.m.u.
ang(3)	tskcom	R8	Surface normal and cosine of track direction.
ara(:)	mcnp_global	R8, ALLOCATABLE	Areas of the surfaces in the problem.
aras(:,:)	mcnp_global	R8, ALLOCATABLE	Area calculated for each side of each surface.
asm(:,:)	mcnp_global	R8, ALLOCATABLE	Mesh indices of superimposed mesh.
asp(:)	mcnp_global	R8, ALLOCATABLE	Ionization loss straggling coefficients.
atsa(:,:)	mcnp_global	R8, ALLOCATABLE	Segment volume or area (for each side) of segment surface.
avgdn = 1.0d-24*avogad/aneut	mcnp_params	R8, parameter	1.e-24*Avogadro's number/neutron mass.
avlm(mlanc)	mcnp_landau	R8	Average electron Landau scattering lambda cutoff.
avogad = 6.022043446928244d+23	mcnp_params	R8, parameter	Avogadro's number.
avrm(6)	mcnp_data	character(len=1)	x,y,z,r,z,t identifier of superimposed mesh.
awc(:)	mcnp_global	R8, ALLOCATABLE	Atomic weights for density conversions.
awn(:)	mcnp_global	R8, ALLOCATABLE	Atomic weights for neutron kinematics.
awt(:)	mcnp_global	R8, ALLOCATABLE	Atomic weights from AWTAB card.
basis(9)	mcnp_plot	real	Basis vectors for plotting.
bbb(4,4)	mcnp_input	R8	Transformation matrix in volume calculator.
bbrem(mtop)	fixcom	R8	Bremsstrahlung energy bias factors.
bbv(:)	mcnp_global	R8, ALLOCATABLE	Equiprobable bins of a source function.
bcw(2,3)	varcom	R8	Coefficients of surface source biasing cylinder.
bnum	fixcom	R8	Bremsstrahlung bias number.
calph(maxi)	fixcom	R8	Cosines of electron scattering group boundaries.
cbwf	tskcom	R8	Weight multiplier for source direction bias.
chcd	mcnp_data	character(len=10)	LANL Charge code.
chite(5)	mcnp_plot	real	Character height parameters.
chup(2)	mcnp_plot	real	Character up vector.
clev(mclevs)	mcplot_module	R8	Contour levels.
cmg(:)	mcnp_global	R8, ALLOCATABLE	Energy-dependent importances.
cmult	tskcom	R8	Collision multiplicity.
coe(:,:,:)	mcnp_global	R8, ALLOCATABLE	Parametric coefficients of plot curves.
coincd	fixcom	R8	Distance of coincidence. See DBCN(9).
coll(mipt)	varcom	R8	Number of collisions in problem.
colltc(mipt)	tskcom	R8	Task copy of COLL.
color	mcnp_plot	type	Type for color name and RGB values.
color_by = 'mat '	mcnp_plot	character(len=4)	Color-fill variable name.
color_mode = 0	mcnp_plot	I4	Color-fill mode 0=distinct, 1=gradient
colors	mcnp_plot	type(color), dimension(ncolor+7)	Defined colors for plotting.

Variable	Contained in Module	Type	Description
colout(3,11)	tskcom	R8	Energy, cosine, time (delayed neutrons) of particles from collisions.
com	mcnp_iofiles	character(len=8)	Plotter command input file name.
comout	mcnp_iofiles	character(len=8)	Plotter command output file name.
contur(3)	mcplot_module	R8	Contour level limits and interval.
cp1	ephcom	R8	Computer time used after beginning MCRUN.
cp2(:)	mcnp_global	R8, ALLOCATABLE	Computer time used so far for each processor.
cp3	ephcom	R8	Computer time of multiprocessing subtasks.
cpk	varcom	R8	Computer time for settling in a KCODE problem.
crs(:)	mcnp_global	R8, ALLOCATABLE	Intersections of plot curves.
cthick = .02	mcplot_module	R8	Thickness of plot line.
ctme	ephcom	R8	User requested ctm endtime.
cts	varcom	R8	Computer time used for transport in current problem including previous runs, if any.
dbcn(30)	varcom	R8	Debug controls from DBCN card.
ddet	tskcom	R8	Distance from collision point to detector.
ddg(2,mxdt)	fixcom	R8	Controls for detector diagnostics.
ddm(:,:)	mcnp_global	R8, ALLOCATABLE	Size and history of largest score of each tally.
ddn(:,:)	mcnp_global	R8, ALLOCATABLE	Detector diagnostics.
ddx(mipt,2,mxdx)	fixcom	R8	Controls for DXTRAN diagnostics.
deb	tskcom	R8	Distance to energy-group boundary.
dec(:,:)	mcnp_global	R8, ALLOCATABLE	Detector contributions by cell.
deltas	tskcom	R8	Delta s offset in image pixel.
deltat	tskcom	R8	Delta t offset in image pixel.
den(:)	mcnp_global	R8, ALLOCATABLE	Mass densities of the cells.
dfdmp = -60.0d+0	mcnp_params	R8, parameter	Default dump interval.
dftint = 100.0d+0	mcnp_params	R8, parameter	Default interval between time interrupts.
dknd = selected_real_kind(15,307)	mcnp_params	I4, parameter	8-byte real kind.
dls	pblcom	R8	Distance to next boundary.
dmp	varcom	R8	Dump control from PRDMP card.
dnb	fixcom	R8	Delayed neutron bias (4th PHYS:N entry).
dptb(:,:)	mcnp_global	R8, ALLOCATABLE	PERT card density change. See page E–42.
drc(:,:)	mcnp_global	R8, ALLOCATABLE	Data saved for coincident detectors.
drs(:)	mcnp_global	R8, ALLOCATABLE	Electron energy substep range.
dtc	pblcom	R8	Distance to time cutoff.
dti(mlgc)	tskcom	R8	Positive distances to surfaces.
dxc(:,:)	mcnp_global	R8, ALLOCATABLE	DXTRAN contributions by cell.
dxcp(:,:,:)	mcnp_global	R8, ALLOCATABLE	DXTRAN cell probabilities.
dxd(:,:,:)	mcnp_global	R8, ALLOCATABLE	DXTRAN diagnostics.
dxl	pblcom	R8	Distance to nearest DXTRAN sphere.
dxw(mipt,3)	fixcom	R8	DXTRAN weight cutoffs.
dxx(mipt,5,mxdx)	fixcom	R8	DXTRAN sphere parameters.
eaa(:)	mcnp_global	R8, ALLOCATABLE	Average values of source distributions.
eacc(4)	varcom	R8	Weight and energy of electrons above EMAX.
eacctc(4)	tskcom	R8	Task copy of EACC.
ear(:)	mcnp_global	R8, ALLOCATABLE	Ionization loss straggling coefficients.
eba(:,:)	mcnp_global	R8, ALLOCATABLE	Unbiased cumulative prob. for photon/elec bremsstrahlung energy loss fractions.

Variable	Contained in Module	Type	Description
ebd(:,:)	mcnp_global	R8, ALLOCATABLE	Bremsstrahlung energy distributions.
ebl(:)	mcnp_global	R8, ALLOCATABLE	Energy group bounds for photon production.
ebt(:,:)	mcnp_global	R8, ALLOCATABLE	Thick-target bremsstrahlung distributions.
ecf(mipt+1)	fixcom	R8	Particle energy cutoffs.
ech(:,:,:)	mcnp_global	R8, ALLOCATABLE	Bremsstrahlung angular distributions.
edg(:)	mcnp_global	R8, ALLOCATABLE	K-edge energies.
eee(:)	mcnp_global	R8, ALLOCATABLE	Energy grid for electron cross-section tables.
eek(:)	mcnp_global	R8, ALLOCATABLE	K x-ray energies.
efac	fixcom	R8	Ratio of adjacent energies in array EEE.
eg0	tskcom	R8	Energy of the particle before last collision.
egg(:,:)	mcnp_global	R8, ALLOCATABLE	Electron scattering angle distribution.
elc(mipt)	pblcom	R8	Energy cutoffs in the current cell.
elp(:,:)	mcnp_global	R8, ALLOCATABLE	Cell-dependent energy cutoffs.
emcf(mipt)	fixcom	R8	Cut-in energy for analog capture (n,p) & detailed photon physics (p).
emx(mipt)	fixcom	R8	Maximum energy in problem for particle type.
enum	fixcom	R8	Secondary electron production bias number.
eqlm(mlam)	mcnp_landau	R8	Landau electron scattering equiprobable bins.
erg	pblcom	R8	Particle energy.
ergace	tskcom	R8	Raw energy extracted from cross-section table.
esa(:)	mcnp_global	R8, ALLOCATABLE	Cut-in energies for thermal S(A,B) tables.
etspl(2,mipt,42)	fixcom	R8	Controls for energy & time splitting. energy/value & time/ value pairs for energy (1) & time (2) splitting/RR by particle type.
euler = .577215664901532861d+0	mcnp_params	R8, parameter	Euler constant used in electron transport.
ewwg(:)	mcnp_global	R8, ALLOCATABLE	Energy bins for weight-window generator.
exms = ‘ ‘	mcnp_data	character(len=80)	Execute message.
exs(:)	mcnp_global	R8, pointer	Electron cross sections.
exsav(2)	mcnp_plot	real	Saved extents.
extent(2) = (/100.,100./)	mcnp_plot	real	Extents for plotting.
fdd(:,:)	mcnp_global	R8, ALLOCATABLE	Inhibitors of source frequency duplication.
febl(:,:)	mcnp_global	R8, ALLOCATABLE	Number, weight of photons produced in each energy group.
fes(33)	mcnp_data	R8, parameter	Fission energy spectrum for KCODE source.
fim(:,:)	mcnp_global	R8, ALLOCATABLE	Particle cell importances.
fiml(mipt)	pblcom	R8	Importance of the current cell.
fismg	pblcom	R8	Multigroup importance.
flam(mlanc)	mcnp_landau	R8	Landau electron scatter cutoff.
flc(:)	mcnp_global	R8, ALLOCATABLE	Electron landau scattering energy cutoff.
flx(:)	mcnp_global	R8, ALLOCATABLE	Tally of multigroup cell fluxes.
fm(:)	fmesh_mod	type(fmarray), ALLOCATABLE	Array of mesh tallys.
fmarray	fmesh_mod	type	Structure containing items unique for each mesh tally. See page E–45.
fme(:)	mcnp_global	R8, ALLOCATABLE	Atom fractions from M cards.
fmg(:)	mcnp_global	R8, ALLOCATABLE	Table for biased adjoint sampling.
fmtal(:)	fmesh_mod	type(fm_temp_array), ALLOCATABLE	History scores for each mesh.
fm_temp_array	fmesh_mod	type	Mesh-tally values for each history. See page E–46.
fnw	fixcom	R8	Normalization of generated weight windows.

Variable	Contained in Module	Type	Description
for(:,:)	mcnp_global	R8, ALLOCATABLE	Controls for forced collisions.
fpi	ephcom	R8	Reciprocal of number of histories.
frc(:)	mcnp_global	R8, ALLOCATABLE	Fraction of source cutoff by energy limits.
freq	ephcom	R8	Interval between MCRUN calls of MCPLOT.
fscon = 137.0393d+0	mcnp_params	R8, parameter	Inverse fine-structure constant.
fso(:)	mcnp_global	R8, ALLOCATABLE	Fission source for KCODE.
fst(:)	mcnp_global	R8, ALLOCATABLE	Bremsstrahlung bias correction factors.
ftt(:)	mcnp_global	R8, ALLOCATABLE	TTB bremsstrahlung bias correction factors.
gbnk(:)	mcnp_global	R8, ALLOCATABLE	The floating-point part of the bank.
gephcm(nephcm)	ephcom	R8	Equivalence to real part of /EPHCM/.
gfixcm(nfixcm)	fixcom	R8	Equivalence to real part of /FXCOM/.
gmg(:)	mcnp_global	R8, ALLOCATABLE	Other-way fluxes for biased adjoint sampling.
gpb9cm(mpb,npblcm+1)	pblcom	R8	Floating-point stack in /PBLCM/.
gpblcm(npblcm+1)	pblcom	R8	Array name of floating-point part of /PBLCM/. See page E–28.
gpt(mipt)	mcnp_data	R8, parameter	Masses of particles.
gtskcm(ntskcm)	tskcom	R8	Equivalence to real part of /TSKCM/.
gvl(:)	mcnp_global	R8, ALLOCATABLE	Group-center velocities.
gwt(:)	mcnp_global	R8, ALLOCATABLE	Minimum gamma production weights.
hbln(maxv,4)	mcnp_data	character(len=3), parameter	Names of SDEF and SSR source variables.
hblw(maxw)	mcnp_data	character(len=3), parameter	Names of SSW source variables.
hcs(2)	mcnp_data	character(len=7)	‘cell’ and ‘surface’.
hft(mkft)	mcnp_data	character(len=3), parameter	Names of FT-card special treatments.
hip = ‘npe’	mcnp_data	character(len=mipt+1)	Initials of particle names.
hlbl(43)	mcplot_module	character(len=40)	Cross-section plot reaction labels.
hmes	mcnp_data	character(len=69)	Expire (bad trouble) message.
hnp(mipt)	mcnp_data	character(len=8)	Names of particles.
hovr = kod	mcnp_data	character(len=8)	Name of the current code section.
hsb(nsp)	fixcom	R8	Statistical analysis history score grid.
hsll = 1.0d-30	mcnp_params	R8,parameter	History score lower bin bound.
hsub	mcnp_data	character(len=6)	Subroutine where expire (bad trouble) occurred.
htn = ‘cdytpmgue’	mcnp_data	character(len=9)	Legal ZAID suffixes.
huge = 1.0d+36	mcnp_params	R8, parameter	A very large number.
i4knd = selected_int_kind(9)	mcnp_params	I4, parameter	4-byte integer kind.
i8fixcm(l8fixcm)	fixcom	I4(i8knd)	Equivalence to integer*8 part of /FXCOM/.
i8knd = selected_int_kind(18)	mcnp_params	I4, parameter	8-byte integer kind.
iafg(:)	mcnp_global	I4, ALLOCATABLE	Reentrant particle weight window generator flag.
iap	pblcom	I4	Program number of the next cell.
iax	tskcom	I4	Flag for presence of AXS vector.
ibad	fixcom	I4	Flag for simple bremsstrahlung distribution.
ibc	tskcom	I4	Index of the tally cosine bin.
ibe	tskcom	I4	Index of the tally energy bin.
ibin = ‘fdusmcet&’	mcnp_data	character(len=9)	Tally-bin type symbols.
i_bins(:,:,:)	fmesh_mod	I4, ALLOCATABLE	Index numbers for mesh tally bins containing scores.
ibl(8,2)	mcplot_module	I4	Bin range for plotting each tally bin type.

Variable	Contained in Module	Type	Description
ibnk(:)	mcnp_global	I4, ALLOCATABLE	The integer part of the bank.
ibs	tskcom	I4	Index of the tally segment bin.
ibt	tskcom	I4	Index of the tally time bin.
ibu	tskcom	I4	Index of the tally user bin.
ic0	tskcom	I4	Index for sampling ENDF law 67 neutrons.
ica = 0	mcnp_input	I4	Index of the type of the current input card.
icl	pblcom	I4	Program number of the current cell.
iclp(5,0:mxlv)	tskcom	I4	Multilevel source cell and lattice indices.
icn	mcnp_input	I4	Number in columns 1-5 of current input card.
icol	mcnp_plot	I4	GKS graphics color.
icolor(mplm)	mcnp_plot	I4	Shading index for materials in plot.
icrn(:,:)	mcnp_global	I4, ALLOCATABLE	Surfaces and label of each cell corner.
ics	ephcom	I4	Flag for error on current input card.
icurs = 0	mcnp_plot	I4	Cursor flag.
icurs1 = 0	mcnp_plot	I4	Flag for saving initial conditions for cursor.
icut(2)	mcplot_module	I4	Index of lower x-axis limit of plot data.
icw	fixcom	I4	Reference cell for generated weight windows.
icx	mcnp_input	I4	Flag for asterisk on current input card.
id0	tskcom	I4	Data index for neutron scattering ENDF law 67.
IDEF = 4	mcnp_params	I4, parameter	Default integer kind.
idefv(maxv)	fixcom	I4	Flags for presence of variable names on SDEF.
ides	fixcom	I4	Flag to inhibit electron production by photons.
idet	tskcom	I4	Index of the current detector.
idmp	ephcom	I4	Number of the dump from which to start a continue run.
idna(:)	mcnp_global	I4, ALLOCATABLE	Macrobody surface facet names. See page E–44.
idne(:)	mcnp_global	I4, ALLOCATABLE	List of identical surfaces. See page E–44.
idns(:)	mcnp_global	I4, ALLOCATABLE	Locator in IDNE for list of identical surfaces. See page E–44.
idnt(:)	mcnp_global	I4, ALLOCATABLE	Program surface number of master identical surfaces. See page E–44.
idrc(mxdt)	fixcom	I4	Links between master and slave detectors.
idtm	mcnp_data	character(len=19)	Current date and time.
idtms	mcnp_data	character(len=19)	IDTM of the surface source write run.
idum(1:n_idum) = 0	mcnp_debug	I4(i4knd)	Data from IDUM card.
idx	pblcom	I4	Number of the current DXTRAN sphere.
iet	tskcom	I4	Index of the current S(a,b) table.
iets(mipt)	fixcom	I4	Energy/Time split flags for particle types. 0 = no esplt/tsplt, 1 = esplt/tsplt.
iex	pblcom	I4	Index of the current cross-section table.
iexp	pblcom	I4	IEX from previous collision.
ifft	fixcom	I4	Flag for FT-card treatments SCX or SCD.
ifile	ephcom	I4	I/O unit of current plot input file.
ifip(mipt+1)	mcnp_input	I4	Flag for presence of IP card.
ifl(:)	mcnp_global	I4, ALLOCATABLE	Nodes at cell exits, for tally flagging.
ifree(2) = (/ 7,8 /)	mcplot_module	I4	Indices of current free variables.
igm	fixcom	I4	Total number of energy groups.
iii	pblcom	I4	First lattice index of particle location.
iint(:)	mcnp_global	I4, ALLOCATABLE	Surfaces crossed at the intersections.
iitm	mcnp_input	I4	Integer form of current item from input card.

Variable	Contained in Module	Type	Description
ikz	fixcom	I4	Number of KCODE cycles to skip before tallying.
ilbl(9)	mcnp_data	character(len=8)	Names of the tally bins.
iln	ephcom	I4	Count of lines of input data.
iln1	ephcom	I4	Saved count of lines of input data.
imd	tskcom	I4	Indicator of monodirectional plane source.
imesh(nmkey)	fixcom	I4	Counts number of entries on each MESH card keyword.
img	fixcom	I4	Flag for electron-photon multigroup problem.
imt	fixcom	I4	Number of times the surface source will occur.
iname	mcnp_iofiles	character(len=8)	Base name holder.
indt	fixcom	I4	Count of entries on MT cards.
inform	ephcom	I4	Flag for output to plot user.
ink(mink)	fixcom	I4	Output controls from PRINT card.
inp	mcnp_iofiles	character(len=8)	Problem spec file name.
inpd	ephcom	I4	TFC rendezvous frequency (5th PRDMP entry).
intrpol(:)	fmesh_mod	I4, ALLOCATABLE	Mesh tally response function interpolation method.
ioid	fixcom	I4	Flag for VOID card.
iovr	ephcom	I4	Index of the current code section.
ipac2(:)	mcnp_global	I4, ALLOCATABLE	Flags used to distinguish between population and tracks entering cell.
ipan(:)	mcnp_global	I4, ALLOCATABLE	Pointers into PAN for all the cells.
ipct = 1	mcplot_module	I4	Flag for percent contours.
iper	tskcom	I4	Current perturbation index.
ipert	fixcom	I4	Number of PERT card keywords, dimension of RPTB.
iphot	fixcom	I4	PHYS:E flag for electrons to produce photons.
ipl = 0	mcnp_input	I4	Pointer into RTP for current tally card.
iplt	fixcom	I4	Indicator of how weight windows are to be used.
ipnt(:,:,:)	mcnp_global	I4, ALLOCATABLE	Pointers into RTP. See page E–35.
iprpts = 0	mcplot_module	I4	Flag for printing instead of plotting points.
ipsc	tskcom	I4	Type of PSC calculation to make.
ipt	pblcom	I4	Type of particle.
iptal(:,:,:)	mcnp_global	I4, ALLOCATABLE	Guide to tally bins. See page E–31.
iptb(:,:)	mcnp_global	I4, ALLOCATABLE	Pointers to RPTB array. See page E–43.
iptr	ephcom	I4	PTRAC option flag.
iptra(nptr)	ephcom	I4	Pointer to PTR() for each PTRAC keyword.
ipty(mipt)	fixcom	I4	Particle types to be written to surface source.
iqc	mcnp_plot	I4	Index of current curve of current surface.
irc	mcnp_input	I4	First column of data field of input line.
irmc(50)	varcom	I4	Data from IRMC input card.
irs	mcnp_input	I4	Index of the current source distribution.
irt	tskcom	I4	Counter for renormalizing direction cosines.
irup	ephcom	I4	Flag set by user with ctrl-c interrupt.
isb	fixcom	I4	Control parameter for adjoint biasing.
isbm = 100	mcplot_module	I4, PARAMETER	X-dimension of contour or 3D sub-block.
isef(:,:)	mcnp_global	I4, ALLOCATABLE	Source position tries and rejections.
isic(maxv)	tskcom	I4	Distribution used for each source variable.
ism(3)	fixcom	I4	Number of fine mesh surfaces in x,y,z or r,z,t.
ispn	fixcom	I4	Flag for photonuclear physics.
iss(:,:)	mcnp_global	I4, ALLOCATABLE	Surfaces where input surface source is to start.

Variable	Contained in Module	Type	Description
issw	fixcom	I4	Flag to cause surface source file to be written.
ist	varcom	I4	Where in FSO to store next KCODE source neutron.
ist0	varcom	I4	Saved IST value to rerun lost history.
istern	fixcom	I4	Memory offset for ITS3.0 Sternhiemer, Berger, Seltzer electron density effect treatment option.
istrg	fixcom	I4	Flag to inhibit electron energy straggling.
isub(ndef)	mcnp_iofiles	character(len=8)	Runtime file names.
ital	tskcom	I4	Index of the current tally.
itask	ephcom	I4	Number of active tasks.
itds(:)	mcnp_global	I4, ALLOCATABLE	Tally specifications. See page E–32.
iterm	ephcom	I4	Type of plotting display.
itfc = 0	mcplot_module	I4	Type of TFC or KCODE plot.
itfxs	ephcom	I4	Flag to indicate need for total-fission tables.
itik(2) = (/ 0,0 /)	mcplot_module	I4	Number of divisions in each axis.
iti(mlgc)	tskcom	I4	Surface numbers associated with DTI values.
ititle(7) = (/ (0,ij = 1,7) /)	mcplot_module	I4	Flags for existence of titles.
itotnu	ephcom	I4	Flag for total vs. prompt nubar.
its30	fixcom	I4	Flag for ITS3.0 electron treatment.
itty	mcnp_iofiles	I4	I/O unit for terminal keyboard.
iu1 = 39	mcnp_params	I4, parameter	I/O unit for a scratch file.
iu2 = 40	mcnp_params	I4, parameter	I/O unit for another scratch file.
iu3 = 48	mcnp_params	I4, parameter	I/O unit for another scratch file.
iu4 = 49	mcnp_params	I4, parameter	I/O unit for another scratch file.
iub = 60	mcnp_params	I4, parameter	I/O unit for bank backup file.
iuc = 44	mcnp_params	I4, parameter	I/O unit for output plot command file.
iud = 35	mcnp_params	I4, parameter	I/O unit for directory of cross-section tables.
iui = 31	mcnp_params	I4, parameter	I/O unit for problem input file.
iuk = 47	mcnp_params	I4, parameter	I/O unit for input plot command file.
iumt = 54	mcnp_params	I4, parameter	I/O unit for the mesh tally output file.
iunr	fixcom	I4	Number of nuclides with probability tables (negative if multi-temperature correlations).
iuo = 32	mcnp_params	I4, parameter	I/O unit for problem output file.
iuou	ephcom	I4	Indicator that OUTP has been opened.
iup = 37	mcnp_params	I4, parameter	I/O unit for intermediate file of plots.
iupc = 51	mcnp_params	I4, parameter	PTRAC scratch file.
iupw = 50	mcnp_params	I4, parameter	PTRAC output file.
iupx = 52	mcnp_params	I4, parameter	Unit of file for writing plot print points.
iur = 33	mcnp_params	I4, parameter	I/O unit for file of restart dumps.
ius = 38	mcnp_params	I4, parameter	I/O unit for KCODE source file.
iusc = 43	mcnp_params	I4, parameter	I/O unit for surface source scratch file.
iusr = 42	mcnp_params	I4, parameter	I/O unit for surface source input file.
iusw = 41	mcnp_params	I4, parameter	I/O unit for surface source output file.
iut = 45	mcnp_params	I4, parameter	I/O unit for output MCTAL file.
iuw = 53	mcnp_params	I4, parameter	I/O unit for input WWINP file.
iuw1 = 82	mcnp_params	I4, parameter	I/O unit for output WWONE file.
iuwe = 81	mcnp_params	I4, parameter	I/O unit for output WWOUT file.
iux = 34	mcnp_params	I4, parameter	I/O unit for files of cross-section tables.
iuz = 46	mcnp_params	I4, parameter	I/O unit for tally input file.

Variable	Contained in Module	Type	Description
ivdd(maxv)	fixcom	I4	For each dependent source variable, the number of the source variable depended upon.
ivdis(maxv)	fixcom	I4	Distribution number for each source variable.
ivord(maxv)	fixcom	I4	Source variable numbers in sampling order.
iw0	tskcom	I4	Index for sampling ENDF law 67 neutrons.
iwwg	fixcom	I4	Weight window generator flag. = –1 fatal error on WWG or MESH cards = 0 no weight window generation = 1 cell-based generator or mesh-based generator with mesh from MESH card = 2 mesh-based generator with mesh from WWINP file
ixak	varcom	I4	Where in FSO to get next KCODE source neutron.
ixak0	varcom	I4	Saved IXAK value to rerun lost history.
ixc(:,:)	mcnp_global	I4, ALLOCATABLE	Encoded cross-section directory entries.
ixcos	tskcom	I4	Pointer to cosine table for PSC calculation.
ixl(:,:)	mcnp_global	I4, ALLOCATABLE	Encoded ZAIDs.
ixre	tskcom	I4	Index of the collision reaction.
ixs(:,:,:)	mcnp_global	I4, ALLOCATABLE	Photonuclear secondary particle types.
iza(:)	mcnp_global	I4, ALLOCATABLE	ZAs from M cards.
izn(:)	mcnp_global	I4, ALLOCATABLE	Photonuclear isotope overrides.
j3d = 0	mcplot_module	I4	Flag: if 2 free variables, plot is 3D not 2D.
jap	tskcom	I4	Program number of the next surface.
jasq(:)	mcnp_global	I4, ALLOCATABLE	Macrobody facets for source surfaces.
jasr(:,:)	mcnp_global	I4, ALLOCATABLE	Input surface source surfaces to be used.
jasw(:)	mcnp_global	I4, ALLOCATABLE	Surfaces from surface source input file.
jbd	tskcom	I4	Indicator for scoring flagged (or direct) bin.
jbnk	tskcom	I4	Number of particles in the bank in memory.
jchar	ephcom	I4	Current character position in input line.
jcond(:)	mcnp_global	I4, ALLOCATABLE	Flags for M card COND option.
jemi(:)	mcnp_global	I4, ALLOCATABLE	Flags for M card GAS option.
jephcm(lephcm)	ephcom	I4	Equivalence to integer part of /EPHCM/.
jev	tskcom	I4	Count of event-log lines printed.
jfcn	ephcom	I4	Flag indicating CN is in the execute message.
jfixcm(lfixcm)	fixcom	I4	Equivalence to integer part of /FIXCM/.
jfl(:)	mcnp_global	I4, ALLOCATABLE	Nodes of surface crossings, for tally flagging.
jfq(:,:)	mcnp_global	I4, ALLOCATABLE	Order for printing tally results.
jft(:)	mcnp_global	I4, ALLOCATABLE	User bin indexes for special tally treatments.
jgf	ephcom	I4	Indicator that plot goes to graphics metafile.
jgm(mipt)	fixcom	I4	Number of energy groups for each particle.
jgp	pblcom	I4	Neutron: particle energy group number. Photon: flag for photon generated electron progeny. Electron: flag for positron.
jgxa(2)	ephcom	I4	Flag for active graphics ports.
jgxo(2)	ephcom	I4	Flag for open graphics ports.
jjj	pblcom	I4	Second lattice index of particle location.
jlbl(2,9)	mcplot_module	I4	Key to cross-section plot labels.
jlim(2) = (/ 0,0 /)	mcplot_module	I4	Flag that user-supplied limits are in effect.
jloc = 0	mcnp_plot	I4	Flag for LOCATE command.
jlock(nlocks)	tskcom	I4	Status variable for multithreading memory/io lock.

Variable	Contained in Module	Type	Description
jmd(:)	mcnp_global	I4, ALLOCATABLE	Material mixture number pointer.
jmt(:)	mcnp_global	I4, ALLOCATABLE	S(a,b) material number pointer.
jovr(novr)	ephcom	I4	Flags for code sections to be executed.
jpb9cm(mpb,lpblcm+1)	pblcom	I4	Stored values of JPBLCM.
jpblcm(lpblcm+1)	pblcom	I4	Array name for the common block. See page E–28
jptal(:,:)	mcnp_global	I4, ALLOCATABLE	Basic tally information. See page E–31.
jptb(:)	mcnp_global	I4, ALLOCATABLE	Flag if perturbation correction required.
jrad	varcom	I4	Latch for warning of unusual radius sampling.
jrwb(21,mipt)	mcnp_data	I4, parameter	PWB columns corresponding to values of NTER.
jsbm = isbm	mcplot_module	I4, PARAMETER	Y-dimension of contour or 3D sub-block.
jscal = 0	mcnp_plot	I4	Indicator of type of scales wanted on plot.
jscn(:)	mcnp_global	I4, ALLOCATABLE	Source comments.
jsd(4,33)	mcnp_input	I4	Flags for distributions that need space in SSO.
jsf(mjsf)	mcnp_data	I4, parameter	Numerical names of built-in source functions.
jsq(:)	mcnp_global	I4, ALLOCATABLE	Macrobody facets for source surfaces.
jss(:)	mcnp_global	I4, ALLOCATABLE	Surfaces for surface source output file.
jst(:,:)	mcnp_global	I4, ALLOCATABLE	Stack of points in the current piece of cell.
jsu	pblcom	I4	Program number of the current surface.
jta(2)	mcnp_plot	I4	Flag for active workstations.
jtasks	ephcom	I4	Number of PVM subtasks, >0 for load balancing.
jtf(:,:)	mcnp_global	I4, ALLOCATABLE	Indices for fluctuation charts. See page E–30.
jtfc	ephcom	I4	Flag to indicate TFC update is due.
jtls	tskcom	I4	Count of the scores in the current history.
jtlx	fixcom	I4	Latch for the TALLYX warning message.
jtr(:)	mcnp_global	I4, ALLOCATABLE	Transformation numbers from surface cards.
jtskcm(ltskcm)	tskcom	I4	Equivalence to integer part of /TSKCM/.
jtty	mcnp_iofiles	I4	I/O unit for terminal printer or CRT.
jui	mcnp_input	I4	Unit number of the current input file.
jun(:)	mcnp_global	I4, pointer	Universe number of each cell.
junf	fixcom	I4	Flag for repeated structures.
jvc(:)	mcnp_global	I4, ALLOCATABLE	Vector numbers from the VECT card.
jvp	ephcom	I4	Flag for square viewport.
jxs(:,:)	mcnp_global	I4, ALLOCATABLE	Blocks of pointers into cross-section tables.
kaw(:)	mcnp_global	I4, ALLOCATABLE	Values of Z*1000+A from the AWTAB card.
kbin(8,2)	mcplot_module	I4	Bin range for plotting each tally bin type.
kbnk	tskcom	I4	Task offset for IBNK array.
kbp	ephcom	I4	Interrupt flag for multitasking mode.
kc8	varcom	I4	-1/0/1 KCODE cycle: settle/not KCODE/active.
kcl(:,:)	mcnp_global	I4, ALLOCATABLE	Cell numbers of grid points in the plot window.
kcolor(ncolor+7)	mcnp_plot	I4	Color indices for geometry plot.
kcp(:)	mcnp_global	I4, ALLOCATABLE	Descriptions of multi-level source cells.
kcsf	varcom	I4	Flag for KCODE source overlap.
kct	varcom	I4	Number of KCODE cycles to run.
kcy	varcom	I4	Current KCODE cycle.
kcz	varcom	I4	The last KCODE cycle completed.
kdb	tskcom	I4	Flag for lost particle or long history.
kdbnps	ephcom	I4	NPS of bad-trouble history in multitasking.
kddm	tskcom	I4	Task offset for DDM array.

Variable	Contained in Module	Type	Description
kddn	tskcom	I4	Task offset for DDN array.
kdec	tskcom	I4	Task offset for DEC array.
kdr(:)	mcnp_global	I4, ALLOCATABLE	ZAs from DRXS card.
kdrc	tskcom	I4	Task offset for DRC array.
kdup(:)	mcnp_global	I4, ALLOCATABLE	List of input cards for detecting duplicates.
kdxc	tskcom	I4	Task offset for DXC array.
kdxd	tskcom	I4	Task offset for DXD array.
keyp	mcnp_plot	character(len=8), dimension(nkeyp)	Command keywords of PLOT.
keys(nkeys)	mcplot_module	character(len=8), PARAMETER	Command keywords of MCPLOT.
kf8	fixcom	I4	Indicator of presence of F8 (pulse-height) tallies.
kfdd	tskcom	I4	Task offset for FDD array.
kfeb	tskcom	I4	Pointer to FEBL array.
kfl	fixcom	I4	Flag for cell or surface tally flagging.
kflx	tskcom	I4	Task offset for FLX array.
kfm(:)	mcnp_global	I4, ALLOCATABLE	Type of curve each surface makes in plot plane.
kfme	tskcom	I4	Task offset for FME array.
kfq	fixcom	I4	Facet number of macrobody surface.
kfso	tskcom	I4	Task offset for FSO array.
kgbn	tskcom	I4	Task offset for GBNK array.
kifg	tskcom	I4	Task offset for IAFG array.
kifl	tskcom	I4	Task offset for IFL array.
kise	tskcom	I4	Task offset for ISEF array.
kitm	mcnp_input	I4	Type of current item from input card.
kjaq	fixcom	I4	Flag for macrobody facets on source tape.
kjfl	tskcom	I4	Task offset for JFL array.
kjft	tskcom	I4	Task offset for JFT array.
kjpb	tskcom	I4	Task offset for JPTB array.
kkk	pblcom	I4	Third lattice index of particle location.
kktc	tskcom	I4	Task offset for KTC array.
klaj	tskcom	I4	Task offset for LAJ array.
klbl(43)	mcplot_module	I4	Key to cross-section plot reaction labels.
klcj	tskcom	I4	Task offset for LCAJ array.
klin = ‘ ‘	mcnp_data	character(len=80)	Input line currently being processed.
kls	mcnp_plot	I4	Phase of interrupted-line pattern.
klse	tskcom	I4	Task offset for LSE array.
kmaz	tskcom	I4	Task offset for maze array.
kmm(:)	mcnp_global	I4, ALLOCATABLE	Encoded IDs from M cards.
kmplot	ephcom	I4	Indicator of < ctrl-e > MCPLOT interrupt.
kmt(:,:)	mcnp_global	I4, ALLOCATABLE	Encoded ZAIDs from MT cards.
kndp	tskcom	I4	Task offset for NDPF array.
kndr	tskcom	I4	Task offset for NDR array.
knhs	tskcom	I4	Task offset for NHSD array.
knmc	tskcom	I4	Task offset for NMCP array.
knod	varcom	I4	Dump number.
knods	fixcom	I4	Last dump in the surface source write run.
knrm	fixcom	I4	Type of normalization of KCODE tallies.
kods	mcnp_data	character(len=8)	Name of the code that wrote surface source file.

Variable	Contained in Module	Type	Description
komout	ephcom	I4	Indicator that COMOUT has been created.
konrun	ephcom	I4	Continue-run flag.
koplot = 0	mcplot_module	I4	Flag for coplot.
kpac	tskcom	I4	Task offset for PAC array.
kpan	tskcom	I4	Task offset for PAN array.
kpc2	tskcom	I4	Task offset for IPAC2 array.
kpcc	tskcom	I4	Task offset for PCC array.
kpik	tskcom	I4	Task offset for PIK array.
kprod	ephcom	I4	Flag for production status.
kptb	tskcom	I4	Task offset for PTB array.
kpt(mipt)	fixcom	I4	Indicators of particle types in problem.
kpwb	tskcom	I4	Task offset for PWB array.
kqss	tskcom	I4	Latch for incrementing NQSW.
krflg	ephcom	I4	Flag to do event printing.
krho	tskcom	I4	Task offset for RHO array.
krq(7,nkcd)	mcnp_input	I4	Attributes of all types of input data cards.
krtc	tskcom	I4	Task offset for RTC array.
krtm	ephcom	I4	Flag for run-time monitor.
ksc(:)	mcnp_global	I4, ALLOCATABLE	Flags for parallel, possibly coincident, surfaces. 0=nonplanar, 2=PX, 3=PY, 4=PZ, N=P plane with orientation N. Parallel planes have same value.
ksd(:,:)	mcnp_global	I4, ALLOCATABLE	Source distribution information. See page E–27.
ksdef	varcom	I4	Flag for KCODE SDEF source.
ksf(39)	mcnp_data	character(len=3), parameter	List of all legal surface-type symbols.
kshs	tskcom	I4	Pointer to SHSD array.
ksm(:)	mcnp_global	I4, ALLOCATABLE	Macrobody surface flag = master surface of facet = -surface type of master surface.
ksr	ephcom	I4	Number of macrobody surface.
kst(:)	mcnp_global	I4, ALLOCATABLE	Surface-type numbers of all the surfaces.
kstt	tskcom	I4	Task offset for STT array.
ksu(:)	mcnp_global	I4, ALLOCATABLE	White (-2), reflecting (-1) or periodic (> 0) surface boundary.
ksum	tskcom	I4	Task offset for SUMP array.
ksww	tskcom	I4	Task offset for the SWWFA array.
ktal	tskcom	I4	Task offset for TAL array.
ktask	tskcom	I4	Index of the current task.
ktc(:,:)	mcnp_global	I4, ALLOCATABLE	Current indices of energy grids. See page E–28.
ktfile	ephcom	I4	Tally file open: none, RUNTPE, or MCTAL.
ktgp	tskcom	I4	Task offset for TGP array.
ktl(ntalmx,2)	mcnp_input	I4	Amount of storage needed for segment divisors.
ktls	fixcom	I4	Length of list scoring space.
ktmp	tskcom	I4	Task offset for TMP array.
ktp(:,:)	mcnp_global	I4, ALLOCATABLE	Particle types included in each tally.
ktr(:)	mcnp_global	I4, ALLOCATABLE	Cell transformation numbers from TRCL card.
ktskpt(ltskpt)	tskcom	I4	Equivalence to /ITSKPT/.
kufil(2,6)	fixcom	I4	Unit numbers and record lengths of user files.
kurv = 0	mcplot_module	I4	Type of plot: histogram, plinear, etc.
kwfa	tskcom	I4	Task offset for the WWFA array.

Variable	Contained in Module	Type	Description
kwns	tskcom	I4	Task offset for WNS array.
kxd(:)	mcnp_global	I4, ALLOCATABLE	Encoded dates of XSDIR entries.
kxs(:)	mcnp_global	I4, pointer	Indices of the cross-section tables on RUNTPE.
kxsmat = 1	mcplot_module	I4	Cross-section plot first material number in MAT array.
kxspar = 1	mcplot_module	I4	Cross-section plot source particle type number.
kxspkm	mcplot_module	I4	Cross-section plot pointer to ZAIDS in a material.
kxsplt = 0	mcplot_module	I4	Cross-section plot number of nuclides in mat.
kxspma = 0	mcplot_module	I4	Cross-section plot material number from input file.
kxspmt = 1	mcplot_module	I4	Cross-section plot reaction number.
kxsptp = 1	mcplot_module	I4	Cross-section plot data type.
kxspu(43)	mcplot_module	I4	Cross-section plot reaction label indices.
l8fixcm = 3	fixcom	I4, parameter	Size of integer*8 part of /FIXCM/.
laf(:,:)	mcnp_global	I4, pointer	Fill data for lattice elements.
laj(:)	mcnp_global	I4, ALLOCATABLE	Cells on the other sides of the surfaces in LJA.
lat(:,:)	mcnp_global	I4, ALLOCATABLE	Lattice type and VCL pointer for each cell.
lax = 0	mcplot_module	I4	Indicator of which axes are logarithmic.
lbb(:)	mcnp_global	I4, ALLOCATABLE	Size of records in bank backup file.
lca(:)	mcnp_global	I4, ALLOCATABLE	For each cell, a pointer into LJA and LCAJ.
lcaj(:)	mcnp_global	I4, ALLOCATABLE	For each surface in LJA, a pointer into the list of other-side cells in LAJ.
lchnk	ephcom	I4	Buffer size for passing PVM data.
lcl(:)	mcnp_global	I4, ALLOCATABLE	List of cells bounded by the current surface.
lcolor = 300	mcnp_plot	I4	Resolution of coloring for geometry plots.
legalc(nkeys)	mcplot_module	I4, PARAMETER	Legality of plot commands for COPLOT.
legalm(nkeys)	mcplot_module	I4, PARAMETER	Legality of plot command during runtime.
legalx(nkeys)	mcplot_module	I4, PARAMETER	Legality of plot commands in XS plots.
legend = 1	mcplot_module	I4	Indicator of type of legend specified.
lephcm	ephcom	I4, parameter	Length of integer part of /EPHCM/.
lev	pblcom	I4	Level of the current particle.
levp	tskcom	I4	Level of the next boundary.
levplt = -1	mcnp_plot	I4	Geometry plot level command level.
lfatl	ephcom	I4	Flag to run in spite of fatal errors.
lfcl(:)	mcnp_global	I4, ALLOCATABLE	Cells where fission is treated like capture.
lfixcm	fixcom	I4, parameter	Size of integer part of /FIXCM/.
lft(:,:)	mcnp_global	I4, ALLOCATABLE	Pointers to FT-card data.
lgc(mlgc+1)	tskcom	I4	Logical expression for the current point with respect to a particular cell.
likef = 0	mcnp_input	I4	Flag for ‘LIKE m BUT’ on cell card.
lit = 0	mcnp_input	I4	Length of ITDS array.
lja(:)	mcnp_global	I4, ALLOCATABLE	Logical geometrical definitions of all cells.
ljav(:)	mcnp_global	I4, ALLOCATABLE	Logical geometrical definition of current cell.
ljsv(:)	mcnp_global	I4, ALLOCATABLE	List of the surfaces of the current cell.
lme(:,:)	mcnp_global	I4, ALLOCATABLE	For each material, a list of the indices of the cross-section tables.
lmn(:)	mcnp_global	I4, ALLOCATABLE	Photonuclear isotope table indices.
lmrkp = 651	mcnp_params	I4, parameter	Minimum number of KCODE cycles to plot (mrkp).
lmt(:)	mcnp_global	I4, ALLOCATABLE	For each material, a list of the indices of the applicable S(a,b) tables.
lmtout = .false.	fmesh_mod	logical	Flag indicating the status of the mesh tally output file.

Variable	Contained in Module	Type	Description
lnstyl	mcplot_module	I4	Line type.
locct(:,:)	mcnp_global	I4, ALLOCATABLE	Cell-tally locators. See page E–33.
locdt(2,mxdt)	fixcom	I4	Detector-tally locators. See page E–32.
locph(:)	mcnp_global	I4, ALLOCATABLE	Pulse-height-tally locators.
locst(:,:)	mcnp_global	I4, ALLOCATABLE	Surface-tally locators. See page E–33.
lods	mcnp_data	character(len=8)	LODDAT of code that wrote surface source file.
lost(2)	varcom	I4	Controls for handling lost particles.
lpac = 0	fixcom	I4	Offset for PAC array.
lpan = 0	fixcom	I4	Offset for PAN array.
lpblcm	pblcom	I4, parameter	Length of /PBLCM/ description.
lpert = 0	mcplot_module	I4	Perturbation number for MCPLOT.
lput = 0	mcplot_module	I4	Flag for title below plot.
lpwb = 0	fixcom	I4	Offset for PWB array.
lrt = 0	mcnp_input	I4	Length of RTP array.
lsat(:)	mcnp_global	I4, ALLOCATABLE	For each segmented tally, a pointer into ATSA.
lsb	tskcom	I4	Latch for the count of bank overflows.
lsc(:)	mcnp_global	I4, ALLOCATABLE	For each surface, a pointer into SCF.
lse(:)	mcnp_global	I4, ALLOCATABLE	Cells where source particles have appeared.
lsg(:)	mcnp_global	I4, ALLOCATABLE	Kind of line to plot for each segment of curve.
lspeed	ephcom	I4	Baud rate of the plotting terminal display.
ltasks	ephcom	I4	Number of PVM tasks =|JTASKS|.
ltd = 0	mcnp_input	I4	Length of TDS array.
ltskcm	tskcom	I4, parameter	Size of integer part of /TSKCM/.
ltype	mcnp_plot	I4	Line type.
lvarcm	varcom	I4, parameter	Size of integer part of /VARCM/.
lvarsw	varcom	I4, parameter	Number of swept variable common.
lxd(:,:)	mcnp_global	I4, ALLOCATABLE	Encoded ZAID extension from M cards.
lxs	fixcom	I4	Length of XSS array.
m10c	mcnp_input	I4	General purpose variable for input phase.
m11c	mcnp_input	I4	General purpose variable for input phase.
m1c	mcnp_input	I4	General purpose variable for input phase.
m2c	mcnp_input	I4	General purpose variable for input phase.
m3c	mcnp_input	I4	General purpose variable for input phase.
m4c	mcnp_input	I4	General purpose variable for input phase.
m5c	mcnp_input	I4	General purpose variable for input phase.
m6c	mcnp_input	I4	General purpose variable for input phase.
m7c	mcnp_input	I4	General purpose variable for input phase.
m8c	mcnp_input	I4	General purpose variable for input phase.
m9c	mcnp_input	I4	General purpose variable for input phase.
mai	fixcom	I4	Index number of reference mesh in mesh-based weight window generator.
mat(:)	mcnp_global	I4, pointer	Material numbers of the cells.
maxf = 16	mcnp_params	I4, parameter	Number of sampleable source variables.
maxi = 34	mcnp_params	I4, parameter	Number of electron scattering angle groups.
maxv = 20	mcnp_params	I4, parameter	Number of SDEF source variables.
maxw = 4	mcnp_params	I4, parameter	Number of SSW source variables.
maze(:)	mcnp_global	I4, ALLOCATABLE	Universe/lattice map values. See page E–42.
mazf(3)	ephcom	I4	Total source, entering, collisions in maze.

Variable	Contained in Module	Type	Description
mazp(:,:)	mcnp_global	I4, pointer	Universe/lattice map addresses. See page E–42.
mazu(:)	mcnp_global	I4, pointer	Universe/lattice map pointers. See page E–42.
mbb	tskcom	I4	Size of the part of bank currently in memory.
mbd(:)	mcnp_global	I4, ALLOCATABLE	Flags for cells for which DBMIN is inappropriate.
mbi(:)	mcnp_global	I4, ALLOCATABLE	Which materials have bremsstrahlung biasing.
mbng = 51	mcnp_params	I4, parameter	Number of possible photon/electron ratio values.
mbnk	fixcom	I4	Size of the bank in words per task.
mcal	fixcom	I4	Type of multigroup problem.
mclb = 19	mcnp_plot	I4, parameter	Number of LABEL command keywords.
mclevs = 20	mcplot_module	I4, parameter	Maximum number of contour levels allowed.
mcnp_opt_mpi	mcnp_params	logical, parameter	Parallel message-passing with MPI.
mcnp_opt_omp	mcnp_params	logical, parameter	Parallel threads using OpenMP (omp).
mcnp_opt_pvm	mcnp_params	logical, parameter	Parallel message-passing with PVM.
mcoh = 55	mcnp_params	I4, parameter	Number of WCO coherent form factors.
mcolor	ephcom	I4	Number of colors available for geometry plots.
mct	fixcom	I4	Flag to write MCTAL file at end of the run.
mctal	mcnp_iofiles	character(len=8)	Tally output file name.
mdc	ephcom	I4	Flag indicating a dump is due to be written.
mephcm	ephcom	I4	Marker variable at end of /EPHCM/.
meshpl = 1	mcnp_plot	I4	Plot line mode 0/1/2/3=none/cells/ww mesh/both.
meshtal	mcnp_iofiles	character(len=8)	Mesh tally output file.
mfiss(22)	mcnp_data	I4, parameter	Fission ZAIDS for fission Q-values.
mfl(:,:)	mcnp_global	I4, pointer	Fill data for each cell.
mfm(:)	mcnp_global	I4, ALLOCATABLE	FM-card material numbers.
mgegbt(mipt)	fixcom	I4	Index of a multigroup table for each particle.
mgm(mipt+1)	fixcom	I4	Cumulative number of multigroup groups.
mgww(mipt+1)	fixcom	I4	Cumulative sum of NGWW.
minc = 21	mcnp_params	I4, parameter	Number of VIC incoherent form factors.
mink = 200	mcnp_params	I4, parameter	Length of INK array.
mipt = 3	mcnp_params	I4, parameter	Number of kinds of particles the code can run.
mipts = 0	mcnp_input	I4	Source particle type.
mix	fixcom	I4	Number of entries in KMM and FME.
mjsf = 9	mcnp_params	I4, parameter	Length of JSF array.
mjss	fixcom	I4	Space needed for surfaces and cells from SSW.
mkc	tskcom	I4	Index of the current material.
mkcp = 0	mcnp_input	I4	Size of array KCP.
mkft = 9	mcnp_params	I4, parameter	Number of kinds of FT card special treatments.
mkpl = 35	mcnp_params	I4, parameter	Number of entries in RKPL for KCODE tally plots.
mktc = 26	mcnp_params	I4, parameter	Number of kinds of tally cards.
mlaf = 0	mcnp_input	I4	Space required for LAF.
mlaj	fixcom	I4	Length of LAJ array.
mlam = 5001	mcnp_landau	I4, parameter	Length of Landau electron scattering array eqlm.
mlanc = 1591	mcnp_landau	I4, parameter	Electron Landau lambda cutoff values.
mlgc = 1000	mcnp_params	I4, parameter	Size of logical arrays for complicated cells.
mlja	fixcom	I4	Length of LJA array.
mmkdb	ephcom	I4	Print history info flag for EXPIRE.
mnk	ephcom	I4	Flag to indicate maximum printing is wanted.
mnnm	fixcom	I4	Maximum number of nuclides on M card.

Variable	Contained in Module	Type	Description
monod	varcom	I4	Counter for image print (can be removed).
mopts = 7	mcnp_input	I4, parameter	Number of M card options (gas, estep, etc.).
mpan	tskcom	I4	Index in PAN of collision material/nuclide.
mpb = 5	pblcom	I4, parameter	Depth of the /PBLCM/.
mpb9cm(mpb)	pblcom	I4	Marker variable in /PBLCM/.
mpblcm	pblcom	I4	Marker variable at end of /PBLCM/.
mpc	ephcom	I4	Flag indicating that printing is due to be done.
mplm = 100	mcnp_plot	I4, parameter	Number of material color shadings in plot.
mpng = 21	mcnp_params	I4, parameter	Number of angle groups in ECH.
mrkp	fixcom	I4	Number of KCODE cycles kept for plotting.
mrl	fixcom	I4	Number of source points in FSO.
mrm	ephcom	I4	Flag indicating that plotting is due to be done.
mscal = 1	mcplot_module	I4	Indicator of type of scales wanted on plot.
msd	fixcom	I4	Number of source distributions.
mseb = 301	mcnp_params	I4, parameter	Maximum number of equiprobable source bins.
mspare = 3	pblcom	I4, parameter	Number of spare entries in /PBLCM/.
msrk	fixcom	I4	Maximum number of source points in FSO.
mssc = 0	mcnp_input	I4	Length of source comments array JSCN.
mstp = 4	mcnp_params	I4, parameter	Coarsening factor for electron energy grids.
msub(ndef)	mcnp_iofiles	character(len=8)	Default file names.
mtal = 0	mcplot_module	I4	Index of the current tally.
mtasks	fixcom	I4	Multi-threading parallel offset, usually MTASKS+1.
mtop = 89	mcnp_params	I4, parameter	Number of bremsstrahlung energy groups + 1.
mtp	pblcom	I4	Reaction MT from previous collision.
mtskcm	tskcom	I4	Marker after integer part of /TSKCM/.
mtskpt	tskcom	I4	Marker after /ITSKPT/.
munit	mcnp_plot	I4	Postscript file unit number.
mvarcm	varcom	I4	Marker variable at end of /VARCM/.
mwng = (mtop+1)/2	mcnp_params	I4, parameter	Number of photon energy groups in ECH.
mww(mipt+1)	fixcom	I4	Cumulative sum of NWW.
mxa	fixcom	I4	Number of cells in the problem.
mxafs	fixcom	I4	Number of cells plus pseudocells for FS cards.
mxdt = 20	mcnp_params	I4, parameter	Maximum number of detectors.
mxdx = 10	mcnp_params	I4, parameter	Maximum number of DXTRAN spheres.
mxe	fixcom	I4	Number of cross-section tables in the problem.
mxf	fixcom	I4	Total number of tally bins.
mxfp	fixcom	I4	Number of tally bins without perturbations.
mxit = 0	mcnp_input	I4	Longest input geometry definition for any cell.
mxj	fixcom	I4	Number of surfaces in the problem.
mxlv = 10	mcnp_params	I4, parameter	Maximum number of levels allowed.
mxss = 6	mcnp_params	I4, parameter	Spare dimension of surface source arrays.
mxt	fixcom	I4	Number of cell-temperature time bins.
mxtr	fixcom	I4	Number of surface transformations.
mxxs	fixcom	I4	Length of SPF and WNS.
mynum	ephcom	I4	PVM index (=0 for master task).
naw = 0	mcnp_input	I4	Number of atomic weights from AWTAB card.
nbal(:)	mcnp_global	I4, ALLOCATABLE	Number of histories processed by each task.
nbands = 4	mcnp_plot	I4, parameter,private	Number of gradient color bands.

Variable	Contained in Module	Type	Description
nbhwm	varcom	I4	Largest number of particles ever in the bank.
nbhwtc	tskcom	I4	Task copy of NBHWM.
nbmx = 640	mcnp_params	I4, parameter	Number of particles IBNK has room for.
nbnk	tskcom	I4	Number of particles in the bank.
nbov	varcom	I4	Count of bank overflows.
nbt(mipt)	varcom	I4	Total numbers particles banked.
nbttc(mipt)	tskcom	I4	Task copy of NBT.
ncel	mcnp_plot	I4	Number of cells bounded by the current surface.
nch(mipt)	tskcom	I4	Counts of neutron and photon collisions or electron substeps.
ncl(:)	mcnp_global	I4, pointer	Problem numbers of the cells.
nclev	mcplot_module	I4	Number of contour levels.
ncolor=64	mcnp_plot	I4, parameter	Number of colors for plotting.
ncomp = 0	mcnp_input	I4	Count of # characters on cell cards.
ncp	pblcom	I4	Count of collisions per track.
ncparf = 0	mcnp_input	I4	Number of cell parameter cards on cell cards.
ncpar(mipt,nkcd)	mcnp_input	I4	Largest cell parameter n, -1 if none.
ncrn	mcnp_input	I4	Number of corners in the current cell.
ncrs	mcnp_plot	I4	Length of LSG and CRS arrays.
ncs(:)	mcnp_global	I4, ALLOCATABLE	Number of curves where surface meets plot plane.
nctext	mcnp_plot	I4	GKS graphics color index.
nde	ephcom	I4	Value of execute-message item DBUG n.
ndef = 32	mcnp_params	I4, parameter	Number of file names.
ndeitm(:)	fmesh_mod	I4, ALLOCATABLE	Number of energy values for the mesh tally response function.
ndet(mipt)	fixcom	I4	Numbers of neutron and photon detectors.
ndfitm(:)	fmesh_mod	I4, ALLOCATABLE	Number of values for the mesh tally response function.
ndmp	varcom	I4	Maximum number of dumps on RUNTPE.
ndnd	fixcom	I4	Number of detectors in the problem.
ndpf(:,:)	mcnp_global	I4, ALLOCATABLE	Accounts of detector scores that failed.
ndp(ntalmx)	mcnp_input	I4	Tally numbers appearing with PD on cell cards.
ndr(:)	mcnp_global	I4, ALLOCATABLE	List of discrete-reaction rejections.
ndtt	fixcom	I4	Total number of detectors in the problem.
ndup(3) = (/0,0,0/)	mcnp_input	I4	Number of cards in each input data block.
ndx(mipt)	fixcom	I4	Numbers of neutron and photon DXTRAN spheres.
nee	fixcom	I4	Number of energies in EEE (0 if no electrons).
nephcm	ephcom	I4, parameter	Length of real part of /EPHCM/.
nerr	varcom	I4	Count of lost particles.
nesm	varcom	I4	Number of tracks that escape the superimposed mesh in mesh-based weight window generation.
netb(2)	varcom	I4	Count of times energy > EMX.
nets(2,mipt)	fixcom	I4	Number of energies & times for e/t split, by particle type.
nfer	varcom	I4	Count of fatal errors found by IMCN or XACT.
nfixcm	fixcom	I4, parameter	Size of floating-point part of /FIXCM/.
nfmsh = 15	fmesh_mod	I4, parameter	Number of keywords on the FMESH card.
nfree = 1	mcplot_module	I4	Number of free variables in current plot.
ngmfl(:)	mcnp_global	I4, ALLOCATABLE	Gamma production flag for material IEX for XS plot.
ngp	tskcom	I4	Electron energy group.
ngww(mipt)	fixcom	I4	Number of weight-window-generator energy bins.

Variable	Contained in Module	Type	Description
nhb	fixcom	I4	Number of history bin computed from DBCN(16).
nhsd(:,:)	mcnp_global	I4, ALLOCATABLE	Number in history score distribution that counts nonzero scores for statistical analysis.
nhtfl(:)	mcnp_global	I4, ALLOCATABLE	Heating number flag for material IEX for XS plot.
n_idum = 50	mcnp_debug	I4, parameter	Length of IDUM.
nii	mcnp_input	I4	Number of interpolated values to make; -1 for J.
nilr(mxss)	fixcom	I4	Number of cells on SSR card.
nilw	fixcom	I4	Number of cells on SSW card.
nips	fixcom	I4	Source particle type.
niss	fixcom	I4	Number of histories in input surface source.
nitm	mcnp_input	I4	Length of current item from input card.
niwr = 0	mcnp_input	I4	Number of cells in RSSA file.
njsr(mxss)	fixcom	I4	Number of surfaces in JASR.
njss	fixcom	I4	Number of surfaces in JSS.
njsw = 0	mcnp_input	I4	Number of surfaces in JASW.
njsx(mxss)	fixcom	I4	Number of surfaces in ISS.
nkcd = 101	mcnp_input	I4, parameter	Number of different types of input cards.
nkeyp = 35	mcnp_plot	I4, parameter	Number of PLOT commands.
nkeys = 58	mcplot_module	I4, PARAMETER	Number of MCPLOT commands.
nkrp	ephcom	I4	Latch for warning in CALCPS.
nkxs	fixcom	I4	Count of cross-section tables written on RUNTPE.
nlaj	tskcom	I4	Number of other-side cells in LAJ.
nlat	fixcom	I4	Number of lattice universes in the problem.
nlb	mcnp_plot	I4	Number of surface labels on the plot.
nlev	fixcom	I4	Number of levels in the problem.
nlja	fixcom	I4	Number of entries in LJA.
nlocks = 10	mcnp_params	I4, parameter	Number of OMP locks.
nlse	tskcom	I4	Number of cells in the LSE list.
nlt	tskcom	I4	Number of entries in DTI.
nltext	mcnp_plot	I4	GKS graphics color index.
nlv(:)	mcnp_global	I4, ALLOCATABLE	Number of levels in each cell.
nmat	fixcom	I4	Number of materials in the problem.
nmaz	fixcom	I4	Length of maze array (0 in 1st pass).
nmc	tskcom	I4	Counter for weight window generator tracking.
nmco	pblcom	I4	Stores value of NMC as it is updated.
nmcp(:,:)	mcnp_global	I4, ALLOCATABLE	Track record array for weight window generator.
nmfm = 0	mcnp_input	I4	2*number of materials on FM cards.
nmip	fixcom	I4	Number of particle types for lattice/universe maze.
nmkey = 11	mcnp_params	I4, parameter	Number of MESH keywords.
nmrkp = 6500	mcnp_params	I4, parameter	Maximum number of KCODE cycles to plot (mrkp).
nmt(:)	mcnp_global	I4, ALLOCATABLE	Names of the materials.
nmxf	fixcom	I4	Number of tally blocks = 3 or = 5 if DBCN(15) set to give VOV in all bins.
nmzu	fixcom	I4	Length of MAZU array.
nnpos	fixcom	I4	Index of first position variable to be sampled.
nocoh	fixcom	I4	Flag to inhibit coherent photon scattering.
node	pblcom	I4	Number of nodes in track from source to here.
nodop	fixcom	I4	Flag to inhibit Doppler photon scattering.
noerbr = 0	mcplot_module	I4	Flag for no error bars on plots.

Variable	Contained in Module	Type	Description
nomore	ephcom	I4	Flag for exhausted surface-source file.
nonorm = 0	mcplot_module	I4	Flag for no tally bin normalization.
nord	fixcom	I4	Number of source variables to be sampled.
notal	varcom	I4	Flag to not print tally bins to outp file from the talnp card: see also ntprt (default=0 to print all tallies).
notrn	varcom	I4	Flag to only calculate direct source to point detector tallies (no particles are transported).
novol = 0	mcnp_input	I4	Flag to inhibit volume calculation.
novr = 5	mcnp_params	I4, parameter	Number of main code sections.
np1	fixcom	I4	Number of histories in surface source write run.
npa	pblcom	I4	Number of tracks in the same bank location.
npages	mcnp_plot	I4	Number of postscript file pages.
npb	tskcom	I4	Number of saved particles in GPB9CM.
npblcm	pblcom	I4, parameter	Size of floating-point part of /PBLCM/.
npc(20)	varcom	I4	NPS for tally fluctuation charts. See page E–35.
npd	varcom	I4	NPS step in tally fluctuation chart.
npert	fixcom	I4	Number of perturbations.
npikmt	fixcom	I4	Number of PIKMT entries.
npkey = 6	mcnp_params	I4, parameter	Number of PERT keywords.
nplb	mcnp_plot	I4	Length of PLB array.
npn	fixcom	I4	Length of adjustable dimension of PAN.
npnm	varcom	I4	Count of times neutron-reaction MT not found.
npp	varcom	I4	Number of histories to run, from NPS card.
nppm	varcom	I4	Count of times photon-production MT not found.
npq(:)	mcnp_global	I4, ALLOCATABLE	Number of components in each material.
nps	varcom	I4	Count of source particles started.
npsmg	varcom	I4	Number of source particles that contribute to image grid (2nd nps card entry).
npsout	varcom	I4	NPS when output was last done.
npsr	varcom	I4	History number last read from surface source.
npsrtc	tskcom	I4	Task copy of nspr.
npstc	tskcom	I4	Task copy of nps.
npsw(:)	mcnp_global	I4, ALLOCATABLE	For each surface source surface, the last history in which a track crossed it.
npt(2)	mcplot_module	I4	Number of points to plot in each direction.
nptb(:)	mcnp_global	I4, ALLOCATABLE	Pointers to DPTB and RPTB arrays. See page E–43.
nptr = 13	mcnp_params	I4, parameter	Number of PTRAC keywords (HPTR).
npum	varcom	I4	Flag for Photonuclear production failure.
nqp(mipt+1)	mcnp_input	I4	Flags for particle-type indicators on card.
nqss	varcom	I4	Number of histories read from surface source.
nqsw	varcom	I4	Number of histories written to surface source.
nqw	mcnp_input	I4	Particle type of input card. See JPTAL array, page E–31.
nrc	ephcom	I4	Count of restarts in the run.
nrcd	fixcom	I4	Number of values in a surface-source record.
n_rdum = 50	mcnp_debug	I4, parameter	Length of RDUM.
nrnh(3)	varcom	I4	Information about number of random numbers used.
nrnhtc(3)	tskcom	I4	Task copy of NRNH.
nrrs	varcom	I4	Number of tracks read from surface source.
nrss	fixcom	I4	Number of tracks on input surface source file.

Variable	Contained in Module	Type	Description
nrsw	varcom	I4	Number of tracks written to surface source.
nsa	varcom	I4	Source particles yet to be done in this cycle.
nsa0	varcom	I4	Saved NSA value to rerun lost history.
nsb(:)	mcnp_global	I4, ALLOCATABLE	Substeps per step for each material.
nsc = 0	mcnp_input	I4	Number of surface coefficients in SCF.
nsfm(:)	mcnp_global	I4, pointer	Problem names of surfaces.
nshades = nbands*16	mcnp_plot	I4, parameter	Number of gradient colors.
nsjv	mcnp_input	I4	Length of cell definition in LJAV.
nskk	varcom	I4	Number of histories in first IKZ KCODE cycles.
nsl(:,:)	mcnp_global	I4, ALLOCATABLE	Summary information for surface source file.
nslr(:,:)	mcnp_global	I4, ALLOCATABLE	Summary information from surface source file.
nsom	varcom	I4	Number of tracks that start outside superimposed mesh in mesh-based weight window generation.
nsp = 602	mcnp_params	I4, parameter	Number of points in history score distribution grid.
nsp12 = nsp+12	mcnp_params	I4, parameter	NSP+12.
nsph	fixcom	I4	Flag for spherical output surface source.
nspt = nsp+ntp+7	mcnp_params	I4, parameter	NSP+NTP+7.
nsr	fixcom	I4	Source type.
nsrc	fixcom	I4	Number of entries on SRC card.
nsrck	fixcom	I4	Nominal size of the KCODE source.
nss	varcom	I4	Count of source points stored for the next cycle.
nss0	varcom	I4	Saved NSS value to rerun lost history.
nssi(10)	varcom	I4	Numbers of rejected surface source tracks.
nst	ephcom	I4	Reasons why the run is terminating.
nstp	fixcom	I4	Value of MSTP for current electron library.
nsv	mcnp_input	I4	Number of surfaces in LJSV.
ntal	fixcom	I4	Number of tallies in the problem.
ntalmx = 100	mcnp_input	I4, parameter	Maximum number of tallies.
ntasks	ephcom	I4	Number of threads.
ntbb(:,:)	mcnp_global	I4, ALLOCATABLE	Counts of scores beyond the last bin.
ntc	varcom	I4	Control variable for time check.
ntc1	varcom	I4	Second control variable for time check.
nter	tskcom	I4	Type of termination of the track.
ntii	tskcom	I4	Indicator of multiple time interrupts.
ntl(0:ntalmx)	mcnp_input	I4	Tally numbers from tally input cards.
ntop	fixcom	I4	MTOP value for current electron library.
ntp = 201	mcnp_params	I4, parameter	Number of tail points in history score distribution statistical analysis table.
ntprt(100)	varcom	I4	List of tally numbers on talnp card that will not have bin values printed to the outp file.
ntskcm	tskcom	I4, parameter	Size of floating-point part of /TSKCM/.
ntss	varcom	I4	Number of surface source tracks accepted.
ntx	tskcom	I4	Number of calls of TALLYX in user bins loop.
nty(:)	mcnp_global	I4, ALLOCATABLE	Type of each cross-section table.
ntyn	tskcom	I4	Type of reaction in current collision.
numb	fixcom	I4	Flag for biasing bremsstrahlung production in each step.
num_bins(:)	fmesh_mod	I4, ALLOCATABLE	Number of mesh tally bins scored per history
nvarcm	varcom	I4, parameter	Size of floating-point part of /VARCM/.
nvarsw	varcom	I4, parameter	Number of swept variable common.

Variable	Contained in Module	Type	Description
nvec	fixcom	I4	Number of vectors on VECT card.
nvs(maxv)	mcnp_data	I4	Number of values required for each source variable.
nwang	fixcom	I4	Weight window mesh file type and adjoint current flag.
nwc	mcnp_input	I4	Count of items on current input card.
nwer	varcom	I4	Count of warning messages printed.
nwgeoa	fixcom	I4	For weight window generation on: 1/2/3=a superimposed rectangular mesh/a superimposed cylindrical mesh/cells.
nwgeom	fixcom	I4	For weight windows from the WWINP file for: 1/2/3=rectangular mesh/cylindrical mesh/cells.
nwgm	fixcom	I4	Number of coarse mesh cells in weight window mesh +9
nwgma	fixcom	I4	Number of coarse mesh cells in superimposed grid for mesh-based weight window generation.
nwng	fixcom	I4	Current number of ratios for bremsstrahlung angular distributions.
nwsb	varcom	I4	Count of source weights below cutoff.
nwse	varcom	I4	Count of source energies below cutoff.
nwsg(3)	varcom	I4	Count of source weights above weight window.
nwst	varcom	I4	Count of source times greater than cutoff.
nwwm	fixcom	I4	Number of fine mesh cells in weight windows mesh.
nwwma	fixcom	I4	Number of fine mesh cells in superimposed grid for mesh-based weight window generation.
nww(mipt)	fixcom	I4	Number of weight-window energy bins.
nwws(2,99)	varcom	I4	Like NWSG and NWSL but binned.
nxnorm	mcnp_plot	I4	Postscript file plot normalization.
nxnx	fixcom	I4	Number of DXTRAN spheres in the problem.
nxp	mcnp_plot	I4	Number of intersections in CRS.
nxs(:,:)	mcnp_global	I4, ALLOCATABLE	Blocks of descriptors of cross-section tables.
nxsc = 0	mcnp_input	I4	Number of XSn cards.
nynorm	mcnp_plot	I4	Postscript file plot normalization.
nziy(8,mxdx,mipt)	varcom	I4	DXTRANs lost to zero importance.
nziytc(8,mxdx,mipt)	tskcom	I4	Task copy of NZIY.
one = 1.0d+0	mcnp_params	R8, parameter	Floating-point constant 1. for arguments.
origin(3) = (/0.,0.,0. /)	mcnp_plot	real	Origin for plotting.
orsav(3)	mcnp_plot	real	Saved origin.
osum2(3,3)	varcom	R8	k_{eff} covariances, cumulative. See page E–40.
osum(3)	varcom	R8	k_{eff}, cumulative. See page E–40.
outp = ‘ ‘	mcnp_iofiles	character(len=8)	Output data file name.
pac(:,:,:)	mcnp_global	R8, ALLOCATABLE	Activity in each cell. See page E–37.
pan(:,:,:)	mcnp_global	R8, ALLOCATABLE	Activity of each nuclide. See page E–39.
pax(6,21,mipt)	varcom	R8	Ledger of creation and loss. See page E–36.
paxtc(6,21,mipt)	tskcom	R8	Task copy of PAX.
pbr(:)	mcnp_global	R8, ALLOCATABLE	Bremsstrahlung production cross sections.
pbt(:)	mcnp_global	R8, ALLOCATABLE	Thick-target bremsstrahlung probabilities.
pcc(:,:)	mcnp_global	R8, ALLOCATABLE	Neutron-induced photons, by cell. See page E–39.
pfp	tskcom	R8	Probability of electron scatter.
pie = 3.1415926535898d+0	mcnp_params	R8, parameter	Pi.
pik(:)	mcnp_global	R8, ALLOCATABLE	Entries from PIKMT card.
pim(10:100)	mcnp_landau	R8	Landau electron mean ionization potentials.
pimph(9,4)	mcnp_landau	R8	Landau electron mean ionization potentials.

Variable	Contained in Module	Type	Description
pkn(:)	mcnp_global	R8, ALLOCATABLE	Knock-on production cross sections.
planck = 4.135732d-13	mcnp_params	R8, parameter	Planck constant.
plb(:)	mcnp_global	R8, ALLOCATABLE	Locations and widths of surface labels.
ple	tskcom	R8	Macroscopic cross section of current cell.
plim(4)	mcplot_module	real	Limits of the plot.
plmx(4,4)	mcnp_plot	R8	Plot matrix.
plotm = ‘ ‘	mcnp_iofiles	character(len=8)	Plot file name.
pmf	tskcom	R8	Distance to next collision.
pmg(:)	mcnp_global	R8, ALLOCATABLE	Table for biased adjoint sampling.
pnt(:)	mcnp_global	R8, ALLOCATABLE	Lowest photonuclear threshold for materials.
pptme(4)	varcom	R8	Wall clock times for multiprocessing.
prb(:)	mcnp_global	R8, ALLOCATABLE	Probabilities for equiprobable-bin iteration.
prn	varcom	R8	Print control from PRDMP card.
probid = ‘ ‘	mcnp_data	character(len=19)	Problem identification string.
probs	mcnp_data	character(len=19)	PROBID of the surface source write run.
pru(:)	mcnp_global	R8, ALLOCATABLE	Part of the knock-on angular distribution.
psc	tskcom	R8	Probability density for scattering toward a detector or DXTRAN sphere.
psize(4)	mcnp_plot	real	Postscript file scale factor.
ptb(:,:)	mcnp_global	R8, ALLOCATABLE	Perturbation coefficients. See page E–43.
ptbtc	tskcom	R8	Total perturbed tally score. See page E–44.
ptr(:)	mcnp_global	R8, ALLOCATABLE	PTRAC input parameters.
ptrac	mcnp_iofiles	character(len=8)	Particle track file name.
pts(:)	mcnp_global	R8, ALLOCATABLE	PTRAC track descriptions.
pwb(:,:,:)	mcnp_global	R8, ALLOCATABLE	Weight-balance tables. See page E–37.
pxr(:)	mcnp_global	R8, ALLOCATABLE	X-ray production cross sections.
pxx(4,4)	mcnp_plot	R8	Plot matrix transformed for all levels.
qav(:)	mcnp_global	R8, ALLOCATABLE	Ionization loss straggling coefficients.
qax(:,:)	mcnp_global	R8, ALLOCATABLE	Exponential transform parameters for each cell.
qcn(:)	mcnp_global	R8, ALLOCATABLE	Ionization loss straggling coefficients.
qfiss(23)	mcnp_data	R8, parameter	Fission Q-values.
qmx(:,:,:,:)	mcnp_global	R8, ALLOCATABLE	Curves where surfaces intersect the plot plane.
qpl	tskcom	R8	Adjusted macroscopic cross section.
ralfp(2)	pblcom	R8	Eigenvalue by 2nd order perturbation method.
RDEF = 4	mcnp_params	I4, parameter	Default real kind.
rdum(1:n_rdum) = 0	mcnp_debug	R8	Data from RDUM card.
res = 1./1500.	mcnp_plot	real, parameter	Plot resolution.
rfq(15)	mcnp_data	character(len=58)	Partial formats for termination messages.
rho(:)	mcnp_global	R8, ALLOCATABLE	Atom densities of the cells.
rim	fixcom	R8	Compression limit for weight windows.
ritm	mcnp_input	R8	Real form of current item from input card.
rka(mbng)	fixcom	R8	Photon/electron energy ratios for angular distributions.
rkk	varcom	R8	Collision estimate of k_{eff}.
rknd = selected_real_kind(6, 37)	mcnp_params	I4, parameter	Real kind.
rkpl(:,:)	mcnp_global	R8, ALLOCATABLE	KCODE quantities for plotting. See page E–41.
rktc(mtop)	mcnp_data	R8, parameter	Bremsstrahlung photon/electron energy ratios for current electron library.

Variable	Contained in Module	Type	Description
rkt(mtop)	fixcom	R8	Bremsstrahlung photon/electron energy ratios for current electron library.
rlt(4,2)	varcom	R8	Removal lifetimes, current cycle. See page E–40.
rlttc(4,2)	tskcom	R8	Task copy of RLT.
rnb(5)	tskcom	R8	Saved random numbers for ENDF law 67 neutrons.
rng(:)	mcnp_global	R8, ALLOCATABLE	Electron ranges.
rnk	pblcom	R8	RNR at point where new track was created.
rnok	fixcom	R8	Knock-on electron production bias.
rnr	varcom	R8	Count of pseudorandom numbers generated.
rnrtc	tskcom	R8	Task copy of RNR.
rptb(:)	mcnp_global	R8, ALLOCATABLE	PERT card keyword entries. See page E–44.
rr0	tskcom	R8	Interpolation fraction for ENDF law 67 neutrons.
rrmc(50)	varcom	R8	Data from RRMC input card.
rscrn(:,:)	mcnp_global	R8, ALLOCATABLE	R and S coordinates of cell corners.
rsint(:,:)	mcnp_global	R8, ALLOCATABLE	R and S coordinates of surface intersections.
rssa	mcnp_iofiles	character(len=8)	Surface source read file name.
rssp	varcom	R8	Radius of spherical surface source.
rsum2(3,3)	varcom	R8	Removal lifetime covariances, cumulative. See page E–41.
rsum(3)	varcom	R8	Removal lifetimes, cumulative. See page E–41.
rtc(:,:)	mcnp_global	R8, ALLOCATABLE	Current interpolated cross sections. See page E–28.
rtp(:)	mcnp_global	R8, ALLOCATABLE	Tally-card data. See page E–35.
runtpe	mcnp_iofiles	character(len=8)	Restart data file name.
scalf(2,3)	mcplot_module	R8	Scale factors for plot data.
scf(:)	mcnp_global	R8, ALLOCATABLE	Surface coefficients for all surfaces.
scfq(:,:)	mcnp_global	R8, ALLOCATABLE	Q-form of surface coefficients.
sch = .03	mcnp_plot	real	Scale factor for geometry plots.
sclabl(4) = (/1.,0.,1.,0./)	mcnp_plot	real	LABEL parameters.
scr(:)	mcnp_global	R8, ALLOCATABLE	Scratch storage for GMGWW.
sfb(:)	mcnp_global	R8, ALLOCATABLE	Probabilities of the source input groups.
sff(3,maxv)	tskcom	R8	Current values of source variables.
shades	mcnp_plot	type(color), dimension(0:nshades)	Colors used for gradients. Zeroth shade is white.
shsd(:,:)	mcnp_global	R8, ALLOCATABLE	Score in the history score distribution for statistical analysis..
siga	tskcom	R8	Capture cross section.
slite = 299.7925d+0	mcnp_params	R8,parameter	Speed of light.
smg(:)	mcnp_global	R8, ALLOCATABLE	Table for biased adjoint sampling.
smul(7)	varcom	R8	Tally of neutron multiplication.
smultc(7)	tskcom	R8	Task copy of SMUL.
snit	varcom	R8	Surface source splitting or RR factor.
spare(mspare)	pblcom	R8	Spare banked array for user modifications.
spf(:,:)	mcnp_global	R8, ALLOCATABLE	Source probability distributions. See page E–27.
sqq(:,:)	mcnp_global	R8, ALLOCATABLE	Coefficients of the built-in source functions.
srctp	mcnp_iofiles	character(len=8)	Source file name (in/out).
srv(3,maxv)	fixcom	R8	Explicit or default values of source variables.
ssb(11)	ephcom	R8	Surface source input buffer.
sso(:)	mcnp_global	R8, ALLOCATABLE	Equiprobable bins for source distributions.
ssr	tskcom	R8	Neutron speed relative to target nucleus.
stp	tskcom	R8	Electron stopping power.

Variable	Contained in Module	Type	Description
stt(:,:)	mcnp_global	R8, ALLOCATABLE	Big and small tally scores for statistical analysis.
sumk(3)	varcom	R8	Sums of KCODE fission weight. See page E–41.
sumktc(3)	tskcom	R8	Task copy of SUMK.
sump(:)	mcnp_global	R8, ALLOCATABLE	Perturbed track length k_{eff}. See page E–41.
swtm	varcom	R8	Minimum weight of source particles.
swtx = 0.	mcnp_input	R8	Minimum source weight for obsolete sources.
swwfa(:)	mcnp_global	R8, ALLOCATABLE	Weight window generator scoring weight array.
tal(:)	mcnp_global	R8, ALLOCATABLE	Tally scores accumulation. See page E–30.
talb(8,2)	mcnp_data	R8, parameter	Bins for detector and DXTRAN diagnostics.
tbt(:)	mcnp_global	R8, ALLOCATABLE	Temperatures of the cross-section tables.
tco(mipt)	fixcom	R8	Particle time cutoffs.
tdc	ephcom	R8	Time of writing latest dump to RUNTPE.
tds(:)	mcnp_global	R8, ALLOCATABLE	Tally specifications. See page E–33.
tensn = 0.	mcplot_module	R8	Tension of a rational spline.
tfc(:,:,:)	mcnp_global	R8, ALLOCATABLE	Tally fluctuation charts. See page E–35.
tgp(:)	mcnp_global	R8, ALLOCATABLE	PIKMT biased photon production probability; or temporary KCODE fission production.
thgf(0:50)	fixcom	R8	Table of the thermal cross-section function.
third = one/3.0d+0	mcnp_params	R8, parameter	Floating-point constant 1/3.
titles(7)	mcplot_module	character(len=40)	Titles, legends, and labels.
titles(7)	ra2_mod	character(len=40)	Titles, legends, and labels.
titles(7)	ra2_mod	character(len=40)	Titles, legends, and labels.
tlc	ephcom	R8	Time of writing latest problem summary to OUTP.
tmav(mipt,3)	varcom	R8	Tallies of time to termination.
tmavtc(mipt,3)	tskcom	R8	Task copy of TMAV.
tme	pblcom	R8	Time at the particle position.
tmp(:)	mcnp_global	R8, ALLOCATABLE	Temperatures of the cells.
totgp1	tskcom	R8	Total biased gamma-production cross section.
totm	tskcom	R8	Total microscopic cross section.
totmp	pblcom	R8	Total cross section for previous track.
totpn	tskcom	R8	Total photonuclear cross section.
tpd(7)	tskcom	R8	Stored collision data for PSC calculation.
tpp(64)	tskcom	R8	General-purpose scratch storage.
trf(:,:)	mcnp_global	R8, ALLOCATABLE	Geometry transformations.
trm	ephcom	R8	Time of latest updata of MCPLOT display.
tth(:)	mcnp_global	R8, ALLOCATABLE	Time bins for cell temperatures.
ttn	tskcom	R8	Temperature of the current cell.
twac	varcom	R8	Total weight accepted from surface source file.
twss	varcom	R8	Total weight read from surface source file.
udt(10,0:mxlv)	tskcom	R8	Particle location, direction at higher levels.
udtsav(3,10*mxlv+10)	tskcom	R8	Bank for UDT info.
uold(3)	tskcom	R8	Old direction cosines of track prior to collision.
uuu	pblcom	R8	Particle direction cosine with X-axis.
uvw(3)	pblcom	R8	uvw = equivalent to (uuu,vvv,www).
vcl(:,:,:)	mcnp_global	R8, ALLOCATABLE	Lattice vectors and search constants.
vco(mcoh)	mcnp_data	R8,parameter	Form factor constants for photon scattering.
vec(:,:)	mcnp_global	R8, ALLOCATABLE	Vectors from the VECT card.
vel	pblcom	R8	Speed of the particle.

Variable	Contained in Module	Type	Description
vers	mcnp_data	character(len=5)	Version of code that wrote surface source file.
vic(minc)	mcnp_data	R8, parameter	Form factors for photon scattering.
vol(:)	mcnp_global	R8, ALLOCATABLE	Volumes of the cells in the problem.
vols(:,:)	mcnp_global	R8, ALLOCATABLE	Calculated volumes of the cells.
vtr(3)	tskcom	R8	Velocity of the target nucleus.
vvv	pblcom	R8	Particle direction cosine with Y-axis.
washme = .false.	mcplot_module	logical	Flag for color fill instead of contours.
wc1(mipt)	fixcom	R8	First weight cutoff.
wc2(mipt)	fixcom	R8	Second weight cutoff.
wco(mcoh)	mcnp_data	R8, parameter	Form factors for photon scattering.
wcs1(mipt)	varcom	R8	First weight cutoff modified by SWTM.
wcs1tc(mipt)	tskcom	R8	Task copy of wcs1.
wcs2(mipt)	varcom	R8	Second weight cutoff modified by SWTM.
wcs2tc(mipt)	tskcom	R8	Task copy of wcs2.
wgm(:)	mcnp_global	R8, ALLOCATABLE	Geometry data for superimposed weight window mesh. See page E–42.
wgma(:)	mcnp_global	R8, ALLOCATABLE	Geometry data for superimposed weight window generator mesh. See page E–42.
wgt	pblcom	R8	Particle weight.
wgts(2)	varcom	R8	Range of actual source weights.
wgtstc(2)	tskcom	R8	Task copy of WGTS.
wns(:,:)	mcnp_global	R8, ALLOCATABLE	Actual frequencies of source sampling.
wnvp(4)	ephcom	R8	Window and viewport limits.
wsf	mcnp_plot	real	Linewidth scale factor.
wssa	mcnp_iofiles	character(len=8)	Surface source write file name.
wssi(10)	varcom	R8	Weights of rejected surface source tracks.
wt0	varcom	R8	Weight of each KCODE source point.
wtfasv	pblcom	R8	Accumulated weight of adjoint particle.
wwe(:)	mcnp_global	R8, ALLOCATABLE	Weight-window energy bins.
wwf(:)	mcnp_global	R8, ALLOCATABLE	Lower weight bounds for weight window.
wwfa(:)	mcnp_global	R8, ALLOCATABLE	Weight window generator entering weight array.
wwg(9)	fixcom	R8	Controls for the weight window generator.
wwinp	mcnp_iofiles	character(len=8)	Weight windows input name.
wwk(:)	mcnp_global	R8, ALLOCATABLE	Auger electron generation probability.
wwm(26)	fixcom	R8	Weight window mesh parameters. See page E–42.
wwma(26)	fixcom	R8	Weight window generator mesh parameters. See page E–42.
wwone	mcnp_iofiles	character(len=8)	Weight windows file name(1).
wwout	mcnp_iofiles	character(len=8)	Weight windows file name.
wwp(mipt,8)	fixcom	R8	Weight-window controls.
www	pblcom	R8	Particle direction cosine with Z-axis.
xhom	ephcom	R8	Horizontal coordinate of home position.
xlf	mcnp_plot	real	Postscript plotting left x-axis tick.
xlg	mcplot_module	R8	Horizontal coordinate of legend.
xlk(:)	mcnp_global	R8, ALLOCATABLE	ln of k_{eff} vs. cycle number.
xnm(:)	mcnp_global	R8, ALLOCATABLE	X-ray production bias factors.
xnum	ephcom	R8	X-ray bias number.
xrt	mcnp_plot	real	Postscript plotting right x-axis tick.
xsdir	mcnp_iofiles	character(len=8)	Cross-section directory name.

Variable	Contained in Module	Type	Description
xse85(:,:)	mcnp_global	R8, ALLOCATABLE	Electron data by cell: 10 columns of print table 85.
xspttl	mcplot_module	character(len=10)	Cross-section plot title.
xss(:)	mcnp_global	R8, pointer	Cross-section tables.
xst	mcplot_module	R8	Horizontal coordinate of subtitle.
xunrl	fixcom	R8	Lowest energy of any unresolved resonance probability table.
xunru	fixcom	R8	Highest energy of any unresolved resonance probability table.
xxx	pblcom	R8	X-coordinate of the particle position.
xyz(3)	pblcom	R8	xyz = equivalent to (xxx,yyy,zzz).
xyzmn(3) = (/0.,0.,0./)	mcplot_module	R8	Lower ends of plot axes.
xyzmx(3) = (/0.,0.,0./)	mcplot_module	R8	Upper ends of plot axes.
ybt	mcnp_plot	real	Postscript plotting top y-axis tick.
ycn	tskcom	R8	Temperature-normalized neutron velocity.
yhom	ephcom	R8	Vertical coordinate of home position.
ylg	mcplot_module	R8	Vertical coordinate of legend.
yst	mcplot_module	R8	Vertical coordinate of subtitle.
ytp	mcnp_plot	real	Postscript plotting bottom y-axis tick.
yval	mcplot_module	R8	Current location in plot legend area.
yyy	pblcom	R8	Y-coordinate of the particle position.
zephcm	ephcom	R8	Marker after floating-point part of /EPHCM/.
zero = 0.0d+0	mcnp_params	R8, parameter	Floating-point constant 0.0 for arguments.
zpb9cm(mpb)	pblcom	R8	Marker after floating-point part of /PBLCM/.
zpblcm	pblcom	R8	Marker after floating-point part of /PBLCM/.
zst(:)	mcnp_global	R8, pointer	Data buffer for picture construction.
ztskcm	tskcom	R8	Marker after floating-point part of /TSKCM/.
zvarcm	varcom	R8	Marker after floating-point part of /VARCM/.
zzz	pblcom	R8	Z-coordinate of the particle position.

II. SOME IMPORTANT COMPLICATED ARRAYS

A. Source Arrays

<u>KSD(21,MSD+3) Array</u> Information About Each Source Distribution

KSD(J,K) contains information of type J about source probability distribution K as listed below.

<u>J</u>	
1	problem name of the distribution
2	index of built-in function, if any
3	length of comment in JSCN
4	number of value sets from SI or DS card
5	flag for discrete distribution: L, S, F, Q, or T option
6	flag for distribution of distributions: S or Q option
7	flag for dependent distribution: DS rather than SI
8	flag for DS Q
9	flag for DS T
10	flag for SP V
11	flag for SI F
12	index of the variable of the distribution
13	offset into SPF
14	offset into SSO
15	offset into JSCN
16	offset into WNS
17	number of equiprobable bins in each group, if any
18	flag for biased distribution: SB card present
19	flag for interpolated distribution: A option
20	number of values on SP and/or SB card
21	number of values per bin, including tag from Q or T option

<u>SPF(4,MXXS+1) Array</u> Source Probability Distributions

Each source distribution that is not just an unbiased function has a section of SPF. For a histogram distribution, the four rows of SPF contain

<u>Row</u>	
1	values of the variable (triples for POS, AXS, or VEC)
2	cumulative probability of each bin, possibly biased
3	weight factor to compensate for the bias
4	not used

If the distribution is linearly interpolated, the four rows contain

Row	
1	values of the variable (never triple)
2	unbiased probability density
3	biased probability density, if any
4	cumulative probability for sampling which bin

The above definitions are for the final SPF table as used in MCRUN. In IMCN, the cumulative probabilities start out as probability per bin and the distributions may not yet be normalized.

B. Transport Arrays

<u>GPBLCM(NPBLCM+1) and JPBLCM(LPBLCM+1) Arrays</u> Particle and Collision Descriptors

GPBLCM and JPBLCM are the floating point and integer variables describing the state of a particle at any given time. GPBLCM is equivalenced to XXX, YYY, ZZZ, UUU, VVV, WWW, ERG, WGT, TME, etc., which describe a particle's x, y, and z-coordinates; u, v, and w-direction cosines; and energy, weight, and time. JPBLCM is equivalenced to NPA, ICL, JSU, IPT, IEX, etc., which describe a particle's multiplicity, cell number, surface number, particle type, collision material index, etc. Having all the attributes of a particle in an array form is convenient for storing them temporarily in the GPB9CM and JPB9CM arrays at the start of a history, when generating secondary particles such as neutrons or photons, when generating "pseudo particles" for detectors and DXTRAN, and for banking particles. Banking a particle consists of copying the GPBLCM and JPBLCM arrays to the next block of space in IBNK, and getting a particle from the bank is the reverse. (Banking also consists of copying the UDT1 array if there are repeated structures and the GENR array if there is a weight window generator.)

<u>KTC(2,MXE) and RTC(15,MXE) Arrays</u> Interpolated Cross Sections

When interpolated values of cross sections are calculated at the current particle energy, they are stored in KTC and RTC for possible use later in the calculation of the collision details. The values stored in KTC(I,J) and RTC(I,J) are as follows:

For neutron cross sections, class C, D, or Y

EGO = neutron energy in laboratory frame
ERG = neutron energy in target-at-rest frame

KTC	
1	index in cross-section table for EGO
2	index in cross-section table for ERG

RTC	
1	table interpolation factor for EGO
2	table interpolation factor for ERG
3	absorption (n,0n) cross section for EGO
4	total cross section for EGO at temperature of table
5	total cross section for EGO at cell temperature
6	EGO
7	cell temperature
8	fission cross section

9	
10	number of neutrons emitted by fission
11	probability table elastic cross section (-1 if not in unresolved range)
12	probability table fission cross section
13	probability table neutron heating number
14	probability table (n,γ) radiative capture cross section
15	random number used to sample probability table cross sections

For neutron $S(\alpha,\beta)$ cross sections, class T

KTC

1	index in inelastic cross-section table
2	index in elastic cross-section table

RTC

1	inelastic interpolation factor
2	
3	
4	elastic interpolation factor
5	
6	neutron energy
7	inelastic cross section plus elastic cross section
8	inelastic cross section
9	
10	

For photon cross sections, class P

RTC

1	incoherent scattering cross section
2	incoherent plus coherent scattering cross section
3	incoherent plus coherent plus photoelectric cross section
4	total cross section
5	photon heating number
6	photon energy
7	
8	
9	
10	

For multigroup neutron cross sections, class M

RTC

3	absorption (n,0n) cross section for EGO
5	total cross section for EGO at cell temperature
8	fission cross section
10	number of neutrons emitted by fission

For multigroup photon cross sections, class G

RTC

4	total cross section

C. Tally Arrays

The tallying facilities in MCNP are very flexible. The places in the code where tally scoring is done are very heavily used. The arrays required for flexible and efficient tallying are numerous and complicated. The main tally arrays, grouped by function, are listed below. Arrays in parentheses are not discussed separately but are mentioned in the discussion of the preceding array.

Accumulation of scores: TAL
Controls: JPTAL, IPTAL, LOCDT, ITDS (LOCCT, LOCST), TDS
Fluctuation charts: TFC (JTF, NPC)
Initiation: RTP (IPNT)

<u>TAL(*) Array</u> Tally Scores Accumulation

TAL is in dynamically allocated storage with offset LTAL. LTAL is usually not explicit in the subscript of TAL because the values of the various pointers into TAL include LTAL. TAL is usually divided into three blocks, each of length MXF. If the 15th DBCN card entry is nonzero, then all tallies have the variance of the variance computed and TAL is divided into five blocks. Unless list scoring is in effect (see below), tally scores made during the course of a history are added into tally bins in the first block. At the end of each history, the scores in the first block are added into corresponding places in the second block, their squares are added into the third block, and the first block is zeroed. The fourth and fifth blocks carry the cumulative cubes and fourth-powers of the tally to compute the variance of the variance when applicable. Whenever printed output is called for, the sums in the second block and the sums of squares in the third block are used to calculate and print the tally estimates and their estimated errors.

Each of the blocks in TAL is divided into sections of various lengths, one for each tally in the problem. Each section is an eight-dimensional array of tally bins. The storage sequence is as if the section of TAL were an eight-dimensional Fortran array. The order of the eight dimensions corresponding to a right-to-left reading of the dimensions of a Fortran array, the kind of bins each dimension represents, and the input cards that define them are as follows.

1	cell, surface, or detector bins	F
2	all vs. flagged or all vs. direct	CF, SF or F
3	user bins	FU
4	segment bins	FS
5	multiplier bins	FM
6	cosine bins	C
7	energy bins	E
8	time bins	T

The number of bins in each dimension is determined by rules set forth in the descriptions of the input cards in Chapter 3.

An alternative way of entering scores into the first block is automatically used if the number of scores per history is sufficiently small compared to the size of the block. Only the first of the three (or five) blocks in TAL is affected. The procedure is as follows. Index JTLS is incremented by 2, the score is entered at TAL(JTLS−1), and the location where the score would otherwise have gone is entered at TAL(JTLS). At the end of the history, scores with the same location are consolidated, the scores and their squares are added into the second and third blocks, and JTLS is set to zero. This technique is called list scoring. The scoring described previously is called table scoring. The reason for using list scoring is speed. It is used in only a small minority of problems but can in some cases make a big difference in running time.

<u>JPTAL(8,NTAL) Array</u>   Basic Tally Information

JPTAL(J,K) contains integer information of type J about tally K. Each pointer in JPTAL includes the offset of the array pointed into.

<u>J</u>	
1	problem number of the tally
2	tally type: 1, 2, 4, 5, 6, 7, or 8
3	NQW particle type: 1=N, 2=P, 3=P,N, 4=E, 6=E,P, 7=E,P,N
4	0 if nothing, 1 if asterisk, 2 if plus, on F card
5	offset in the first block in TAL of the section for tally K
6	location of the tally comment in ITDS
7	location in TAL of the tally fluctuation chart bin
8	0 if not a point detector tally 1 for a point detector tally 2 for a ring detector tally 3 for a flux image pinhole tally 4 for a flux image radiograph tally 5 for a flux image cylindrical surface tally

<u>IPTAL(8,6,NTAL) Array</u>   Guide to Tally Bins

IPTAL(I,J,K) contains information of type J about the bins of type I of tally K. The eight bin types I are defined above under TAL. The information types J are listed below, subject to the exceptions noted. Each pointer in IPTAL includes the offset of the array pointed into.

<u>J</u>	
1	offset in TDS or ITDS of specifications for the bins. If there is just one unbounded bin, the value is zero. Exceptions I=2: for cell or surface tally: location in ITDS of flagging cells for detector tally: the number of direct bins (0 or 1) I=4: program number of pseudocell for segmenting surfaces
2	offset in TDS of bin multipliers Exceptions I=1: no meaning I=2: for cell or surface tally: location in ITDS of flagging surfaces for detector tally: offset in TDS of cell contributions I=3: location in TDS of the dose function I=4: offset in TDS of the table of segment divisors

3	number of bins, which is never less than one
4	number of bins including a total bin whether there actually is a total bin or not Exceptions I=1 and I=2 have no meaning.
5	coefficients for calculating the location of a bin, given the eight bin indices
6	flag (0/1 = no/yes) cumulative tally bin

<u>LOCDT(2,MXDT) Array</u>   Detector-Tally Locators

LOCDT(1,J) is the program number of the tally of which detector J is a part. LOCDT(2,J) is the offset in the first block of TAL of the seven-dimensional array where scores for detector J are made.

<u>ITDS(LIT+1) Array</u>   Tally Specifications

ITDS contains blocks (which are in no particular order and are accessed only through pointers) that contain some of the specifications of the tallies of the problem. ITDS is in dynamically allocated storage with offset LITD. LITD is usually not explicit in the subscript of ITDS because the values of the various pointers into ITDS include LITD.

Tally Comment

The value of JPTAL(6,K) is the location in ITDS of the comment for tally K. The first element of the comment is the number of additional elements in the comment. Each line of 67 characters is contained in 23 elements of ITDS and packed three characters per element. The packing uses the ICHAR function and a shift factor of 256. The characters are unpacked and processed by the CHAR function before being printed.

Flagging Cells and Surfaces

The values of IPTAL(2,1,K) and the values of IPTAL(2,2,K) are the locations in ITDS of lists of the program numbers of flagging cells and flagging surfaces, respectively, for tally K. The first item of each list is the number of cells or surfaces in the list.

Cell and Surface Bins

The value of IPTAL(1,1,K) is the offset in ITDS of the description of the cell or surface bins of cell or surface tally K. The structure of the description is

$$\begin{tikzcd}[column sep=small]
P_1 P_2 \ldots P_N \arrow[r, bend left] \arrow[dr, bend right] & n_{11} & I_{11} & I_{21} \ldots I_{nI} & M_1 & L_{12} \arrow[rr, bend left] & L_{13} \ldots L_{1M} \arrow[rrr, bend left] & n_{12} & I_{12} I_{22} \ldots I_{n2} & n_{13} & I_{13} & I_{23} \ldots \\
 & n_{21} & I_{11} & I_{21} \ldots I_{n1} & M_2 & L_{22} \arrow[rr, bend left] & L_{23} \ldots L_{2M} \arrow[rrr, bend left] & n_{22} & I_{12}\ I_{22} \ldots I_{n2} & n_{23} & I_{13} & I_{23} \ldots
\end{tikzcd}$$

where

N	=	number of cell or surface bins in tally K
P_i	=	pointer to specifications for bin *i*
n_{ij}	=	number of cells or surfaces in level *j* of bin *i*

I_{ij}	=	program number of a cell or surface in level *j*. If negative, it is a lattice cell and the following three entries are element indices I,J,K.
M_i	=	number of levels in bin *i* minus one. If zero, no remaining data follows for this bin.
L_{ij}	=	pointer to specifications for level *j* of bin *i*

Cell and Surface Tally Pointers

The value of LOCCT(I,J) if J is a cell—or LOCST(I,J) if J is a surface—is the location in ITDS of a table which locates the sections of TAL where tally scoring is done when a particle of type I passes through cell or surface J. The table is organized this way:

$$N\ \ T_1\ \ m_1\ \ L_{11}\ \ L_{21}\ldots L_{m1}\ldots T_N\ \ m_N\ \ L_{1N}\ \ L_{2N}\ldots L_{mN}$$

where

N	=	number of tallies for particle type I which include cell or surface J
T_i	=	program number of a tally
m_i	=	number of bins that involve cell or surface J
L_{ji}	=	cell or surface bin number

<u>TDS(LTD+1) Array</u> Tally Specifications

TDS contains blocks, in no particular order and accessed only through pointers, that contain some of the specifications of the tallies of the problem. TDS is in dynamically allocated storage with offset LTDS. LTDS is usually not explicit in the subscript of TDS because the values of the various pointers into TDS include LTDS.

Detector Bins

For detector tally K, the value of IPTAL(1,1,K) is the offset in TDS of the description of the detector bins. The description contains the information from the F card, modified for faster use in TALLYD. Five elements of TDS are used for each detector:

	Point detector	Ring detector
1	X	a
2	Y	r
3	Z	1, 2, or 3 for x, y, or z
4	R	R
5	$\lvert 2\pi R^3/3\rvert$	$\lvert 2\pi R^3/3\rvert$

Flux Image Detectors

1-3	pinhole center (FIP) or image grid center (FIR, FIC)
4-6	image grid center for FIP
7-9	direction cosines of axis perpendicular to image grid
10-12	direction cosines of the t image axis
13-15	direction cosines of the s image axis
16	pinhole radius (FIP) or cylinder radius (FIC)
17	collimator radius for restricting image size
18	pinhole-to-grid distance (FIP) or flag for random grid location (FIR, FIC)
19	distance from problem origin to image grid center

Cell Contributions

For detector tally K, the value of IPTAL(2,2,K) is the offset in TDS of the table of cell contributions. The information in the table is exactly as it is on the PD card.

Simple Bins and Multipliers

The value of IPTAL(I,1,K) for I = 3, 6, 7, or 8 is the offset in TDS of a table of bins for tally K. The information in the table is as it came from the corresponding input card except that no T or NT on the card appears in the table. The value of IPTAL(I,2,K) for I = 6, 7, or 8 is the offset in TDS of a table of bin multipliers for tally K. The information in the table is exactly as it is on the input card.

Segment Bin Divisors

For cell or surface tally K, the value of IPTAL(4,2,K) is the offset in TDS of the table of segment bin divisors. Except for a type 1 tally without any SD card, the table exists even if there is no FS card. The table is a two-dimensional array. One dimension is for cell or surface bins and the other is for the segment bins. The segment bin index changes faster. If segment bin divisors are not provided on an SD card, they are calculated or derived from VOL or AREA data, if possible, by MCNP according to the tally type:

tally type	2	4	6	7
divisor	area	volume	mass	mass

Multiplier Bins

The value of IPTAL(5,2,K) is the offset in TDS of a table of the constant multipliers for the multiplier bins from the FM card of tally K. If there is anything more on the FM card than just a constant multiplier for each bin, the value of IPTAL(5,1,K) is the offset in TDS of a table of bin descriptions:

$$\begin{tikzcd}[column sep=small] N & P_1 \arrow[rrrr, bend left] & P_2 \arrow[rrrrrrr, bend left] & \ldots P_N & I_1 & n_1 & R_{11} & R_{21}\ldots R_{n1} & I_2 & n_2 & R_{12} & R_{22}\ldots R_{n2}\ldots \end{tikzcd}$$

where

N = number of P's.

P_i = pointer to the description of a bin or attenuator. If the FM card has only a constant for some bin, then $P_i = 0$ for that bin. If the FM card has C m but nothing more for a bin (which makes it a track-count bin) then $P_i = -1$. If P_i points to an attenuator that appears inside parentheses on the FM card, it is negative.

I_i = for a regular bin, the program number of the material *m* specified on the FM card; for an attenuator, $I_i = -1$.

n_i = for a regular bin, the number of entries (including both reaction numbers and operators) in the bin description. If the list of reaction numbers in the bin includes the elastic or the total cross section, n_i is negative. For an attenuator, n_i is the number of entries, including material numbers and superficial-density values. If a regular bin appears on the FM card within parentheses that also contain an attenuator, n_i has 10000000 added to it for an attenuator to the right of the bin and 20000000 for an attenuator to the left.

R_{ji} = for a regular bin, a reaction number or operator. The sum operator, indicated by a colon on the FM card, is stored here as the value 100003. For an attenuator, the R_{ji} are alternating cell numbers and superficial-density values.

Dose Function

The value of IPTAL(3,2,K) is the location in TDS of the dose function table for tally K. The first element in the table is the length N. It is followed by the N values of the energy and then the N values of the function. N is preceded by an indicator of the type of interpolation: 0 for log-log, 1 for lin-log, 2 for log-lin, and 3 for lin-lin.

<u>TFC(6,20,NTAL*(NPERT+1)) Array</u> Tally Fluctuation Charts

The value of TFC(I,J,K) is the tally value (I=1), the error (I=2), the figure of merit (I=3), the variance of the variance (I=4), the Pareto slope (I=5), and a locator for the Pareto tail plot (I=6) for line J of the tally fluctuation chart for tally K. The tally bin involved is designated by the eight indices in JTF(I,K) for I = 1 to 8. The number of histories run at the point where the entries for a line were calculated is stored in NPC(J). Initially a line is calculated every 1000 histories. When the 20th line is generated, the history increment is doubled. When the time comes to generate the 21st line, the odd-numbered lines are eliminated, the data in line J are moved to line J/2 for J = 2 to 20 by 2, and the new data are put in line 11.

<u>RTP(LRT) Array</u> Information from Tally Input Cards

The information from most tally input cards is stored without much modification in temporary array RTP. Numbers are stored as is. Special characters are encoded. After all the input cards have been read, subroutine ITALLY sets up the permanent tally control arrays from the information in RTP. The main reason for this two-step process is that some of the control arrays depend in a complicated way on information from more than one input card. It is simpler to generate the control arrays with all the input data available at the same time than to do it as the cards are read.

Pointer array IPNT(2,MKTC,0:NTAL) is defined as the tally cards are read. The information from tally card type J of tally K begins at RTP(IPNT(1,J,K)) and occupies IPNT(2,J,K) elements of RTP. The tally card type numbers J are given in KRQ(3,N) for each type N of input card. KRQ(3,N) is defined by DATA statements in module MCNP_INPUT. KRQ(3,N) is zero for nontally input cards. There is no tally card type 1. IPNT(1,1,K) is used for bits that reflect T or NT on certain cards and indicate whether a total bin needs to be included. The value of IPNT(1,2,K) is 1, 2, 3, 4, or 5, depending on whether the F card for the tally has blank, X, Y, Z, or W with the F, and it is negative if there is an asterisk on that card.

D. *Accounting Arrays*

MCNP regularly collects and prints data on the behavior of the particles transported through the problem geometry. This is accounting information which shows what MCNP actually did, in contrast to the tallies that are estimates of physically measurable quantities. The accounting information is essential to a user who is trying to make his problem run faster. The arrays where the accounting data are collected and the titles of the tables where they are printed are as follows:

PAX	Problem Summary
PAC	Problem Activity in Each Cell (Print Table 126)
PWB	Weight Balance in Each Cell (Print Table 130)
PAN	Activity of Each Nuclide in Each Cell (Print Table 140)
PCC, FEBL	Summary of Photons Produced in Neutron Collisions

PAX(6,21,MIPT) Array   Problem Summary

The value of PAX(I,J,K) is the total of type I data for mechanism J and particle type K.

I	
1	number of tracks created
2	weight created
3	energy created
4	number of tracks terminated
5	weight terminated
6	energy terminated

J	Particle	Creation Mechanism	Loss Mechanism
1	NPE	source	escape
2	NPE		energy cutoff
3	NPE		time cutoff
4	NPE	weight window	weight window
5	NPE	cell importance	cell importance
6	NPE	weight cutoff	weight cutoff
7	NPE	e or t importance	e or t importance
8	NP	DXTRAN	DXTRAN
9	NP	forced collisions	forced collisions
10	NP	exponential transform	exponential transform
For neutrons only			
11	N	upscattering	downscattering
12	N	photonuclear	capture
13	N	(n,xn)	loss to (n,xn)
14	N	prompt fission	loss to fission
15	N	delayed fission	
For photons only			
11	P	from neutrons	Compton scatter
12	P	bremsstrahlung	capture
13	P	p-annihilation	pair production
14	P	photonuclear	photonuclear absorption
15	P	electron x-rays	
16	P	1st fluorescence	

17	P	2nd fluorescence	

For electrons only

11	E	pair production	scattering
12	E	Compton recoil	bremsstrahlung
13	E	photo-electric	
14	E	photon auger	
15	E	electron auger	
16	E	knock-on	

For the printed table, the weight totals are divided by the number of histories and the energy totals are divided by the total weight of source particles.

<u>PAC(MIPT,10,MXA) Array</u>   Problem Activity in Each Cell

The value of PAC(LPAC+I,J,K) is the total of type J data for particle type I in cell K. If a particle becomes lost, a small amount of erroneous information gets added into PAC.

<u>J</u>	
1	number of tracks entering cell K
2	population of cell K: the number of tracks, including source tracks, entering for the first time
3	number of collisions in cell K
4	weight entering collisions
5	energy * time interval in cell K * weight
6	energy * path length * weight
7	path length in cell K
8	mean free path * path length * weight
9	time interval * weight
10	path length * weight

The quantities printed are
Tracks Entering = PAC(LPAC+I,1,K)
Population = PAC(LPAC+I,2,K)
Collisions = PAC(LPAC+I,3,K)
Collisions * weight (per history) = PAC(LPAC+I,4,K) / number of histories
Number Weighted Energy = PAC(LPAC+I,5,K) / PAC(LPAC+I,9,K)
Flux Weighted Energy = PAC(LPAC+I,6,K) / PAC(LPAC+I,10,K)

Average Track Weight (Relative) = PAC(LPAC+I,10,K) * importance of cell K / [PAC(LPAC+I,7,K) * importance of source cell]
Average Track MFP = PAC(LPAC+I,8,K) / PAC(LPAC+I,10,K)

<u>PWB(MIPT,22,MXA) Array</u>   Weight Balance in Each Cell

The value of PWB(LPWB+I,J,K) is the net weight change of type J for particle type I in cell K. If a particle becomes lost, a small amount of erroneous information gets added into PWB. Table values are divided by the number of histories before being printed.

J	Table Heading	
	External	
1	Entering	weight of particles entering cell K
2	Source	weight of created source particles
3	Time Cutoff	weight of particles killed by time cutoff
4	Energy Cutoff	weight of particles killed by energy cutoff
5	Exiting	weight of particles exiting cell K
	Variance Reduction	
6	Weight Window	net weight change due to weight-window Russian roulette
7	Cell Importance	net weight change due to splitting and Russian roulette in importance sampling
8	Weight Cutoff	net weight change due to weight cutoff
9	E or T Importance	net weight change due to energy or time splitting, Russian roulette
10	DXTRAN	net weight change due to DXTRAN
11	Forced Collision	net weight change due to forced collision
12	Exponential Transform	net weight change due to exponential transform
	Physical (neutrons)	
13	(n,xn) processes	weight of new tracks produced by other nonfission
14	Fission	weight of fission neutrons produced
15	Capture	weight lost to capture
16	Loss to (n,xn)	weight of neutrons lost to (n,xn)
17	Loss to Fission	weight of neutrons lost to fission
21	Photonuclear	weight of neutrons created from (γ,n) reactions
	Physical (photons)	
13	From Neutrons	weight of neutron-induced photons
14	Bremsstrahlung	net weight created by bremsstrahlung
15	P-annihilation	net weight created by p-annihilation
16	Electron x-rays	net weight created by electron x-rays
17	Fluorescence	net weight created by double fluorescence
18	Capture	weight lost to photoatomic capture
19	Pair Production	net weight created by pair production
20	Photonuclear absorption	weight lost to photonuclear absorption
21	Photonuclear	weight created by photonuclear reactions
	Physical (electrons)	
13	Pair production	net weight created by pair production
14	Compton recoil	net weight created by Compton scatter
15	Photoelectron	net weight created by photoelectrons
16	Photon Auger	net weight created by photon auger
17	Electron Auger	net weight created by electron auger
18	Knock-on	net weight created by knock-ons

PAN(3,8,NPN) Array   Activity of Each Nuclide in Each Cell

The value of PAN(LPAN+I,J,IPAN(K)+N–1) is the total of type J data for particle type I for the N^{th} nuclide in cell K. IPAN(M+1) = IPAN(M) + number of nuclides in the material of cell M. IPAN(1) = 1 and NPN = IPAN(MXA+1) −1. If a particle becomes lost, a small amount of erroneous information gets added into PAN.

J	(for neutron activity)
1	number of collisions with N^{th} nuclide of cell K
2	weight entering collisions
3	weight lost to capture
4	weight gain by fission
5	weight gain by other (n,xn) inelastic processes
6	number of neutron-induced photons produced by N^{th} nuclide
7	average neutron-induced photon weight produced by N^{th} nuclide
8	average neutron-induced photon energy produced by N^{th} nuclide

J	(for photoatomic activity)
1	number of photoatomic collisions with N^{th} nuclide of cell K
2	weight entering collisions
3	weight lost to photoatomic capture

J	(for photon photonuclear activity)
1	number of photonuclear collisions with N^{th} nuclide of cell K
2	weight entering photonuclear collisions
3	number of photonuclear photons produced
4	average photon weight produced by photonuclear reactions
5	average photon energy produced by photonuclear reactions
6	number of neutrons produced by (γ,n) photonuclear reactions
7	average neutron weight produced from photonuclear reactions
8	average neutron energy produced from photonuclear reactions

The quantities printed are
Total Collisions = PAN(LPAN+I,1,L)
Collisions * Weight = PAN(LPAN+I,2,L) / number of histories
Weight Lost to Capture = PAN(LPAN+I,3,L) / number of histories
Weight Gain by Fission = PAN(LPAN+1,4,L) / number of histories
Weight Gain by (n,xn) = PAN(LPAN+1,5,L) / number of histories
Total from Neutrons = PAN(LPAN+2,4,L)
Weight from Neutrons = PAN(LPAN+2,5,l) / number of histories
Average Photon Energy = PAN(LPAN+2,6,L) / PAN(LPAN+2,5,L)

PCC(3,MXA*KPT(2)) Array   Summary of Photons Produced in Neutron Collisions

The value of PCC(J,K) is the total of type J data for cell K. If a particle becomes lost, a small amount of erroneous information may be added into PCC.

J	
1	number of neutron-induced photons
2	weight of neutron-induced photons
3	weight * energy of neutron-induced photons

The quantities printed are
Number of Photons = PCC(1,K)
Weight Per Source Neutron = PCC(2,K) / number of histories
Energy Per Source Neutron = PCC(3,K) / number of histories
Average Photon Energies = PCC(3,K) /PCC(2,K)
Energy/Gram Per Source Neutron = PCC(3,K) /
[cell mass * number of histories]
Weight/Neutron Collision = PCC(2,K) / PAC(LPAC+1,4,K)
Energy/Neutron Collision = PCC(3,K) / PAC(LPAC+1,4,K)

FEBL(2,K) Array   Summary of Photons Produced in Neutron Collisions

The value of FEBL(J,K) is the total of type J data for photon energy bin K, where K=16 for continuous energy problems and K=IGM=number of multigroup energy groups. The energy bin bounds are in array EBL(K) in common block /TABLES/.

J	
1	number of neutron-induced photons
2	weight of neutron-induced photons

The quantities printed are
Number of Photons = FEBL(1,K)
Number Frequency = FEBL(1,K) / PAX(2,1,3)
Weight of Photons = FEBL(2,K) / number of histories
Weight Frequency = FEBL(2,K) / PAX(2,2,3)

E. KCODE Arrays

OSUM(I) Array   Cumulative k_{eff} over active cycles
OSUM(I) = OSUM(I) + SUMK(I)/NSRCK, I=1,3.

OSUM2(I,J) Array   Cumulative k_{eff} covariance quantities
OSUM2(I,J) = OSUM2(I,J) + ZZ(I) * ZZ(J)
where ZZ(K) = SUMK(K)/NSRCK.

RLT(I,J) Array   Prompt removal lifetimes for current active cycle
RLT(I,J)   Prompt removal lifetimes for current active cycle.
I = 1/2/3/4 = collision/absorption/track length/fission
J = 1 sum of WGT*TME over cycle
J = 2 sum of WGT over cycle
Note: RLT(4,1) is summed over all histories and used only for the prompt fission lifespan. RLT(4,2) is unused.

RKPL(19,MRKP) Array KCODE Quantities for Plotting

The value of RKPL(I,J) for the J^{th} cycle of a KCODE problem:

J	
1	k_{eff} (collision)
2	k_{eff} (absorption)
3	k_{eff} (track length)
4	prompt removal life (collision)
5	prompt removal life (absorption)
6	average collision k_{eff}
7	average collision k_{eff} standard deviation
8	average absorption k_{eff}
9	average absorption k_{eff} standard deviation
10	average track length k_{eff}
11	average track length k_{eff} standard deviation
12	average col/abs/trk-len k_{eff}
13	average col/abs/trk-len k_{eff} standard deviation
14	average col/abs/trk-len k_{eff} by cycles skipped
15	average col/abs/trk-len k_{eff} by cycles skipped standard deviation
16	prompt removal lifetime (col/abs/trk-len)
17	prompt removal lifetime (col/abs/trk-len) standard deviation
18	number of histories used in each cycle
19	col/abs/trk-len k_{eff} figure of merit

RSUM(I) Array Cumulative prompt removal lifetimes over active cycles
RSUM(I) = RSUM(I) + RLT(I,1)/RLT(I,2), I=1,3.

RSUM2(I,J) Array Cumulative prompt removal lifetime covariance quantities
RSUM2(I,J) = RSUM2(I,J) + RL(I) * RL(J)
where RL(K) = RLT(K,1)/RLT(K,2)

SUMK(I) Array SUMK(I)/NSRCK is k_{eff} for current cycle
I = 1/2/3 = collision/absorption/track length

SUMP(3*NPERT) Array Track length estimate of k_{eff} for each perturbation, IP=1,NPERT

SUMP(IP)	track length estimate of k_{eff} for current cycle
SUMP(NPERT+IP)	cumulative SUMP(IP) over all cycles
SUMP(2*NPERT+IP)	cumulative SUMP(IP)**2 to get standard deviations

SUMP(IP), IP=1, NPERT is like SUMK(3)
SUMP(NPERT+IP) is like OSUM(3)
SUMP(2*NPERT+IP) is like OSUM2(3,3)

In multitasking, SUMP(KSUM+IP) is accumulated into SUMP(LSUM+IP), but there is no need for nor space saved for SUMP(KSUM+NPERT+IP) or SUMP(KSUM+2*NPERT+IP).

F. *Universe Map/Lattice Activity Arrays for Table 128*

MAZP(3,MXA) Array Used in RSLMAZ to point inside MAZE array.
  MAZP(1,IC) = I, index of cell IC in MAZU(j) list.
  MAZP(2,IC) = universe address J of cell IC.
  MAZP(3,IC) = address J of universe filling cell IC.

MAZU(NMZU) Array Used in RSLMAZ to point inside MAZE array. The MAZE(NMAZ) array contains the number of sources, tracks entering and collisions in each repeated structures/ lattice element:

  MAZU(J-3) = I = universe name.
  MAZU(J-2) = finite lattice cell filling universe I.
  MAZU(J-1) = total number of lowest level elements below U=I.
  MAZU(J) = NE = number of cells/elements in universe I.
  MAZU(J+K) = number of elements below K^{th} cell/universe.
  MAZU(J+NE+K) = K^{th} cell in universe I (repeated structures).
  MAZU(J+NE+K) = first cell of universe filling K^{th} lattice element.

G. *Weight Window Mesh Parameters*

WWM(1-3)	total number of fine meshes in x,y,z or r,z,theta directions
WWM(4-6)	origin (corner of box for rectangular geometry, bottom and center point for cylindrical geometry)
WWM(7-9)	number of coarse meshes in each direction
WWM(10-12)	cylindrical geometry top center point
WWM(13-15)	cylindrical geometry point on radius and bottom plane
WWM(16-18)	cylindrical geometry direction cosines from bottom center point to point on radius
WWM(19)	cylindrical geometry radius
WWM(20-22)	cylindrical geometry cosines of axis
WWM(23)	cylindrical geometry axis length
WWM(24-26)	cylindrical geometry direction cosines of the cross product of the radial direction and axial direction; necessary for full revolution theta determination
WGM(NWGM)	weight window mesh geometric data with the inclusion of 0^{th} index entries for each dimension. The data are stored as cumulative values.

H. *Perturbation Parameters*

DPTB(3,NPERT*MNNM) Array PERT card density changes that become the perturbation coefficients fixed at code initiation. For each nuclide J of perturbation IP, where J=NPTB(IP),NPTB(IP+1)+1, DPTB(I,J), I has the following values:

I	Description
1	nuclide index, IEX
2	$\delta_1 \Delta v$
3	$\delta_2 \Delta v$

where Δv is the density change term (see page 2–186) of the Taylor Series expansion. δ_1 = 1/0 if the 1st order perturbation is on (METHOD=1,2) or off. δ_2 = 1/0 if the 2nd order perturbation is on (METHOD=1,3) or off.

IPTB(2+2*NPKEY,NPERT) Array   Pointers to RPTB array and other perturbation parameters from PERT card

The six NPKEY perturbation keywords are CELL, MAT, RHO, RXN, ERG and METHOD. For perturbation IP=1,NPERT,

IPTB(1,IP) = perturbation number from PERT card
IPTB(2,IP) = particle type from PERT card
IPTB(1+2*K,IP) = number of entries for keyword K
IPTB(2+2*K,IP) = location in RPTB of PERT card data for keyword K

Exception:

IPTB(13,IP) = 1/2/3 = METHOD
IPTB(14,IP) = 0 for method = 1/2/3;
= 1 for METHOD = –1/–2/–3

```
Example:  PERT6:N,P   CELL  7  8  9  12   METHOD = –2
          IPTB = 6 3    4 12345    0 0    0 0    0 0    0 0    2 1
          RPTB(12345) = 7.   8.   9.   12.
```

NPTB(NPERT+1) Array   Cumulative number of perturbed cross sections used as pointers to DPTB and PTB arrays. NPTB(IP) points to the first nuclide data in DPTB and PTB for the material of perturbation IP. Thus perturbation IP has NPTB(IP+1) – NPTB(IP) ≤ MNNM nuclides in its perturbed material, and the entries in the PTB and RPTB arrays for these nuclides are stored from NPTB(IP) to NPTB(IP+1) –1.

PTB(5,NPERT*MNNM) Array   Perturbation coefficients. The perturbation coefficients $P_{1j'}$ and $P_{2j'}$ described in Chapter 2 (see page 2–192) are stored in the PTB(I,J) array where J=NPTB(IP),NPTB(IP+1) –P1 for the NPTB(IP+1) – NPTB(IP) nuclides of perturbation IP.

PTB(KPTB+1,J) = $P_{1j'}$
PTB(KPTB+2,J) = $P_{2j'}$
PTB(KPTB+3,J) = $x_b(E')$ the macroscopic cross section nuclide J at E'
PTB(KPTB+4,IP) = $P_{1j'}\Delta v + \frac{1}{2}\,(P_{2j'} + P^2_{1j'}\Delta v^2\,)$
PTB(KPTB+5,J) = $x_c\,(E)$

The perturbed value of k_{eff} or a tally is then the unperturbed value times PTB(KPTB+4,IP). If the nuclides in the perturbation are also in the tally (F6, F7, or F4 with FM card with negative constant for atom density multiplier), then PTB(KPTB+4,IP) is corrected by adding $R_{1j'}\Delta v + P_{1j'}R_{1j'}\Delta v^2$ where

$$R_{1j'} = \frac{\sum\limits_{c \in B} \sum\limits_{E \in H} x_c(E)}{\sum\limits_{c \in C} x_c(E)} = \frac{\sum\limits_{J} \text{PTB(KPTB+5,J)}}{\text{PTBTC}}$$

Note that $x_b(E)$ at collision k is saved as PTB(KPTB+3,J) to be used as $x_b(E')$ at collision $k+1$. Also note that PTB(KPTB+4,IP) is stored by perturbation number IP, not J like the rest of the PTB array, leaving NPERT*MNNM - NPERT words unused.

RPTB(IPERT) Array Perturbation parameters from PERT card. RPTB(I) stores the keywords read from the PERT card as pointed to by the IPTB array (see above).

I. *Macrobody and Identical Surface Arrays*

IDNA(K) exactly parallels the LJA(K) array for cell cards
- = 0 when slot k does not involve a macrobody surface
- = n with n>o, is facet n of macrobody
- = –n is facet, but cell card is only using this one facet

IDNT(J) program surface number of master identical surface
- = 0, j is not an identical surface
- = j′, |j′| is the master surface of identical surfaces. The sense gives the sense of surface j with respect to the sense of the master surface j′

IDNS(J) locator in IDNE for list of identical surfaces
- = no identical surfaces
- = m with m locator in IDNE

IDNE(M) list of identical surfaces
- = n number of identical surfaces for surface j
  next n entries are the identical program surfaces (j’s)

IDNE(1) is the number of identical surface sets
IDNE(2) is the total length of IDNE

III. DERIVED STRUCTURES

FMARRY Mesh Tallies

FMARRY is a Fortran 90 allocatable derived structure array that contains all the information unique to each mesh tally. In the code, only one structure, named FM, is used, and it is dimensioned to the number of mesh tallies in the problem. The components of FMARRY are listed below:

Name	Dimension(s)	Attributes	Definition
axs(3)		real	Axis vector for the cylindrical mesh
crs(3)		real	Cross product of AXS and VEC
de	(:)	real allocatable	Energy values for tally dose function
df	(:)	real allocatable	Dose function values
enbin	(:)	real allocatable	Bin values for energy coordinate
fact		real	Multiplication factor
fmarry	(:,:,:,:,:)	real allocatable	Track length tally values
fmerr	(:,:,:,:,:)	real allocatable	Track length tally errors
fmult		real	FM card multiplier
icrd		integer	Mesh coordinate system, 1=rec, 2=cyl
icx		integer	Mesh tally flag for energy times weight tally
id		integer	Mesh tally number assigned by the user
ifm_card		integer	Flag for mesh tally FM card
intrpol		integer	Dose function interpolation method
ipt		integer	Particle type number
ireact	(:)	integer	FM card reaction numbers
itr		integer	Mesh transformation number
lemesh		logical	Flag for EMESH card
mat		integer	Material number for FM reactions
ndfb		integer	Number of dose function bins
nenb		integer	Number of energy bin boundaries
nireact		integer	Size of ireact array

Name	Dimension(s)	Attributes	Definition
nreact		integer	Number of reactions on the FM card
nxrb		integer	Number of x/r bin boundaries
nyzb		integer	Number of y/z bin boundaries
nztb		integer	Number of z/theta bin boundaries
org(3)		real	Origin of the mesh
outf		integer	Output format: 0=column, 1=ij, 2=ik, 3=jk, 4=column-full
vec(3)		real	Vector defining, along with AXS, plane for $\theta=0$
xrbin	(:)	real allocatable	Bin values for x/r coordinate
yzbin	(:)	real allocatable	Bin values for y/z coordinate
ztbin	(:)	real allocatable	Bin values for z/theta coordinate

FM_TEMP_ARRAY   Mesh Tally Scores for Each History

FM_TEMP_ARRAY is an allocatable derived structure that stores the mesh tally scores for each history. Only one of these structures, named FMTAL, is used in the code. There is only one component of this structure:

Name	Dimension(s)	Attributes	Definition
Tally	(:,:,:,:,:)	real allocatable	Stores mesh tally scores for each history

At the end of each history, FMTAL%TALLY is added to FM%FMARRY and the square of FMTAL%TALLY is added to FM%FMERR. FMTAL%TALLY is also used to store the volume of each mesh tally cell in the mesh tally print routine.

APPENDIX F - DATA TABLE FORMATS

MCNP has two *types* and nine *classes* of data. These data are kept in individual *tables* that are often organized into *libraries*. These tables are located with the XSDIR data directory file. These terms, tables, and the basic data table formats are described in this appendix in the following sections:

I. DATA TYPES AND CLASSES

MCNP reads nine *classes* of data from two *types* of data tables. The two types of data tables are:

1. Type 1—standard formatted tables (sequential, 80 characters per record). These portable libraries are used to transmit data from one installation to another. They are bulky and slower to read. Often installations generate Type 2 tables from Type 1 tables using the MAKXSF code (see Appendix C).
2. Type 2—standard unformatted tables (direct-access, binary) locally generated from Type 1 tables. They are not portable except between similar systems such as various UNIX platforms. Type 2 tables are used most because they are more compact and faster to read than Type 1 tables.

Data tables exist for nine *classes* of data: continuous-energy neutron, discrete-reaction neutron, continuous-energy photoatomic interaction, continuous-energy electron interaction, continuous-energy photonuclear interaction, neutron dosimetry, S(α,β) thermal, neutron multigroup, and photoatomic multigroup. A user should think of a data table as an entity that contains evaluation-dependent information about one of the nine *classes* of data for a specific target isotope, isomer, element, or material. For how the data are used in MCNP, a user does not need to know whether a particular table is in Type 1 or Type 2. For a given ZAID, the data contained on Type 1 and Type 2 tables are identical. Problems run with one data type will track problems run with the same data in another format type.

When we refer to data libraries, we are talking about a series of data tables concatenated into one file. All tables on a single library must be of the same *type* but not necessarily of the same *class*. There is no reason, other than convenience, for having data libraries; MCNP could read exclusively from individual data tables not in libraries.

II. XSDIR— DATA DIRECTORY FILE

MCNP determines where to find data tables for each ZAID in a problem based on information contained in a system-dependent directory file XSDIR. The directory file is a sequentially formatted ASCII file with 80-character records (lines) containing free-field entries delimited by blanks.

The XSDIR file has three sections. In the first section, the first line is an optional entry of the form:

DATAPATH = *datapath*

where the word DATAPATH (case insensitive) must start in columns 1–5. The = sign is optional. The directory where the data libraries are stored is *datapath*. The XSDIR directory file can be renamed by item 1. The search hierarchy to find XSDIR and/or the data libraries is:

1. XSDIR = cross-section directory file name on the MCNP execution line,
2. DATAPATH = *datapath* in the INP file message block,
3. the current directory,
4. the DATAPATH entry on the first line of the XSDIR file,
5. the UNIX environmental variable setenv DATAPATH *datapath*,
6. the individual data table line in the XSDIR file (see below under Access Route), or
7. the directory specified at MCNP compile time in the BLOCK DATA subroutine.

The second section of the XSDIR file is the atomic weight ratios. This section starts with the words "ATOMIC WEIGHT RATIOS" (case insensitive) beginning in columns 1–5. The following lines are free-format pairs of ZAID AWR, where ZAID is an integer of the form ZZAAA and AWR is the atomic weight ratio. These atomic weight ratios are used for converting from weight fractions to atom fractions and for getting the average Z in computing electron stopping powers. If the atomic weight ratio is missing for any nuclide requested on an Mn card, it must be provided on the AWTAB card.

The third section of the XSDIR file is the listing of available data tables. This section starts with the word "DIRECTORY" (case insensitive) beginning in columns 1–5. The lines following consist of the seven– to ten–entry description of each table. The ZAID of each table must be the first entry. If a table requires more than one line, the continuation is indicated by a + at the end of the line. A zero indicates the entry is inapplicable. Unneeded entries at the end of the line can be omitted.

The directory file has seven to eleven entries for each table. They are:

1.	Name of the Table	character * 10
2.	Atomic Weight Ratio	real

3.	File Name	character * 8
4.	Access Route	character * 70
5.	File Type	integer
6.	Address	integer
7.	Table Length	integer
8.	Record Length	integer
9.	Number of Entries per Record	integer
10.	Temperature	real
11.	Probability Table Flag	character * 6

1. Name of the Table. This is usually the ZAID: 3 characters for Z, 3 characters for A, a decimal point, 2 characters for evaluation identification, and a tenth character used to identify continuous-energy neutron tables by the letter C, discrete-reaction neutron tables by D, dosimetry tables by Y, S(α,β) thermal tables by T, continuous-energy photoatomic tables by P, continuous-energy photonuclear tables by U, continuous-energy electron tables by E, multigroup neutron tables by M, and multigroup photon tables by G. For the S(α,β) tables, the first 6 characters contain a mnemonic character string, such as LWTR.01T.
2. Atomic Weight Ratio. This is the atomic mass divided by the mass of a neutron. The atomic weight ratio here is used only for neutron kinematics and should be the same as it appears in the cross-section table so that threshold reactions are correct. It is the quantity *A* used in all the neutron interaction equations of Chapter 2. This entry is used only for neutron tables.
3. File Name. The file name is the name of the library that contains the table and is a string of eight characters in a form allowed by the local installation.
4. Access Route. The access route is a string of up to 70 characters that tells how to access the file if it is not already accessible, such as a UNIX directory path. If there is no access route, this entry is zero.
5. File Type. 1 or 2.
6. Address. For Type 1 files the address is the line number in the file where the table starts. For Type 2 files, it is the record number of the first record of the table.
7. Table Length. A data table consists of two blocks of information. The first block is a collection of pointers, counters, and character information. The second block is a solid sequence of numbers. For Type 1 and Type 2 tables, the table length is the length (total number of words) of the second block.
8. Record Length. This entry is unused for Type 1 files and therefore is zero. For Type 2 direct access files it is a processor-dependent attribute. The record length is a multiple of the number of entries per record, the number of 8-bit bytes in the record for most systems. Thus for 512 entries per record, the record length is 4096 for double-precision data on most UNIX workstations, 2048 for single-precision data on most UNIX workstations, etc.
9. Number of Entries per Record. This is unused for Type 1 files and therefore is zero. For Type 2 files it is the number of entries per record. Usually this entry is set to 512.
10. Temperature. This is the temperature in MeV at which a neutron table is processed. This entry is used only for neutron data.
11. Probability Table Flag. The character word “ptable” indicates a continuous-energy neutron nuclide has unresolved resonance range probability tables.

III. DATA TABLES

The remainder of this Appendix is designed for the user who wishes to know a great deal about how data are stored in data tables and in MCNP. First we describe how to find a specific table on a Type 1 or Type 2 library. Then we document the detailed format of the various blocks of information for each *class* of data.

Three arrays are associated with each data table. The NXS array contains various counters and flags. The JXS array contains pointers. The XSS array contains all of the data. These arrays are the same regardless of the *type* of a specific table. The arrays are manipulated internally by MCNP. Within a data table, the counter and pointer arrays are dimensioned to NXS(16) and JXS(32). In MCNP the same arrays are dimensioned to NXS(16,IEX) and JXS(32,IEX), where IEX is the index of the particular table in the problem. There is no limit to the number of tables or their size other than available space on a particular computing platform.

To locate data for a specific table (external to MCNP) it is necessary to extract several parameters associated with that table from the directory file XSDIR. The file name obviously indicates the name of the library on which the table is stored. Other important parameters from the viewpoint of this Appendix are file type (NTY), address (IRN), table length (ITL), and number of entries per record (NER).

A. Locating Data on a Type 1 Table

Because Type 1 tables are 80-character card-image files, the XSDIR address IRN is the line number of the first record, or the beginning, of the table. The first 12 records (lines) contain miscellaneous information as well as the NXS and JXS arrays. The format follows.

Address			
Relative	Absolute	Contents	Format
1	IRN	HZ,AW(0),TZ,HD	A10,2E12.0,1X,A10
2	IRN+1	HK,HM	A70,A10
3–6	IRN+2	(IZ(I),AW(I), I=1,16)	4(I7,F11.0)
7–8	IRN+6	(NXS(I), I=1,16)	8I9
9–12	IRN+8	(JXS(I), I=1,32)	8I9

The variables are defined in Tables F.1–F.3 for neutron, photoatomic, dosimetry, and S(α,β) thermal libraries. These variables are defined in Table F.34 and Table F.35 for multigroup data. They are defined in Table F.57 and Table F.8 for photonuclear data.

The XSS array immediately follows the JXS array. All data from the XSS array are read into MCNP with a 4E20.0 format. (When Type 1 tables are created, floating-point numbers are written in 1PE20.12 format and integers are written in I20 format.) The length of the XSS array is given by the table length, ITL, in the directory (also by NXS(1) in the table itself). The number of records required for the XSS array is (ITL+3)/4. A Type 1 library is shown in Figure F-1.

Starting Address (Line Number)	Number of Records	Contents
IRN_1=1	12	misc. including NXS_1, JXS_1
IRN_1+12	$(ITL_1+3)/4$	XSS_1
IRN_2	12	misc. including NXS_2, JXS_2
IRN_2+12	$(ITL_2+3)/4$	XSS_2
.	.	.
.	.	.
IRN_n	12	misc. including NXS_n, JXS_n
IRN_n+12	$(ITL_n+3)/4$	XSS_n

IRN_i, ITL_i are the addresses and table lengths from XSDIR
n=number of tables contained in library

Figure F-1. Layout of a Type 1 Library

Table F.1
Definition of the NXS Array

NTY	1 or 2 Continuous Energy or Discrete Reaction Neutron	3 Dosimetry	4 Thermal	5 Continuous Energy Photoatomic
NXS(1)	Length of second block of data	Length of second block of data	Length of second block of data	Length of second block of data
NXS(2)	ZA=1000*Z+A	ZA=1000*Z+A	IDPNI=inelastic scattering mode	Z
NXS(3)	NES=number of energies		NIL=inelastic dimensioning parameter	NES=number of energies
NXS(4)	NTR=number of reactions excluding elastic	NTR=number of reactions	NIEB=number of inelastic exiting energies	NFLO=length of the fluorescence data divided by 4
NXS(5)	NR=number of reactions having secondary neutrons excluding elastic		IDPNC=elastic scattering mode	NSH=number of electron shells
NXS(6)	NTRP=number of photon production reactions		NCL=elastic dimensioning parameter	

Table F.1 (Cont.)
Definition of the NXS Array

NTY	1 or 2 Continuous Energy or Discrete Reaction Neutron	3 Dosimetry	4 Thermal	5 Continuous Energy Photoatomic
NXS(7)			IFENG=secondary energy mode	
NXS(8)	NPCR=number of delayed neutron precursor families			
......				
......				
......				
NXS(15)	NT=number of PIKMT reactions			
NXS(16)	0=normal photon production −1=do not produce photons			
Note that many variables are not used, allowing for expansion in the future.				

Table F.2
Definition of the JXS Array

NTY	1 or 2 Continuous Energy or Discrete Reaction Neutron	3 Dosimetry	4 Thermal	5 Continuous Energy Photoatomic
JXS(1)	ESZ=location of energy table	LONE=location of first word of table	ITIE=location of inelastic energy table	ESZG=location of energy table
JXS(2)	NU=location of fission nu data		ITIX=location of inelastic cross sections	JINC=location of incoherent form factors
JXS(3)	MTR=location of MT array	MTR=location of MT array	ITXE=location of inelastic energy/ angle distributions	JCOH=location of coherent form factors
JXS(4)	LQR=location of Q-value array		ITCE=location of elastic energy table	JFLO=location of fluorescence data

Table F.2 (Cont.)
Definition of the JXS Array

NTY	1 or 2 Continuous Energy or Discrete Reaction Neutron	3 Dosimetry	4 Thermal	5 Continuous Energy Photoatomic
JXS(5)	TYR=location of reaction type array		ITCX=location of elastic cross sections	LHNM=location of heating numbers
JXS(6)	LSIG=location of table of cross-section locators	LSIG=location of table of cross- section locators	ITCA=location of elastic angular distributions	LNEPS=location of the number of electrons per shell
JXS(7)	SIG=location of cross sections	SIGD=location of cross sections		LBEPS=location of binding energy per shell
JXS(8)	LAND=location of table of angular distribution locators			LPIPS=location of probability of interaction per shell
JXS(9)	AND=location of angular distributions			LSWD=location of array of offsets to shellwise data
JXS(10)	LDLW=location of table of energy distribution locators			SWD=location of shellwise data in PDF and CDF form
JXS(11)	DLW=location of energy distributions			
JXS(12)	GPD=location of photon production data			
JXS(13)	MTRP=location of photon production MT array			
JXS(14)	LSIGP=location of table of photon production cross-section locators			
JXS(15)	SIGP=location of photon production cross sections			

Table F.2 (Cont.)
Definition of the JXS Array

NTY	1 or 2 Continuous Energy or Discrete Reaction Neutron	3 Dosimetry	4 Thermal	5 Continuous Energy Photoatomic
NXS(16)	LANDP=location of table of photon production angular distribution locators			
JXS(17)	ANDP=location of photon production angular distributions			
JXS(18)	LDLWP=location of table of photon production energy distribution locators			
JXS(19)	DLWP=location of photon production energy distributions			
JXS(20)	YP=location of table of yield multipliers			
JXS(21)	FIS=location of total fission cross section			
JXS(22)	END=location of last word of this table	END=location of last word of this table		
JXS(23)	LUNR=location of probability tables			
JXS(24)	DNU=location of delayed nubar data			
JXS(25)	BDD=location of basic delayed data (λ's, probabilities)			

Table F.2 (Cont.)
Definition of the JXS Array

NTY	1 or 2 Continuous Energy or Discrete Reaction Neutron	3 Dosimetry	4 Thermal	5 Continuous Energy Photoatomic
JXS(26)	DNEDL=location of table of energy distribution locators			
JXS(27)	DNED=location of energy distributions			
......				
JXS(32)				

Notes: Many variables are not used, allowing for easy expansion in the future.
All pointers in the JXS array refer to locations in the XSS array.
JXS(1) always points to the first entry in the second block of data.

Table F.3
Definition of Miscellaneous Variables on Data Tables

HZ—10 character name (ZAID) of table. The form of HZ is

ZZZAAA.nnC	continuous-energy neutron
ZZZAAA.nnD	discrete-reaction neutron
ZZZAAA.nnY	dosimetry
XXXXXX.nnT	thermal S(α, β)
ZZZ000.nnP	continuous-energy photoatomic
ZZZ000.nnM	neutron multigroup
ZZZ000.nnG	photoatomic multigroup
ZZZ000.nnE	continuous-energy electron
ZZZAAA.nnU	continuous-energy photonuclear

where ZZZ is the atomic number
AAA is the mass number
XXXXXX for thermal data is a Hollerith name or abbreviation of the material
nn is the evaluation identifier

AW(0)—atomic weight ratio; the atomic weight divided by the mass of a neutron
TZ—temperature at which the data were processed (in MeV)
HD—10-character date when data were processed
HK—70-character comment

Table F.3 (Cont.)
Definition of Miscellaneous Variables on Data Tables

HM—10-character MAT identifier
(IZ(I),AW(I), I=1,16)—16 pairs of ZZZAAAs and atomic weight ratios. In the past these were needed for photoatomic tables but are now ignored. The IZ entries are still needed for thermal tables to indicate for which isotope(s) the scattering data are appropriate.

B. *Locating Data on a Type 2 Table*

A standard unformatted file consists of many records, each with NER entries, where NER is the number of entries per record defined on XSDIR. A Type 2 data table consists of one record that contains pointers, counters, and character information, followed by one or more records containing the XSS array.

The information contained in the first record for each table is the same as that contained in the first twelve lines of a Type 1 table described above. The variables, in order, are HZ; AW(0); TZ; HD; HK; HM; (IZ(I), AW(I), I=1,16); (NXS(I), I=1,16); and (JXS(I), I=1,32). The variables are defined in Tables F.1–F.3. HZ, HD, and HM are 10-character variables and HK is a 70-character variable. Floating-point variables may be double precision in some cases. The number of words contained in this "package" of information is therefore different for different computing systems. The remainder of the first record is empty. The next NREC records (NREC ≥ 1) contain the XSS data array, with NREC=(ITL+NER−1)/NER, where ITL is the table length. A Type 2 library is shown in Figure F-2.

Address	Contents	
$IRN_1 = 1$	misc. including NXS_1, JXS_1	
2	XSS_1	$NER < ITL_1 \leq 2*NER$
3	XSS_1 (cont)	
$IRN_2 = 4$	misc. including NXS_2, JXS_2	
5	XSS_2	$ITL_2 \leq NER$
.	.	.
.	.	
$IRN_n = MAX–3$	misc. including NXS_n, JXS_n	
MAX–2	XSS_n	
MAX–1	XSS_n (cont)	$2*NER < ITL_n \leq 3*NER$
MAX	XSS_n (cont)	

(Records per table are examples only)

n=number of tables contained in library

MAX=number of records contained in library

IRN_i, ITL_i, NER are the addresses, table lengths, and entries per record from XSDIR

Figure F-2. Layout of a Type 2 Library

C. Locating Data Tables in MCNP

The NXS and JXS arrays exist in MCNP for each data table. The information contained in the (2-dimensional) arrays in MCNP mirrors the information contained in NXS and JXS (1-dimensional) on the individual tables. The current dimensions are NXS(16) and JXS(32) on the data tables and NXS(16,∞) and JXS(32,∞) in MCNP, where ∞ indicates variable dimensioning. In the code, the arrays are usually referenced as NXS(I,IEX) and JXS(I,IEX), where IEX is the index to a particular table.

The data from all cross-section tables used in an MCNP problem are in the XSS array, a part of a dynamically allocated common. The data from the first table appear first, followed by the data from the second table, etc., as shown in Figure F-3. The pointers in the JXS array indicate absolute locations in the XSS array.

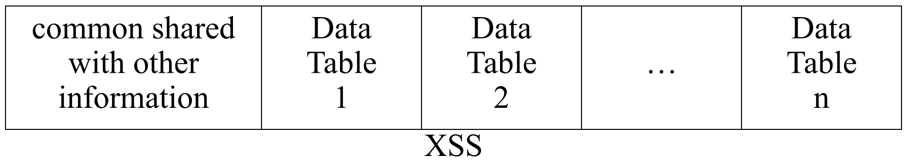

Figure F-3. Diagram of Data Storage in MCNP

The definitions of the variables in the NXS and JXS arrays (Table F.1 and Table F.2) are the same in MCNP as on a data table with one exception. For discrete-reaction neutron tables, NXS(16,IEX) is used in MCNP as an indicator of whether discrete tables in a problem have cross sections tabulated on identical energy grids. Although the definitions of the variables are the same, the contents are generally not. Pointers in the JXS array are pointing to locations in the MCNP internal XSS array that are different from the locations in the data table XSS array. Flags in the NXS array will generally retain the same value in MCNP. Counters in the NXS array may retain the same value, primarily depending on the degree to which MCNP is able to expunge data for a particular problem.

D. Individual Data Blocks

Several blocks of data exist for every cross-section table. The format of an individual block is essentially the same in MCNP as on a data table. In either case, the absolute location of a data block in the XSS array is determined by pointers in the JXS array. The specific data blocks available for a particular table are a function of the *class* of data. We next describe the detailed format of individual data blocks for each *class* of data.

IV. DATA BLOCKS FOR CONTINUOUS/DISCRETE NEUTRON TRANSPORT TABLES

The format of individual data blocks found on neutron transport tables is identical for continuous-energy (NTY=1) and discrete-reaction (NTY=2) tables. Therefore, the format for both are described in this section. All data blocks are now listed with a brief description of their contents and the table numbers in which their formats are detailed.

****Note**: In the tables that follow these descriptions, it is understood that NXS(I) or JXS(I) really means NXS(I,IEX) or JXS(I,IEX) when locating data blocks in MCNP.

1. ESZ Block—contains the main energy grid for the table and the total, absorption, and elastic cross sections as well as the average heating numbers. The ESZ Block always exists. See Table F.4.
2. NU Block—contains prompt, delayed and/or total $\bar{\nu}$ as a function of incident neutron energy. The NU Block exists only for fissionable isotopes (that is, if JXS(2) $\neq$ 0). See Table F.5.
3. MTR Block—contains a list of ENDF/B MT numbers for all neutron reactions other than elastic scattering. The MTR Block exists for all isotopes that have reactions other than elastic scattering (that is, all isotopes with NXS(4) $\neq$ 0). See Table F.6.
4. LQR Block—contains a list of kinematic Q-values for all neutron reactions other than elastic scattering. The LQR Block exists if NXS(4) $\neq$ 0. See Table F.7.
5. TYR Block—contains information about the type of reaction for all neutron reactions other than elastic scattering. Information for each reaction includes the number of secondary neutrons and whether secondary neutron angular distributions are in the laboratory or center-of-mass system. The TYR Block exists if NXS(4) $\neq$ 0. See Table F.8.
6. LSIG Block—contains a list of cross-section locators for all neutron reactions other than elastic scattering. The LSIG Block exists if NXS(4) $\neq$ 0. See Table F.9.
7. SIG Block—contains cross sections for all reactions other than elastic scattering. The SIG Block exists if NXS(4) $\neq$ 0. See Table F.10.
8. LAND Block—contains a list of angular-distribution locators for all reactions producing secondary neutrons. The LAND Block always exists. See Table F.11.
9. AND Block—contains angular distributions for all reactions producing secondary neutrons. The AND Block always exists. See Table F.12.
10. LDLW Block—contains a list of energy distribution locators for all reactions producing secondary neutrons except for elastic scattering. The LDLW Block exists if NXS(5) $\neq$ 0. See Table F.13.
11. DLW Block—contains energy distributions for all reactions producing secondary neutrons except for elastic scattering. The DLW Block exists if NXS(5) $\neq$ 0. See Table F.14.
12. GPD—contains the total photon production cross section tabulated on the ESZ energy grid and a 30x20 matrix of secondary photon energies. The GPD Block exists only for those older evaluations that provide coupled neutron/photon information (that is, if JXS(12) $\neq$ 0). See Table F.15.
13. MTRP Block—contains a list of MT numbers for all photon production reactions. (We will use the term "photon production reaction" for any information describing a specific neutron-in photon-out reaction.) The MTRP Block exists if NXS(6) $\neq$ 0. See Table F.6.
14. LSIGP Block—contains a list of cross-section locators for all photon production reactions. The LSIGP Block exists if NXS(6) $\neq$ 0. See Table F.9.

15. SIGP Block—contains cross sections for all photon production reactions. The SIGP Block exists if NXS(6) ≠ 0. See Table F.16.
16. LANDP Block—contains a list of angular-distribution locators for all photon production reactions. The LANDP Block exists if NXS(6) ≠ 0. See Table F.17.
17. ANDP Block—contains photon angular distributions for all photon production reactions. The ANDP Block exists if NXS(6) ≠ 0. See Table F.18.
18. LDLWP Block—contains a list of energy-distribution locators for all photon production reactions. The LDLWP Block exists if NXS(6) ≠ 0. See Table F.13.
19. DLWP Block—contains photon energy distributions for all photon production reactions. The DLWP Block exists if NXS(6) ≠ 0. See Table F.14.
20. YP Block—contains a list of MT identifiers of neutron reaction cross sections required as photon production yield multipliers. The YP Block exists if NXS(6) ≠ 0. See Table F.19.
21. FIS Block—contains the total fission cross section tabulated on the ESZ energy grid. The FIS Block exists if JXS(21) ≠ 0. See Table F.20.
22. UNR Block—contains the unresolved resonance range probability tables. The UNR block exists if JXS(23) ≠ 0. See Table F.21.

Table F.4
ESZ Block

Location in XSS	Parameter	Description
JXS(1)	E(I), I=1,NXS(3)	Energies
JXS(1)+NXS(3)	σ_t(I), I=1,NXS(3)	Total cross sections
JXS(1)+2*NXS(3)	σ_a(I), I=1,NXS(3)	Total absorption cross sections
JXS(1)+3*NXS(3)	σ_{el}(I), I=1,NXS(3)	Elastic cross sections
JXS(1)+4*NXS(3)	H_{ave}(I), I=1,NXS(3)	Average heating numbers

Table F.5
NU Block

There are four possibilities for the NU Block:

1. JXS(2)=0	no NU Block
2. XSS(JXS(2))>0	either prompt $\bar{\nu}$ or total $\bar{\nu}$ is given. The NU array begins at location XSS(KNU) where KNU=JXS(2).
3. XSS(JXS(2))<0	both prompt $\bar{\nu}$ and total $\bar{\nu}$ are given. The prompt NU Array begins at XSS(KNU) where KNU=JXS(2)+1; the total NU array begins at XSS(KNU), where KNU=JXS(2)+ABS(XSS(JXS(2)))+1.
4. JXS(24)>0	delayed $\bar{\nu}$ is given. The $\bar{\nu}$ array begins at XSS(KNU) where KNU=JXS(24). Delayed $\bar{\nu}$ data must be given in form b).

Table F.5 (Cont.)
NU Block

The NU array has two forms if it exists:

a) Polynomial function form of NU array

Location in XSS	Parameter	Description
KNU	LNU=1	Polynomial function flag
KNU+1	NC	Number of coefficients
KNU+2	C(I), I=1,NC	Coefficients

$$\bar{\nu}(E) = \sum_{I=1}^{NC} C(I)*E^{I-1} \quad E \text{ in MeV}$$

b) Tabular data form of NU array

Location in XSS	Parameter	Description
KNU	LNU=2	Tabular data flag
KNU+1	NR	Number of interpolation regions
KNU+2	NBT(I), I=1,NR	ENDF interpolation parameters
KNU+2+NR	INT(I), I=1,NR	If NR=0, NBT and INT are omitted and linear-linear interpolation is used.
KNU+2+2*NR	NE	Number of energies
KNU+3+2*NR	E(I), I=1,NE	Tabular energy points
KNU+3+2*NR+NE	$\bar{\nu}$ (I), I=1,NE	Corresponding values of $\bar{\nu}$

If delayed $\bar{\nu}$ data exist, the precursor distribution format is given below. The energy distribution for delayed fission neutrons is given by data that follows the format in Table F.13 and Table F.14, where LED=JXS(26) and LDIS=JXS(27).

JXS(25)	DEC_1	Decay constant for this group
JXS(25)+1	NR	Number of interpolation regions
JXS(25)+2	NBT(I), I=1,NR	ENDF interpolation parameters
JXS(25)+2+NR	INT(I), I=1,NR	If NR=0, NBT and INT are omitted and linear-linear interpolation is used.
JXS(25)+2+2*NR	NE	Number of energies
JXS(25)+3+2*NR	E(I), I=1,NE	Tabular energy points
JXS(25)+3+2*NR+NE	P(I), I=1,NE	Corresponding probabilities
JXS(25)+3+2*NR+2NE	DEC_2	Decay constant for this group
.		
.		

Table F.6
MTR, MTRP Blocks

Location in XSS	Parameter	Description
LMT	MT_1	First ENDF reaction available
LMT+1	MT_2	Second ENDF reaction available
.	.	.
.	.	.
.	.	.
LMT+NMT−1	MT_{NMT}	Last ENDF reaction available

where LMT=JXS(3) for MTR Block
LMT=JXS(13) for MTRP Block
NMT=NXS(4) for MTR Block
NMT=NXS(6) for MTRP Block		

Note: For MTR Block: MT_1, MT_2, ... are standard ENDF MT numbers, that is, MT=16=(n,2n); MT=17=(n,3n); etc.
For MTRP Block: the MT values are somewhat arbitrary. To understand the scheme used for numbering the photon production MTs, it is necessary to realize that in ENDF/B format, more than one photon can be produced by a particular neutron reaction that is itself specified by a single MT. Each of these photons is produced with an individual energy-dependent cross section. For example, MT 102 (radiative capture) might be responsible for 40 photons, each with its own cross section, angular distribution, and energy distribution. We need 40 photon MTs to represent the data; the MTs are numbered 102001, 102002, ... , 102040. Therefore, if ENDF/B MT “N” is responsible for “M” photons, we shall number the photon MTs 1000*N+1, 1000*N+2, ... , 1000*N+M.

Table F.7
LQR Block

Location in XSS	Parameter	Description
JXS(4)	Q_1	Q-value of reaction MT_1
JXS(4)+1	Q_2	Q-value of reaction MT_2
.	.	.
.	.	.
.	.	.
JXS(4)+NXS(4)−1	$Q_{NXS(4)}$	Q-value of reaction $MT_{NXS(4)}$

Note: The MT_i’s are given in the MTR Block.

Table F.8
TYR Block

Location in XSS	Parameter	Description
JXS(5)	TY_1	Neutron release for reaction MT_1
JXS(5)+1	TY_2	Neutron release for reaction MT_2
.	.	.
.	.	.
.	.	.
JXS(5)+NXS(4)–1	$TY_{NXS(4)}$	Neutron release for reaction $MT_{NXS(4)}$

Notes: The possible values of TY_i are ±1, ±2, ±3, ±4, 19, 0 and integers greater than 100 in absolute value. The sign indicates the system for scattering: negative = CM system; positive = LAB system. Thus if TY_i = +3, three neutrons are released for reaction MT_i, and the data on the cross-section tables used to determine the exiting neutrons' angles are given in the LAB system.
TY_i=19 indicates fission. The number of secondary neutrons released is determined from the fission $\bar{\nu}$ data found in the NU Block.
TY_i=0 indicates absorption (ENDF reactions MT > 100); no neutrons are released.
$|TY_i| > 100$ signifies reactions other than fission that have energy–dependent neutron multiplicities. The number of secondary neutrons released is determined from the yield data found in the DLW Block.The MT_i's are given in the MTR Block.

Table F.9
LSIG, LSIGP Blocks

Location in XSS	Parameter	Description
LXS	$LOCA_1$=1	Location of cross sections for reaction MT_1
LXS+1	$LOCA_2$	Location of cross sections for reaction MT_2
.	.	.
.	.	.
.	.	.
LXS+NMT–1	$LOCA_{NMT}$	Location of cross sections for reaction MT_{NMT}

where LXS=JXS(6) for LSIG Block
LXS=JXS(14) for LSIGP Block
NMT=NXS(4) for LSIG Block
NMT=NXS(6) for LSIGP Block		

Note: All locators are relative to JXS(7) for LSIG or JXS(15) for LSIGP. The MT_i's are given in the MTR Block for LSIG or the MTRP Block for LSIGP. LOCA$-i$ values must be monotonically increasing or data will be overwritten in subroutine EXPUNG.

Table F.10
SIG Block

Location in XSS	Description
JXS(7)+LOCA$_1$–1	Cross-section array* for reaction MT$_1$
JXS(7)+LOCA$_2$–1	Cross-section array* for reaction MT$_2$
.	.
.	.
.	.
JXS(7)+LOCA$_{NXS(4)}$–1	Cross-section array* for reaction MT$_{NXS(4)}$

*The i^{th} array has the form:

Location in XSS	Parameter	Description
JXS(7)+LOCA$_i$–1	IE$_i$	Energy grid index for reaction MT$_i$
JXS(7)+LOCA$_i$	NE$_i$	Number of consecutive entries for MT$_i$
JXS(7)+LOCA$_i$+1	σ_i[E(K)],K=IE$_i$, IE$_i$+NE$_i$–1	Cross sections for reaction MT$_i$

Note: The values of LOCA$_i$ are given in the LSIG Block. The energy grid E(K) is given in the ESZ Block. The energy grid index IE$_i$ corresponds to the first energy in the grid at which a cross section is given. The MT$_i$'s are defined in the MTR Block.

Table F.11
LAND Block

Location in XSS	Parameter	Description
		Location of angular distribution data for:
JXS(8)	LOCB$_1$=1	elastic scattering
JXS(8)+1	LOCB$_2$	reaction MT$_1$
.	.	.
.	.	.
.	.	.
JXS(8)+NXS(5)	LOCB$_{NXS(5)+1}$	reaction MT$_{NXS(5)}$

Note: All locators (LOCB$_i$) are relative to JXS(9). If LOCB$_i$=0, no angular distribution data are given for this reaction, and isotropic scattering is assumed in either the LAB or CM system. Choice of LAB or CM system depends upon value for this reaction in the TYR Block. The MT$_i$'s are given in the MTR Block.
If LOCB$_i$ = –1, no angular distribution data are given for this reaction in the AND Block. Angular distribution data are specified through LAW$_i$=44 in the DLW Block.
The LOCB$_i$ locators must be monotonically increasing or data will be overwritten in subroutine EXPUNG.

Table F.12
AND Block

Location in XSS	Description
JXS(9)+LOCB$_1$–1	Angular distribution array* for elastic scattering
JXS(9)+LOCB$_2$–1	Angular distribution array* for reaction MT$_1$
.	.
.	.
.	.
JXS(9)+LOCB$_{NXS(5)+1}$–1	Angular distribution array* for reaction MT$_{NXS(5)}$

Note: The values of LOCB$_i$ are given in the LAND Block. If LOCB$_i$ = 0, no angular distribution array is given and scattering is isotropic in either the LAB or CM system. Choice of LAB or CM system depends on value in the TYR Block. The MT$_i$'s are given in the MTR Block.

*The i^{th} array has the form:

Location in XSS	Parameter	Description
JXS(9)+LOCB$_i$–1	NE	Number of energies at which angular distributions are tabulated.
JXS(9)+LOCB$_i$	E(J),J=1,NE	Energy grid
JXS(9)+LOCB$_i$+NE	LC(J),J=1,NE	Location of tables* associated with energies E(J)
	If LC(J) is positive, it points to a 32 equiprobable bin distribution. If LC(J) is negative, it points to a tabular angular distribution. If LC(J)=0, isotropic and no further information is needed.	
*The J^{th} array for a 32 equiprobable bin distribution has the form:		
JXS(9)+|LC(J)|–1	P(1,K),K=1,33	32 equiprobable cosine bins for scattering at energy E(1)
*The J^{th} array for a tabular angular distribution has the form:		
JXS(9)+|LC(J)|–1 is now defined to be:		
LDAT(K+1)	JJ	Interpolation flag: 1=histogram, 2=lin-lin
LDAT(K+2)	NP	Number of points in the distribution
LDAT(K+3)	CSOUT(I), I=1,NP	Cosine scattering angular grid
LDAT(K+3+NP)	PDF(I), I=1,NP	Probability density function
LDAT(K+3+2*NP)	CDF(I), I=1,NP	Cumulative density function

Note: All values of LC(J) are relative to JXS(9). If LC(J) = 0, no table is given for energy E(J) and scattering is isotropic in the coordinate system indicated by entry in the TYR Block.

Table F.13
LDLW, LDLWP Block

Location in XSS	Parameter	Description
LED	$LOCC_1$	Location of energy distribution data for reaction MT_1 or group 1 if delayed neutron
LED+1	$LOCC_2$	Location of energy distribution data for reaction MT_2 or group 2 if delayed neutron
.	.	.
.	.	.
LED+NMT–1	$LOCC_{NMT}$	Location of energy distribution data for reaction MT_{NMT} or group NMT if delayed neutron
where LED=JXS(10) for LDLW Block		NMT=NXS(5) for LDLW Block
LED=JXS(18) for LDLWP Block		NMT=NXS(6) for LDLWP Block
LED=JXS(26) for delayed neutron		NMT=NXS(8) for delayed neutrons

Note: All locators are relative to JXS(11) for LDLW or JXS(19) for LDLWP. The MT_i's are given in the MTR Block for LDLW or MTRP Block for LDLWP. The $LOCC_i$ locators must be monotonically increasing or data will be overwritten in subroutine EXPUNG. For delayed neutrons, the $LOCC_i$ values are relative to JXS(27).

Table F.14
DLW, DLWP Block

Location in XSS	Description
$JED+LOCC_1–1$	Energy distribution array* for reaction MT_1
$JED+LOCC_2–1$	Energy distribution array* for reaction MT_2
.	.
.	.
.	.
$JED+LOCC_{NMT}-1$	Energy distribution array* for reaction MT_{NMT}
where JED=JXS(11) for DLW	
JED=JXS(19) for DLWP	
NMT=NXS(5) for DLW	
NMT=NXS(6) for DLWP	

Note: Values of $LOCC_i$ are given in the LDLW and LDLWP Blocks. Values of MT_i are given in the MTR and MTRP Blocks.

*The i^{th} array has the form:

Location in XSS	Parameter	Description
LDIS+LOCC$_i$–1	LNW$_1$	Location of next law. If LNW$_i$=0, then LAW$_1$ is used regardless of other circumstances.
LDIS+LOCC$_i$	LAW$_1$	Name of this law
LDIS+LOCC$_i$+1	IDAT$_1$	Location of data for this law relative to LDIS
LDIS+LOCC$_i$+2	NR	Number of interpolation regions to define law applicability regime
LDIS+LOCC$_i$+3	NBT(I), I=1,NR	ENDF interpolation parameters.
LDIS+LOCC$_i$+3+NR	INT(I), I=1,NR	If NR=0, NBT and INT are omitted and linear-linear interpolation is used.
LDIS+LOCC$_i$+3+2*NR	NE	Number of energies
LDIS+LOCC$_i$+4+2*NR	E(I), I=NE	Tabular energy points
LDIS+LOCC$_i$+4+2*NR+NE	P(I), I=1,NE	Probability of law validity. If the particle energy E is E<E(1), then P(E)=P(1). If E>E(NE), then P(E)=P(NE). If more than one law is given, then LAW$_1$ is used only if ξ < P(E) where ξ is a random number between 0 and 1.
LDIS+IDAT$_1$–1	LDAT(I), I=1,L**	Law data array for LAW$_1$. The length L of the law data array LDAT is determined from parameters within LDAT. The various law data arrays LDAT for each law LAW$_i$ are given in the following tables.
LDIS+LNW$_1$–1	LNW$_2$	Location of next law
LDIS+LNW$_1$	LAW$_2$	Name of this law
LDIS+LNW$_1$+1	IDAT$_2$	Location of data for this law
.	.	.
.	.	.
.	.	.

where LDIS=JXS(11) for DLW
LDIS=JXS(19) for DLWP
LDIS=JXS(27) for delayed neutrons		

Note: The locators LOCC$_i$ are defined in the LDLW Block or the LDLWP Block. All locators (LNW$_i$, IDAT$_i$) are relative to LDIS.

**We now define the format of the LDAT array for each law. Laws 2 and 4 are used to describe the spectra of secondary photons from neutron collisions. All laws except for Law 2 are used to describe the spectra of scattered neutrons. In the following tables we provide relative locations of data in the LDAT array rather than absolute locations in the XSS array. The preceding table defines the starting location of the LDAT array within the XSS array.

a. LAW$_i$=1 Tabular Equiprobable Energy Bins (From ENDF Law 1)

Location	Parameter	Description

LDAT(1)
LDAT(2)
LDAT(2+NR)
	NR
	NBT(I), I=1,NR
	INT(I), I=1,NR
		Interpolation scheme between tables of E$_{out}$. If NR=0 or if INT(I) ¦1 (histogram), linear-linear interpolation is used

LDAT(2+2*NR)
LDAT(3+2*NR)
	NE
	E$_{in}$(I), I=1,NE
		Number of incident energies tabulated
		List of incident energies for which E$_{out}$ is tabulated

LDAT(3+2*NR+NE)	NET	Number of outgoing energies in each E$_{out}$ table
LDAT(4+2*NR+NE)	E_{out_1} (I), I=1,NET E_{out_2} (I), I=1,NET $E_{out_{NE}}$ (I), I=1,NET	E$_{out}$ tables are NET boundaries of NET−1 equally likely energy intervals. Linear-linear interpolation is used between intervals

b. LAW$_i$ = 2 Discrete Photon Energy

Location	Parameter	Description
LDAT(1)	LP	Indicator of whether the photon is a primary or nonprimary photon
LDAT(2)	EG	Photon energy (if LP=0 or LP=1), or Binding energy (if LP=2)
If LP=0 or LP=1, the photon energy is EG		
If LP=2, the photon energy is EG+(AWR)/(AWR+1)*EN, where AWR is the atomic weight ratio and EN is the incident neutron energy		

c. LAW$_i$ = 3 Level Scattering (From ENDF Law 3)

$$LDAT(1) = \left(\frac{A+1}{A}\right) |Q| \quad LDAT(2) = \left(\frac{A}{A+1}\right)^2$$

$$E_{out}^{CM} = LDAT(2) * (\mathrm{E} - \mathrm{LDAT(1)})$$

where E_{out}^{CM}	=	outgoing center-of-mass energy
E	=	incident energy
A	=	atomic weight ratio
Q	=	Q-value

The outgoing neutron energy in the laboratory system, E_{out}^{LAB} , is

$$E_{out}^{LAB} = E_{out}^{CM} + \left\{ E + 2\mu_{cm}(A+1)(EE_{out}^{CM})^{1/2} \right\} / (A+1)^2 \ ,$$

where μ_{cm} = cosine of the center-of-mass scattering angle.

d. LAW$_i$=4 Continuous Tabular Distribution (From ENDF Law 1)

Location	Parameter	Description
LDAT(1)	NR	Number of interpolation regions
LDAT(2)	NBT(I), I=1,NR	ENDF interpolation parameters. If NR=0, NBT and INT are omitted and linear-linear interpolation is used.
LDAT(2+NR)	INT(I), I=1,NR	
LDAT(2+2*NR)	NE	Number of energies at which distributions are tabulated
LDAT(3+2*NR)	E(I), I=1,NE	Incident neutron energies
LDAT(3+2*NR+NE)	L(I), I=1,NE	Locations of distributions (relative to JXS(11) or JXS(19))
Data for E(1) (let K=3+2*NR+2*NE):		
LDAT(K)	INTT′	Combination of the number of discrete photon lines, ND, and the interpolation scheme for subsequent data, INTT=1 histogram distribution INTT=2 linear-linear distribution
LDAT(K+1)	NP	Number of points in the distribution
LDAT(K+2)	EOUT(I), I=1,NP	Outgoing energy grid
LDAT(K+2+NP)	PDF(I), I=1,NP	Probability density function
LDAT(K+2+2*NP)	CDF(I), I=1,NP	Cumulative density function
Data for E(2):		
.	.	.
.	.	.

If the value of LDAT(K) is INTT′ > 10, then

$$\text{INTT}' = (\text{ND}*10) + \text{INTT}$$

where INTT is the interpolation scheme and the first *ND* values of NP points describe discrete photon lines. The remaining *NP* – *ND* values describe a continuous distribution. In this way the distribution may be discrete, continuous, or a discrete distribution superimposed upon a continuous background.

e. LAW$_i$=5 General Evaporation Spectrum (From ENDF–6 File 5 LF=5)

Location	Parameter	Description
LDAT(1)	NR	
LDAT(2)	NBT(I), I=1,NR	Interpolation scheme between T’s
LDAT(2+NR)	INT(I), I=1,NR	
LDAT(2+2*NR)	NE	Number of incident energies tabulated
LDAT(3+2*NR)	E(I), I=1,NE	Incident energy table
LDAT(3+2*NR+NE)	T(I), I=1,NE	Tabulated function of incident energies
LDAT(3+2*NR+2*NE)	NET	Number of X’s tabulated
LDAT(4+2*NR+2*NE)	X(I), I=1,NET	Tabulated probabilistic function
E_{out} = X(ξ)*T(E), where X(ξ) is a randomly sampled table of X's, and E is the incident energy.		

f. LAW$_i$=7 Simple Maxwell Fission Spectrum (From ENDF–6 File 5 LF=7)

Location	Parameter	Description
LDAT(1)	NR	
LDAT(2)	NBT(I), I=1,NR	Interpolation scheme between T’s
LDAT(2+NR)	INT(I), I=1,NR	
LDAT(2+2*NR)	NE	Number of incident energies tabulated
LDAT(3+2*NR)	E(I), I=1,NE	Incident energy table
LDAT(3+2*NR+NE)	T(I), I=1,NE	Tabulated T’s
LDAT(3+2*NR+2*NE)	U	Restriction energy

$$f(E \rightarrow E_{out}) = C\sqrt{E_{out}}\; e^{-E_{out}/T(E)}$$

with restriction $0 \le E_{out} \le E - U$

$$C = T^{-3/2}\left[\frac{\sqrt{\pi}}{2} erf(\sqrt{(E-U)/T}) + -\sqrt{(E-U)/T}\;\; e^{-(E-U)/T}\right]^{-1}$$

g. LAW$_i$=9 Evaporation Spectrum (From ENDF–6 File 5 LF=9)

Location	Parameter	Description
LDAT(1)	NR	
LDAT(2)	NBT(I), I=1,NR	Interpolation scheme between T’s
LDAT(2+NR)	INT(I), I=1,NR	
LDAT(2+2*NR)	NE	Number of incident energies tabulated
LDAT(3+2*NR)	E(I), I=1,NE	Incident energy table

Location	Parameter	Description
LDAT(3+2*NR+NE)	T(I), I=1,NE	Tabulated T's
LDAT(3+2*NR+2*NE)	U	Restriction energy

$$f(E \rightarrow E_{out}) = CE_{out}e^{-E_{out}/T(E)}$$

with restriction $0 \le E_{out} \le E - U$

$$C = T^{-2}\left[1 - e^{(E-U)/T}\ (1 + (E-U)/T)\right]^{-1}$$

h. LAW$_i$=11 Energy Dependent Watt Spectrum (From ENDF–6 File 5 LF=11)

Location	Parameter	Description
LDAT(1)	NR_a	Interpolation scheme between a's
LDAT(2)	NBT_a(I),I=1,NR_a	
LDAT(2+NR_a)	INT_a(I), I=1,NR_a	
LDAT(2+2*NR_a)	NE_a	Number of incident energies tabulated for a(E_{in}) table
LDAT(3+2*NR_a)	E_a(I), I=1,NE_a	Incident energy table
LDAT(3+2*NR_a+NE_a)	a(I), I=1,NE_a	Tabulated a's
let L=3+2*(NR_a+NE_a)		
LDAT(L)	NR_b	Interpolation scheme between b's
LDAT(L+1)	NBT_b(I),I=1,NR_b	
LDAT(L+1+NR_b)	INT_b(I), I=1,NR_b	
LDAT(L+1+2*NR_b)	NE_b	Number of incident energies tabulated for b(E_{in}) table
LDAT(L+2+2*NR_b)	E_b(I), I=1,NE_b	Incident energy table
LDAT(L+2+2*NR_b+NE_b)	b(I), I=1,NE_b	Tabulated b's
LDAT(L+2+2*NR_b+2*NE_b)	U	Rejection energy

$$f(E \rightarrow E_{out}) = C_o exp[-E_{out}/a(E)]sinh[b(E)E_{out}]^{1/2}$$

with restriction $0 \le E_{out} < E - U$

This law is sampled by the rejection scheme in LA-9721-MS.[1]

i. LAW$_i$=22 Tabular Linear Functions (from UK Law 2)

Location in XSS	Parameter	Description
LDAT(1)	NR	Interpolation parameters that are not used by
LDAT(2)	NBT(I), I=1,NR	MCNP
LDAT(2+NR)	INT(I), I=1,NR	(histogram interpolation is assumed)
LDAT(2+2*NR)	NE	
LDAT(3+2*NR)	E_{in}(I), I=1,NE	Number of incident energies tabulated
LDAT(3+2*NR+NE)	LOCE(I), I=1,NE	List of incident energies for E_{out} tables
Data for E_{in}(1) (Let L=3+2*NR+2*NE):		Locators of E_{out} tables (relative to JXS(11))
LDAT(L)	NF_1	if $E_{in}(I)_i$ E < E_{in}(I+1) and ξ is a
LDAT(L+1)	P_1(K),K=1,NF_1	random number [0,1] then if
LDAT(L+1+NF_1)	T_1(K),K=1,NF_1	$$\sum_{k=1}^{k=K} P_I(k) < \xi \le \sum_{k=1}^{k=K} P_I(k)$$
LDAT(L+1+2*NF_1)	C_1(K),K=1,NF_1	
Data for E_{in}(2):		
.	.	$E_{out} = C_I(K)*(E-T_I(K))$

j. LAW$_i$=24 (From UK Law 6)

Location in XSS	Parameter	Description
LDAT(1)	NR	Interpolation parameters that are not used
LDAT(2)	NBT(I), I=1,NR	by MCNP
LDAT(2+NR)	INT(I), I=1,NR	(histogram interpolation is assumed)
LDAT(2+2*NR)	NE	Number of incident energies
LDAT(3+2*NR)	E_{in}(I), I=1,NE	List of incident energies for which T is tabulated
LDAT(3+2*NR+NE)	NET	Number of outgoing values in each table
LDAT(4+2*NR+NE)	T_1(I), I=1,NET T_2(I), I=1,NET . . T_{NE}(I), I=1,NET	Tables are NET boundaries of NET−1 equally likely intervals. Linear-linear interpolation is used between intervals.
$E_{out} = T_K(I)*E$ where T_K(I) is sampled from the above tables E is the incident neutron energy		

k. LAW$_i$=44 Kalbach-87 Formalism (From ENDF File 6 Law 1, LANG=2)

Location	Parameter	Description
LDAT(1)	NR	Number of interpolation regions
LDAT(2)	NBT(I), I=1,NR	ENDF interpolation parameters. If NR=0,
LDAT(2+NR)	INT(I), I=1,NR	NBT and INT are omitted and linear-linear interpolation is used.
LDAT(2+2*NR)	NE	Number of energies at which distributions are tabulated
LDAT(3+2*NR)	E(I), I=1,NE	Incident neutron energies
LDAT(3+2*NR+NE)	L(I), I=1,NE	Locations of distributions (relative to JXS(11) or JXS(19))
Data for E(1) (let K=3+2*NR+2*NE):		
LDAT(K)	INTT′	Interpolation scheme for subsequent data INTT=1 histogram distribution INTT=2 linear-linear distribution
LDAT(K+1)	NP	Number of points in the distribution
LDAT(K+2)	EOUT(I), I=1,NP	Outgoing energy grid
LDAT(K+2+NP)	PDF(I), I=1,NP	Probability density function
LDAT(K+2+2*NP)	CDF(I), I=1,NP	Cumulative density function
LDAT(K+2+3*NP)	R(I), I=1,NP	Precompound fraction r
LDAT(K+2+4*NP)	A(I), I=1,NP	Angular distribution slope value *a*
Data for E(2):		
.	.	.
.	.	.
If the value of LDAT(K) is $INTT' > 10$, then		
	$INTT' = 10 * ND + INTT$	

where INTT is the interpolation scheme and the first *ND* values of *NP* describe discrete photon lines. The remaining $NP - ND$ values describe a continuous distribution. In this way the distribution may be discrete, continuous, or a discrete distribution superimposed upon a continuous background.

The angular distributions for neutrons are then sampled from

$$p(\mu, E_{in}, E_{out}) = \frac{1}{2}\frac{A}{\sinh(A)}[\cosh(A\mu) + R\sinh(A\mu)]$$

as described on page 2-45 in Chapter 2.

l. LAW$_i$=61 Like LAW 44 but tabular angular distribution instead of Kalbach-87

Location	Parameter	Description
LDAT(1)	NR	Number of interpolation regions
LDAT(2)	NBT(I), I=1,NR	ENDF interpolation parameters. If NR=0,
LDAT(2+NR)	INT(I), I=1,NR	NBT and INT are omitted and linear-linear interpolation is used.
LDAT(2+2*NR)	NE	Number of energies at which distributions are tabulated
LDAT(3+2*NR)	E(I), I=1,NE	Incident neutron energies
LDAT(3+2*NR+NE)	L(I), I=1,NE	Locations of distributions (relative to JXS(11) or JXS(19))
Data for E(1) (let K=3+2*NR+2*NE):		

LDAT(K)	INTT′	Interpolation scheme for subsequent data
		INTT=1 histogram distribution
		INTT=2 linear-linear distribution

LDAT(K+1)	NP	Number of points in the distribution
LDAT(K+2)	EOUT(I), I=1,NP	Outgoing energy grid
LDAT(K+2+NP)	PDF(I), I=1,NP	Probability density function
LDAT(K+2+2*NP)	CDF(I), I=1,NP	Cumulative density function
LDAT(K+2+3*NP)	LC(I), I=1,NP	Location of tables* associated with incident energies E(I)
	If LC(I) is positive, it points to a tabular angular distribution. If LC(I)=0=isotropic and no further information is needed. 32 equiprobable bin distribution is not allowed.	
*The J^{th} array for a tabular angular distribution has the form:		
L=JXS(11)+|LC(J)|−1 or JXS(19)+|LC(J)|−1 is now defined to be:		
LDAT(L+1)	JJ	Interpolation flag: 0=histogram, 1=lin-lin
LDAT(L+2)	NP	Number of points in the distribution
LDAT(L+3)	CSOUT(I), I=1,NP	Cosine scattering angular grid
LDAT(L+3+NP)	PDF(I), I=1,NP	Probability density function
LDAT(L+3+2*NP)	CDF(I), I=1,NP	Cumulative density function
Data for E(2):		
.	.	.
.	.	.
If the value of LDAT(K) is $INTT' > 10$, then		
	$INTT' = 10 * ND + INTT$	

m. LAW$_i$=66 N-body phase space distribution (From ENDF File 6 Law 6)

Location	Parameter	Description
LDAT(1)	NPSX	Number of bodies in the phase space
LDAT(2)	A_p	Total mass ratio for the NPSX particles

$$E_{out} = T(\xi) * E_i^{max}$$

where

$$E_i^{max} = \frac{A_p - 1}{A_p}\left(\frac{A}{A+1}E_{in} + Q\right)$$

and $T(\xi)$ is sampled from

$$P_i(\mu, E_{in}, T) = C_n\sqrt{T}(E_i^{max} - T)^{3n/2-4}$$

where the sampling scheme is from R28 of LA-9721-MS[1] and is described on page 2-47 in Chapter 2.

n. LAW$_i$=67 Laboratory Angle–Energy Law (From ENDF File 6 Law 7)

Location	Parameter	Description
LDAT(1)	NR	Number of interpolation regions
LDAT(2)	NBT(I), I=1,NR	ENDF interpolation parameters. If NR=0,
LDAT(2+NR)	INT(I), I=1,NR	NBT and INT are omitted and linear-linear interpolation is used.
LDAT(2+2*NR)	NE	Number of energies at which distributions are tabulated
LDAT(3+2*NR)	E(I), I=1,NE	Incident neutron energies
LDAT(3+2*NR+NE)	L(I), I=1,NE	Locations of distributions (relative to JXS(11) or JXS(19))
Data for E(1) (let K=3+2*NR+2*NE):		
LDAT(K)	INTMU	Interpolation scheme for secondary cosines INTMU=1 histogram distribution INTMU=2 linear-linear distribution
LDAT(K+1)	NMU	Number of secondary cosines
LDAT(K+2)	XMU(I), I=1,NMU	Secondary cosines
LDAT(K+2+NMU)	LMU(I), I=1,NMU)	Location of data for each secondary cosine (relative to JXS(11) or JXS(19))

Location	Parameter	Description
Data for XMU(1) (let J=K+2+2*NMU):		

LDAT(J)	INTEP	Interpolation parameter between secondary energies
		INTEP=1 histogram distribution
		INTEP=2 linear-linear distribution

LDAT(J+1)	NPEP	Number of secondary energies
LDAT(J+2)	EP(I), I=1,NPEP	Secondary energy grid
LDAT(J+2+NPEP)	PDF(I), I=1,NPEP	Probability density function
LDAT(J+2+2*NPEP)	CDF(I), I=1,NPEP	Cumulative density function
Data for XMU(2)		
.		
.		
Data for XMU(NMU)		
.		
.		
Data for E(2)		
.		
.		
Data for E(NE)		
.		
.		

o. Energy–Dependent Neutron Yields

There are additional numbers to be found for neutrons in the DLW array. For those reactions with entries in the TYR block that are greater than 100 in absolute value, there must be neutron yields Y(E) provided as a function of neutron energy. The neutron yields are handled similarly to the average number of neutrons per fission $\bar{\nu}$ (E) that is given for the fission reactions. These yields are a part of the coupled energy–angle distributions given in File 6 of ENDF–6 data.

Location in XSS

JED + $|TY_i|$ – 101 Neutron yield data for reaction MT_i
where JED=JXS(11)=DLW
$i \leq$ number of reactions with negative angular distributions locators

The i^{th} array has the form:

Location in XSS	Parameter	Description
KY	NR	Number of interpolation regions
KY+1	NBT(I), I=1,NR	ENDF interpolation parameters. If NR=0
KY+1+NR	INT(I), I=1,NR	NBT and INT are omitted and linear-linear interpolation is used.
KY+1+2*NR	NE	Number of energies
KY+2+2*NR	E(I), I=1,NE	Tabular energy points
KY+2+2*NR+NE	Y(I), I=1,NE	Corresponding Y(E) values
where KY=JED+$\|TY_i\|$−101		

Table F.15
GPD Block

Location in XSS	Parameter	Description
JXS(12)	σ_γ(I), I=1,NXS(3)	Total photon production cross section
JXS(12)+NXS(3)	EG(1,K),K=1,20	20 equally likely outgoing photon energies for incident neutron energy E < EN(2)
JXS(12)+NXS(3)+20	EG(2,K),K=1,20	20 equiprobable outgoing photon energies for incident neutron energy EN(2) ≤ E < EN(3)
.	.	.
.	.	.
.	.	.
JXS(12)+NXS(3)+580	EG(30,K),K=1,20	20 equiprobable outgoing photon energies for incident neutron energy E ≥ EN(30)

Note: (1) The discrete incident neutron energy array in MeV is EN(J), J=1,30: 1.39E-10, 1.52E-7, 4.14E−7, 1.13E−6, 3.06E−6, 8.32E−6, 2.26E−5, 6.14E−5, 1.67E−4, 4.54E−4, 1.235E−3, 3.35E−3, 9.23E−3, 2.48E−2, 6.76E−2, .184, .303, .500, .823, 1.353, 1.738, 2.232, 2.865, 3.68, 6.07, 7.79, 10., 12., 13.5, 15.
(2) The equiprobable photon energy matrix is used only for those older tables that do not provide expanded photon production data, and no currently–supported libraries use this data.

Table F.16
SIGP Block

Location in XSS	Description
JXS(15)+$LOCA_1$−1	Cross-section array* for reaction MT_1
JXS(15)+$LOCA_2$−1	Cross-section array* for reaction MT_2
.	.
.	.
JXS(15)+$LOCA_{NXS(6)}$−1	Cross-section array* for reaction $MT_{NXS(6)}$

*The i^{th} array has three possible forms, depending on the first word in the array:

(a) If MFTYPE=12 (Yield Data taken from ENDF File 12) or
If MFTYPE=16 (Yield Data taken from ENDF File 6)

Location in XSS	Parameter	Description
JXS(15)+$LOCA_i$−1	MFTYPE	12 or 16
JXS(15)+$LOCA_i$	MTMULT	Neutron MT whose cross section should multiply the yield
JXS(15)+$LOCA_i$+1	NR	Number of interpolation regions
JXS(15)+$LOCA_i$+2	NBT(I), I=1,NR	ENDF interpolation parameters. If NR=0, NBT and INT are omitted and
JXS(15)+$LOCA_i$+2+NR	INT(I), I=1,NR	linear-linear interpolation is used.
JXS(15)+$LOCA_i$+2+2*NR	NE	Number of energies at which the yield is tabulated
JXS(15)+$LOCA_i$+3+2*NR	E(I), I=1,NE	Energies
JXS(15)+$LOCA_i$+3 +2*NR+NE	Y(I), I=1,NE	Yields

$$\sigma_{\gamma,\,i} = Y(E) * \sigma_{MTMULT}(E)$$

(b) If MFTYPE=13 (Cross-Section Data from ENDF File 13)

Location in XSS	Parameter	Description
JXS(15)+$LOCA_i$−1	MFTYPE	13
JXS(15)+$LOCA_i$	IE	Energy grid index
JXS(15)+$LOCA_i$+1	NE	Number of consecutive entries
JXS(15)+$LOCA_i$+2	$\sigma_{\gamma,\,i}[E(K)], K = IE, IE + NE - 1$	Cross sections for reaction MT_i

Note: The values of $LOCA_i$ are given in the LSIGP Block. The energy grid E(K) is given in the ESZ Block. The MT_i's are defined in the MTRP Block.

Table F.17
LANDP Block

Location in XSS	Parameter	Description
JXS(16)	$LOCB_1$=1	Location of angular distribution data for reaction MT_1
JXS(16)+1	$LOCB_2$	Location of angular distribution data for reaction MT_2
.	.	.
.	.	.
.	.	.
JXS(16)+NXS(6) – 1	$LOCB_{NXS(6)}$	Location of angular distribution data for reaction $MT_{NXS(6)}$

Note: All locators ($LOCB_i$) are relative to JXS(17). If $LOCB_i$=0, there are no angular distribution data given for this reaction and isotropic scattering is assumed in the LAB system. MT_i's are defined in the MTRP Block.

Table F.18
ANDP Block

Location in XSS	Description
JXS(17)+$LOCB_1$–1	Angular distribution array* for reaction MT_1
JXS(17)+$LOCB_2$–1	Angular distribution array* for reaction MT_2
JXS(17)+$LOCB_{NXS(6)}$–1	Angular distribution array* for reaction $MT_{NXS(6)}$

Note: The values of $LOCB_i$ are given in the LANDP Block. If $LOCB_i$=0, then no angular distribution array is given and scattering is isotropic in the LAB system. The MT_i's are given in the MTRP Block.

*The i^{th} array has the form:

Location in XSS	Parameter	Description
JXS(17)+$LOCB_i$–1	NE	Number of energies at which angular distributions are tabulated.
JXS(17)+$LOCB_i$	E(J),J=1,NE	Energy grid
JXS(17)+$LOCB_i$+NE	LC(J),J=1,NE	Location of tables associated with energies E(J)
JXS(17)+LC(1)–1	P(1,K),K=1,33	32 equiprobable cosine bins for scattering at energy E(1)
JXS(17)+LC(2)–1	P(2,K),K=1,33	32 equiprobable cosine bins for scattering at energy E(2)
.	.	.
.	.	.
JXS(17)+LC(NE)–1	P(NE,K),K=1,33	32 equiprobable cosine bins for scattering at energy E(NE)

Note: All values of LC(J) are relative to JXS(17). If LC(J)=0, no table is given for energy E(J) and scattering is isotropic in the LAB system.

Table F.19
YP Block

Location in XSS	Parameter	Description
JXS(20)	NYP	Number of neutron MTs to follow
JXS(20)+1	MTY(I), I=1,NYP	Neutron MTs

Note: The MTY array contains all neutron MTs that are required as photon-production yield multipliers (See Table F.16). MCNP needs this information when expunging data.

Table F.20
FIS Block

Location in XSS	Parameter	Description
JXS(21)	IE	Energy grid index
JXS(21)+1	NE	Number of consecutive entries
JXS(21)+2	$\sigma_f[E(K)], K = IE, IE+NE-1$	Total fission cross sections

Note: The FIS Block generally is not provided on individual data tables because the total fission cross section is a redundant quantity [that is, $\sigma_{f,tot}(E) = \sigma_{n,f}(E) + \sigma_{n,n'f}(E) + \sigma_{n,2nf}(E) + \sigma_{n,3nf}(E)$]. MCNP forms the FIS Block if conditions warrant (for example, for KCODE calculations, coupled neutron/photon calculations, etc.). The energy grid E(K) is given in the ESZ Block.

Table F.21
UNR Block

Location in XSS	Parameter	Description
JXS(23)	N	Number of incident energies where there is a probability table
JXS(23)+1	M	Length of table; i.e., number of probabilities, typically 20
JXS(23)+2	INT	Interpolation parameter between tables =2 lin-lin; =5 log-log
JXS(23)+3	ILF	Inelastic competition flag (see below)
JXS(23)+4	IOA	Other absorption flag (see below)
JXS(23)+5	IFF	Factors flag (see below)
JXS(23)+6	E(I), I=1,N	Incident energies
JXS(23)+6+N	P(I,J,K)	Probability tables (see below)

Note: ILF is the inelastic competition flag. If this flag is less than zero, the inelastic cross section is zero within the entire unresolved energy range. If this flag is more than zero, then its value is a special MT number whose tabulation is the sum of the inelastic levels. An exception to this scheme is typically made when there is only one inelastic level within the unresolved energy range, because the flag can then just be set to its MT number and the special tabulation is not needed. The flag can also be set to zero, which means that the sum of the contribution of the inelastic reactions will be made using a balance relationship involving the smooth cross sections.

IOA is the other absorption flag for determining the contribution of "other absorptions" (no neutron out or destruction reactions). If this flag is less than zero, the "other absorption" cross section is zero within the entire unresolved energy range. If this flag is more than zero, then its value is a special MT number whose tabulation is the sum of the "other absorption" reactions. An exception to this scheme is typically made when there is only one "other absorption" reaction within the unresolved energy range, because the flag can then just be set to its MT number and the special tabulation is not needed. The flag can also be set to zero, which means that the sum of the contribution of the "other absorption" reactions will be made using a balanced relationship involving the smooth cross sections.

IFF is the factors flag. If this flag is zero, then the tabulations in the probability tables are cross sections. If the flag is one, the tabulations in the probability tables are factors that must be multiplied by the corresponding "smooth" cross sections to obtain the actual cross sections.

P(I,J,K), where I=1,N, J=1,6 , and K=1,M, are the tables at N incident energies for M cumulative probabilities. For each of these probabilities the J values are:

J	Description
1	cumulative probability
2	total cross section or total factor
3	elastic cross section or elastic factor
4	fission cross section or fission factor
5	(n,γ) cross section or (n,γ) factor
6	neutron heating number or heating factor
The ordering of the probability-table entries is as follows	
	M cumulative probabilities for energy I=1 (K=1 through K=M)
	M total cross sections (or factors) for energy I=1 (K=1 through K=M)
	...
	M cumulative probabilities for energy I=2 (K=1 through K=M)
	...
	M neutron heating numbers (or factors) for energy I=N (K=1 through K=M)

Note: The cumulative probabilities are monotonically increasing from an implied lower value of zero to the upper value of P(I,1,K=M) = 1.0. The total cross section, P(I,2,J), is not used in MCNP; the total is recalculated from sampled partials to avoid round-off error. The (n,γ) cross section is radiative capture only; it is not the usual MCNP "capture" cross section, which is really absorption or destruction with other no-neutron-out reactions.

V. DATA BLOCKS FOR DOSIMETRY TABLES

Dosimetry tables (NTY=3) provide cross sections that are useful as response functions with the FM feature in MCNP. They can never be used for actual neutron transport. Therefore, there is a more limited set of information available on dosimetry tables than on neutron transport tables (NTY=1 or 2). Only three blocks of data exist on dosimetry tables. The three blocks follow, with the table numbers in which their formats are detailed.

1. MTR Block—contains a list of the MT numbers for all reactions provided on the table. The MTR Block always exists on dosimetry tables. The format of the block is identical to that of the MTR Block previously described for neutron transport tables. See Table F.6.

2. LSIG Block—contains a list of cross-section locators for all reactions provided on the table. The LSIG Block always exists on dosimetry tables. The format of the block is identical to that of the LSIG Block previously described for neutron transport tables. See Table F.9.
3. SIGD Block—contains (energy, cross-section) pairs for all reactions provided on the table. The SIGD Block always exists on dosimetry tables. See Table F.22.

Table F.22
SIGD Block

Location in XSS	Description
JXS(7)+LOCA$_1$−1	Cross-section array* for reaction MT$_1$
JXS(7)+LOCA$_2$−1	Cross-section array* for reaction MT$_2$
.	.
.	.
JXS(7)+LOCA$_{NXS(4)}$−1	Cross-section array* for reaction MT$_{NXS(4)}$

*The i^{th} array has the form:

Location in XSS	Parameter	Description
JXS(7)+LOCA$_i$−1	NR	Number of interpolation regions
JXS(7)+LOCA$_i$	NBT(I), I=1,NR	ENDF interpolation parameters. If NR=0,
JXS(7)+LOCA$_i$+NR	INT(I), I=1,NR	NBT and INT are omitted and linear-linear interpolation is assumed.
JXS(7)+LOCA$_i$+2*NR	NE	Number of (energy, cross section) pairs
JXS(7)+LOCA$_i$+1 +2*NR	E(I), I=1,NE	Energies
JXS(7)+LOCA$_i$+1+2*NR+NE	σ(I), I=1,NE	Cross sections

Note: The locators (LOCA$_i$) are provided in the LSIG Block. The MT$_i$'s are given in the MTR Block.

VI. DATA BLOCKS FOR THERMAL S(α,β) TABLES

Data from thermal S(α,β) tables (NTY=4) provide a complete representation of thermal neutron scattering by molecules and crystalline solids. Cross sections for elastic and inelastic scattering are found on the tables (typically for neutron energies below 4 eV). A coupled energy/angle representation is used to describe the spectra of inelastically scattered neutrons. Angular distributions for elastic scattering are also provided.

Four unique blocks of data are associated with S(α,β) tables. We now briefly describe each of the four data blocks and give the table numbers in which their formats are detailed.

1. ITIE Block—contains the energy-dependent inelastic scattering cross sections. The ITIE Block always exists. See Table F.23.

2. ITCE Block—contains the energy-dependent elastic scattering cross sections. The ITCE Block exists if JXS(4) $\neq$ 0. See Table F.24.
3. ITXE Block—contains coupled energy/angle distributions for inelastic scattering. The ITXE Block always exists. See Table F.25.
4. ITCA Block—contains angular distributions for elastic scattering.
   The ITCA Block exists if JXS(4) $\neq$ 0 and NXS(6) $\neq -1$. See Table F.26.

Table F.23
ITIE Block

Location in XSS	Parameter	Description
JXS(1)	NE_{in}	Number of inelastic energies
JXS(1)+1	$E_{in}(I)$, I=1,NE_{in}	Energies
JXS(1)+1+NE_{in}	$\sigma_{in}(I)$, I=1,NE_{in}	Inelastic cross sections

Note: JXS(2)=JXS(1)+1+NE_{in} . Linear-linear interpolation is assumed between adjacent energies.

Table F.24
ITCE Block

Location in XSS	Parameter	Description
JXS(4)	NE_{el}	Number of elastic energies
JXS(4)+1	$E_{el}(I)$, I=1,NE_{el}	Energies
JXS(4)+1+NE_{el}	P(I), I=1,NE_{el}	(See below)
If NXS(5) $\neq$ 4: $\sigma_{el}(I)$=P(I), with linear-linear interpolation between points		
If NXS(5)=4: $\sigma_{el}(E)$=P(I)/E, for $E_{el}(I)_i < E < E_{el}(I+1)$		

Note: JXS(5)=JXS(3)+1+NE_{el}

Table F.25
ITXE Block

For NXS(2)=3 (equally-likely cosines; currently the only scattering mode allowed for inelastic angular distributions)

Location in XSS	Parameter	Description
JXS(3)	$E_1^{OUT}[E_{in}(1)]$	First of NXS(4) equally-likely outgoing energies for inelastic scattering at $E_{in}(1)$
JXS(3)+1	$\mu_I(1 \rightarrow 1)$, I=1,NXS(3)+1	Equally-likely discrete cosines for scattering from $E_{in}(1)$ to $E_1^{OUT}[E_{in}(1)]$
JXS(3)+2+NXS(3)	$E_2^{OUT}[E_{in}(1)]$	Second of NXS(4) equally-likely outgoing energies for inelastic scattering at $E_{in}(1)$
JXS(3)+3+NXS(3)	$\mu_I(1 \rightarrow 2)$, I=1,NXS(3)+1	Equally-likely discrete cosines for scattering from $E_{in}(1)$ to $E_2^{OUT}[E_{in}(1)]$
.	.	.
.	.	.
JXS(3)+(NXS(4)–1)*(NXS(3)+2)	$E_{NXS(4)}^{OUT}[E_{in}(1)]$	Last of NXS(4) equally-likely outgoing energies for inelastic scattering at $E_{in}(1)$
JXS(3)+(NXS(4)–1)*(NXS(3)+2)+1	$\mu_I(1 \rightarrow NXS(4))$, I=1,NXS(3)+1	Equally-likely discrete cosines for scattering from $E_{in}(1)$ to $E_{NXS(4)}^{OUT}[E_{in}(1)]$
.	.	.
(Repeat for all remaining values of E_{in})		.
.	.	.

Note: Incident inelastic energy grid $E_{in}(I)$ is given in ITIE Block. Linear-linear interpolation is assumed between adjacent values of E_{in}.

Table F.26
ITCA Block

Location in XSS	Parameter	Description
JXS(6)	$\mu_I[E_{el}(1)]$, I=1,NXS(6)+1	Equally-likely discrete cosines for elastic scattering at $E_{el}(1)$
JXS(6)+NXS(6)+1	$\mu_I[E_{el}(2)]$, I=1,NXS(6)+1	Equally-likely discrete cosines for elastic scattering at $E_{el}(2)$
.	.	.
.	.	.
JXS(6)+(NE_{el}–1)*(NXS(6)+1)	$\mu_I[E_{el}(NE_{el})]$, I=1,NXS(6)+1	Equally-likely discrete cosines for elastic scattering at $E_{el}(NE_{el})$

Note: Incident elastic energy grid $E_{el}(I)$ and number of energies NE_{el} are given in ITCE Block. Linear-linear interpolation is assumed between adjacent values of E_{el}.

VII. DATA BLOCKS FOR PHOTOATOMIC TRANSPORT TABLES

Ten data blocks are found on photoatomic transport tables (NTY=5). Information contained on the blocks includes: cross sections for coherent and incoherent scattering, pair production, and the photoelectric effect; scattering functions and form factors that modify the differential Klein-Nishina and Thomson cross sections; energy deposition data; fluorescence data; and shellwise Compton profile data for photon doppler broadening. The ten data blocks follow, with brief descriptions and table numbers where detailed formats can be found.

1. ESZG Block—contains the coherent, incoherent, photoelectric, and pair production cross sections, all tabulated on a common energy grid. The ESZG Block always exists. See Table F.27.
2. JINC Block—contains the incoherent scattering functions that are used to modify the differential Klein-Nishina cross section. The JINC Block always exists. See Table F.28.
3. JCOH Block—contains the coherent form factors that are used to modify the differential Thomson cross section. The JCOH Block always exists. See Table F.29.
4. JFLO Block—contains fluorescence data. The JFLO Block exists if NXS(4) ≠ 0. See Table F.30.
5. LHNM Block—contains average heating numbers. The LHNM Block always exists. See Table F.31.
6. LNEPS Block contains the number of electrons per shell and is located at XSS(I): I=1,NXS(5). This Block exists if NXS(5) is not zero.
7. LBEPS Block contains the binding energy per shell and is located at XSS(I): I=1...NXS(5)–1.
8. LPIPS Block contains the probability of interaction per shell and is located at XSS(I): I=1...NXS(5)–1.
9. LSWD block contains the list of locators for the shell-wise Compton Profile data. The LSWD Block exists if NXS(5) is not zero. See Table F.32.
10. SWD block contains the shell-wise Compton Profile data. The SWD Block exists if NXS(5) is not zero. See Table F.33.

Table F.27
ESZG Block

Location in XSS	Parameter	Description
JXS(1)	ln[E(I), I=1,NXS(3)]	Logarithms of energies
JXS(1)+NXS(3)	ln[σ_{IN}(I), I=1,NXS(3)]	Logarithms of incoherent cross sections
JXS(1)+2∗NXS(3)	ln[σ_{CO}(I), I=1,NXS(3)]	Logarithms of coherent cross sections
JXS(1)+3∗NXS(3)	ln[σ_{PE}(I), I=1,NXS(3)]	Logarithms of photoelectric cross sections
JXS(1)+4∗NXS(3)	ln[σ_{PP}(I), I=1,NXS(3)]	Logarithms of pair production cross sections

Note: Linear-linear interpolation is performed on the logarithms as stored, resulting in effective log-log interpolation for the cross sections. If a cross section is zero, a value of 0.0 is stored on the data table

Table F.28
JINC Block

Location in XSS	Parameter	Description
JXS(2)	FF$_{INC}$(I), I=1,21	Incoherent scattering functions

Note: The scattering functions for all elements are tabulated on a fixed set of v(I), where v is the momentum of the recoil electron (in inverse angstroms). The grid is: v(I), I=1,21 / 0. , .005 , .01 , .05 , .1 , .15 , .2, .3 , .4 , .5 , .6 , .7 , .8 , .9 , 1. , 1.5 , 2. , 3. , 4. , 5. , 8. /
Linear-linear interpolation is assumed between adjacent v(I).
The constants v(I) are stored in the VIC array in common block RBLDAT.

Table F.29
JCOH Block

Location in XSS	Parameter	Description
JXS(3)	FFINT$_{COH}$(I), I=1,55	Integrated coherent form factors
JXS(3)+55	FF$_{COH}$(I), I=1,55	Coherent form factors

Note: The form factors for all elements are tabulated on a fixed set of v(I), where v is the momentum transfer of the recoil electron (in inverse angstroms). The grid is: v(I), I=1,55 / 0., .01, .02, .03, .04, .05, .06, .08, .10, .12, .15, .18, .20, .25, .30, .35, .40, .45, .50, .55, .60, .70, .80, .90, 1.0, 1.1, 1.2, 1.3, 1.4, 1.5, 1.6, 1.7, 1.8, 1.9, 2.0, 2.2, 2.4, 2.6, 2.8, 3.0, 3.2, 3.4, 3.6, 3.8, 4.0, 4.2, 4.4, 4.6, 4.8, 5.0, 5.2, 5.4, 5.6, 5.8, 6.0 /
The integrated form factors are tabulated on a fixed set of $v(I)^2$, where the v(I) are those defined above. See LA-5157-MS[2] for a description of the integrated form factors and the sampling technique used in MCNP. The constants v(I) are stored in the VCO array. The constants $v(I)^2$ are stored in the WCO array. Both arrays are in common block RBLDAT.

Table F.30
JFLO Block

Location in XSS	Parameter	Description
JXS(4)	e(1), ... ,e(NXS(4))	(See Below)
JXS(4) + NXS(4)	Φ(1), ... ,Φ(NXS(4))	(See Below)
JXS(4) + 2∗NXS(4)	Y(1), ... ,Y(NXS(4))	(See Below)
JXS(4) + 3∗NXS(4)	F(1), ... ,F(NXS(4))	(See Below)
.	.	.
.	.	.
.	.	.

Note:
A complete description of the parameters given in this block can be found in LA-5240-MS.[3]
Briefly:
e(I) are the edge energies,
Φ(I) are relative probabilities of ejection from various shells,
Y(I) are yields, and
F(I) are fluorescent energies.

Table F.31
LHNM Block

Location in XSS	Parameter	Description
JXS(5)	H_{ave}(I), I=1,NXS(3)	Average heating numbers

Note: Log-log interpolation is performed between adjacent heating numbers. The units of H_{ave} are MeV per collision. Heating numbers are tabulated on the energy grid given in the ESZG Block.

Table F.32
LSWD Block

Location in XSS	Parameter	Description
JXS(9)	$LOCA_1$	Location of Compton Profile data for shell 1
JXS(9)+1	$LOCA_2$	Location of Compton Profile data for shell 2
.	.	
.	.	
.	.	
JXS(9) + NXS(5) –1	$LOCA_{NSH}$	Location of Compton Profile data for shell NSH

Table F.33
SWD Block

Location in XSS	Parameter	Description
JXS(10)+LOCA$_1$ –1	JJ$_i$	Interpolation parameter for Compton Profile shell$_i$
JXS(10)+LOCA$_1$	NE$_i$	Number of momentum entries for Compton Profile shell$_i$
JXS(10)+LOCA$_1$ +1	PZ(I)=1...NE$_i$	Momentum entries for Compton Profile shell$_i$
JXS(10)+LOCA$_1$+2+NE	PDF(I)=1...NE$_i$	PDF for Compton Profile shell$_i$
JXS(10)+LOCA$_1$+2+2*NE	CDF(I)=1...NE$_i$	CDF for Compton Profile shell$_i$

VIII. FORMAT FOR MULTIGROUP TRANSPORT TABLES

Table F.34
NXS Array

Parameter		Description
NXS(1)	LDB	Length of second block of data
NXS(2)	ZA	1000*Z+A for neutrons, 1000*Z for photons
NXS(3)	NLEG	Number of angular distribution variables
NXS(4)	NEDIT	Number of edit reactions
NXS(5)	NGRP	Number of groups
NXS(6)	NUS	Number of upscatter groups
NXS(7)	NDS	Number of downscatter groups
NXS(8)	NSEC	Number of secondary particles
NXS(9)	ISANG	Angular distribution type ISANG=0 for equiprobable cosines bins ISANG=1 for discrete cosines
NXS(10)	NNUBAR	Number of nubars given

NXS(11)	IBFP	Boltzmann-Fokker-Planck indicator
		IBFP=0 for Boltzmann only
		IBFP=1 for Boltzmann-Fokker-Planck
		IBFP=2 for Fokker-Planck only

NXS(12)	IPT	Identifier for incident particle
		IPT=1 for neutrons
		IPT=2 for photons
		IPT=0 for other particles (temporary)

NXS(13)–NXS(16) are currently unused.		
All data in the NXS array are appropriate for the incident particle only.		

Table F.35
JXS Array

Parameter		Description
JXS(1)	LERG	Location of incident particle group structure=1
JXS(2)	LTOT	Location of total cross sections
JXS(3)	LFISS	Location of fission cross sections
JXS(4)	LNU	Location of nubar data
JXS(5)	LCHI	Location of fission chi data
JXS(6)	LABS	Location of absorption cross sections
JXS(7)	LSTOP	Location of stopping powers
JXS(8)	LMOM	Location of momentum transfers
JXS(9)	LMTED	Location of edit reaction numbers
JXS(10)	LXSED	Location of edit cross sections
JXS(11)	LIPT	Location of secondary particle types
JXS(12)	LERG2L	Location of secondary group structure locators
JXS(13)	LPOL	Location of P0 locators
JXS(14)	LSANG2	Location of secondary angular distribution types
JXS(15)	LNLEG2	Location of number of angular distribution variables for secondaries
JXS(16)	LXPNL	Location of XP$_N$ locators
JXS(17)	LPNL	Location of P$_N$ locators
JXS(18)	LSIGMA	Location of SIGMA Block locators
JXS(19)	LSIGSC	Location of cumulative P0 scattering cross sections
JXS(20)	LSIGSCS	Location of cumulative P0 scattering cross sections to secondary particle

Note: JXS(18)–JXS(20) are calculated and used internally in MCNP. These parameters have a value of 0 on the cross-section file.
JXS(21)–JXS(32) are currently unused.

Table F.36
ERG Block

Location in XSS	Parameter	Description
JXS(1)	ECENT(1)	Center energy of Group 1
.	.	.
.	.	.
JXS(1)+NXS(5)−1	ECENT(NXS(5))	Center energy of Group NXS(5)
JXS(1)+NXS(5)	EWID(1)	Width of Group 1
.	.	.
.	.	.
JXS(1)+2∗NXS(5)−1	EWID(NXS(5))	Width of Group NXS(5)
JXS(1)+2∗NXS(5)	GMASS(1)	Mass of Group-1 particle
.	.	.
.	.	.
JXS(1)+3∗NXS(5)−1	GMASS(NXS(5))	Mass of Group – NXS(5) particle
Length: 2∗NXS(5) if NXS(12) ≠ 0; 3∗NXS(5) if NXS(12)=0		
Exists: Always		

Note: Group masses are given only if NXS(12)=0.
All entries are in MeV.
Group energies are descending, unless NXS(12)=0, in which case there may be discontinuities.

Table F.37
TOT Block

Location in XSS	Parameter	Description
JXS(2)	SIGTOT(1)	Total cross section in Group 1
.	.	.
.	.	.
JXS(2)+NXS(5)−1	SIGTOT(NXS(5))	Total cross section in Group NXS(5)
Length: NXS(5)		
Exists: If JXS(2) ≠ 0		

Table F.38
FISS Block

Location in XSS	Parameter	Description
JXS(3)	SIGFIS(1)	Fission cross section in Group 1
.	.	.
.	.	.
JXS(3)+NXS(5)−1	SIGFIS(NXS(5))	Fission cross section in Group NXS(5)
Length: NXS(5)		
Exists: If JXS(3) ≠ 0		

Table F.39
NU Block

Location in XSS	Parameter	Description
JXS(4)	NUBAR(1)	See below
.	.	.
.	.	.
JXS(4)+NXS(10)∗NXS(5)−1	NUBAR(NXS(10)∗NXS(5))	See below
Length: NXS(5)∗NXS(10)		
Exists: If JXS(3) ≠ 0		

Note: If NXS(10)=1, then one set of nubars is given (NUBAR(1) ⟶ NUBAR(NXS(5))). The nubars may be either prompt or total.
If NXS(10) = 2, then both prompt and total nubars are given. In this case, NUBAR(1) ⟶ NUBAR(NXS(5)) are prompt nubars and NUBAR(NXS(5)+1) ⟶ NUBAR (2∗NXS(5)) are total nubars.

Table F.40
CHI Block

Location in XSS	Parameter	Description
JXS(5)	FISFR(1)	Group 1 fission fraction
.	.	.
.	.	.
JXS(5)+NXS(5)−1	FISFR(NXS(5))	Group NXS(5) fission fraction
Length: NXS(5)		
Exists: If JXS(3) ≠ 0		

Note: The fission fractions are normalized so that their sum is 1.0.

Table F.41
ABS Block

Location in XSS	Parameter	Description
JXS(6)	SIGABS(1)	Absorption cross section in Group 1
.	.	.
.	.	.
JXS(6)+NXS(5)−1	SIGABS(NXS(5))	Absorption cross section in Group NXS(5)
Length: NXS(5)		
Exists: If JXS(6) ≠ 0		

Table F.42
STOP Block

Location in XSS	Parameter	Description
JXS(7)	SPOW(1)	Stopping power in Group 1
.	.	.
.	.	.
JXS(7)+NXS(5)−1	SPOW(NXS(5))	Stopping power in Group NXS(5)
Length: NXS(5)		
Exists: If JXS(7) ≠ 0		

Table F.43
MOM Block

Location in XSS	Parameter	Description
JXS(8)	MOMTR(1)	Momentum transfer in Group 1
,	.	.
.	.	.
JXS(8)+NXS(5)−1	MOMTR(NXS(5))	Momentum transfer in Group NXS(5)
Length: NXS(5)		
Exists: If JXS(8) ≠ 0		

Table F.44
MTED Block

Location in XSS	Parameter	Description
JXS(9)	MT(1)	Identifier for edit reaction 1
.	.	.
.	.	.
JXS(9)+NXS(4)–1	MT(NXS(4))	Identifier for edit reaction NXS(4)
Length: NXS(4)		
Exists: If JXS(4) ≠ 0		

Table F.45
XSED Block

Location in XSS	Parameter	Description
JXS(10)	XS(1,1)	Edit cross section for reaction 1, Group 1
.	.	.
.	.	.
JXS(10)+NXS(5)–1	XS(1,NXS(5))	Edit cross section for reaction 1, Group NXS(5)
.	.	.
.	.	.
JXS(10)+(NXS(4)–1) *(NXS(5))	XS(NXS(4),1)	Edit cross section for reaction NXS(4), Group 1
.	.	.
.	.	.
JXS(10)+NXS(4)∗NXS(5)–1	XS(NXS(4), NXS(5))	Edit cross section for reaction NXS(4), Group NXS(5)
Length: NXS(4)∗NXS(5)		
Exists: If NXS(4) ≠ 0		

Table F.46
IPT Block

Location in XSS	Parameter	Description
JXS(11)	IPT(1)	Identifier for secondary particle 1
.	.	.
.	.	.
.	.	.
JXS(11)+NXS(8)−1	IPT(NXS(8))	Identifier for secondary particle NXS(8)
Length: NXS(8)		
Exists: If NXS(8) ≠ 0		

Note: Present values of IPT are:
IPT=1 for neutrons
IPT=2 for photons

Table F.47
ERG2L Block

Location in XSS	Parameter	Description
JXS(12)	LERG2(1)	Location of ERG2 Block* for secondary particle 1
.	.	.
.	.	.
JXS(12)+NXS(8)−1	LERG2(NXS(8))	Location of ERG2 Block* for secondary particle NXS(8)
Length: NXS(8)		
Exists: If NXS(8) ≠ 0		

*The ERG2 Block for secondary particle i has the form:

Location in XSS	Parameter	Description
LERG2(i)	NERG(i)	Number of energy groups for secondary particle i
LERG2(i)+1	ECENT2(1)	Center energy of Group 1 for secondary particle i
.	.	.
.	.	.
LERG2(i)+NERG(i)	ECENT2(NERG(i))	Center energy of Group NERG(i) for secondary particle i
LERG2(i)+NERG(i)+1	EWID2(1)	Width of Group 1 for secondary particle i
.	.	.
.	.	.

Location in XSS	Parameter	Description
LERG2(i)+2∗NERG(i)	EWID2(NERG(i))	Width of Group NERG(i) for secondary particle i
Length: 2∗NERG(i)+1		
Exists: If NXS(8) ≠ 0, then ERG2 Block is repeated NXS(8) times.		

Note: Values of LERG2(i) are from ERG2L Block. Group energies are descending.

Table F.48
POL Block

Location in XSS	Parameter	Description
JXS(13)	LPO(1)	Location of P0 Block* for incident particle
.	.	.
.	.	.
JXS(13)+NXS(8)	LPO(NXS(8)+1)	Location of P0 Block* for secondary particle NXS(8)
Length: NXS(8)+1		
Exists: If JXS(13) ≠ 0		

*The P0 Block for particle i is of the form:

Location in XSS	Parameter	Description
LPO(i)	SIG(1 → 1)	P0 cross section for scattering from incident particle Group 1 to exiting particle Group 1
.	.	.
.	.	.
LPO(i+L – 1)	SIG(NXS(5) → K)	P0 cross section for scattering from incident particle group NXS(5) to exiting particle Group K
Exists: If JXS(13) ≠ 0, then the P0 Block is repeated NXS(8)+1 times.		

Note: See Table F.56 for a complete description of the ordering and length of the P0 block.

Table F.49
SANG2 Block

Location in XSS	Parameter	Description
JXS(14)	ISANG2(1)	Angular distribution type for secondary particle 1
.	.	.
.	.	.
JXS(14)+NXS(8)−1	ISANG2(NXS(8))	Angular distribution type for secondary particle NXS(8)
Length: NXS(8)		
Exists: If NXS(8) ≠ 0		

Note: ISANG2(i)=0 for equiprobable cosine bins; ISANG2(i)=1 for discrete cosines.

Table F.50
NLEG2 Block

Location in XSS	Parameter	Description
JXS(15)	NLEG2(1)	Number of angular distribution variables for secondary particle 1
.	.	.
.	.	.
JXS(15)+NXS(8)−1	NLEG2(NXS(8))	Number of angular distribution variables for secondary particle NXS(8)
Length: NXS(8)		
Exists: If NXS(8) ≠ 0		

Table F.51
XPNL Block

Location in XSS	Parameter	Description
JXS(16)	LXPN(1)	Location of XPN Block* for incident particle
.	.	.
.	.	.
JXS(16)+NXS(8)	LXPN(NXS(8)+1)	Location of XPN Block* for secondary particle NXS(8)
Length: NXS(8)+1		
Exists: If JXS(13) ≠ 0		

Note: If LXPN(i)=0, then all possible scattering is isotropic and no XPN block exists.

*The XPN Block for particle i is of the form:

Location in XSS	Parameter	Description
LXPN(i)	LPND(1 → 1)	Location of PND Block† for scattering from incident particle Group 1 to exiting particle Group 1
.	.	.
.	.	.
LXPN(i+L – 1)	LPND(NXS(5) → K)	Location of PND Block† for scattering from incident particle Group NXS(5) to exiting particle Group K
Exists: If JXS(13) ≠ 0, then the XPN Block is repeated NXS(8)+1 times.		

† See Table F.52 for a description of the PND Block.

Note: See Table F.56 for a complete description of the ordering and length of the XPN Block. Also see the notes to the PN Block in Table F.52 for more complete description of the meanings of the LPND parameters.

Table F.52
PNL Block

Location in XSS	Parameter	Description
JXS(17)	LPN(1)	Location of PN Block* for incident particle
.	.	.
.	.	.
JXS(17)+NXS(8)	LPN(NXS(8)+1)	Location of PN Block* for secondary particle NXS(8)
Length: NXS(8)+1		
Exists: If JXS(13) ≠ 0.		

Note: If LPN(i)=0, then all possible scattering is isotropic and no PN Block exists.

*The PN Block for particle i is of the form:

Location in XSS	Parameter	Description
LPN(i)+LPND(1 → 1)–1	PND(1 → 1,I) I=1,NLEG(i)	Angular distribution data for scattering from incident particle Group 1 to exiting particle Group 1
.	.	.
.	.	.
LPN(i)+LPND(NXS(5) → K)–1	PND(NXS(5) → K,I), I=1, NLEG(i)	Angular distribution data for scattering from incident particle Group NXS(5) to exiting particle Group K
Exists: If JXS(13) ≠ 0, then the PN Block is repeated NXS(8)+1 times.		

Note: Values of LPND are from the XPN Block (see Table F.51). Values of LPN(i) are from the PNL Block. If LPND>0, then data exists in the PN Block as described above. If LPND=0, scattering is isotropic in the laboratory system and no data exist in the PN Block. If LPND=−1, then scattering is impossible for the combination of incident and exiting groups; again no data exist in the PN Block. The appropriate value of NLEG is found in Table F.34 or Table F.50. The value of ISANG (from Table F.34 or Table F.49) determines what data are found in the PND array. If ISANG=0, then PND contains NLEG cosines, which are boundaries of NLEG–1 equiprobable cosine bins. If ISANG=1, then PND contains (NLEG–1)/2 cumulative probabilities followed by (NLEG+1)/2 discrete cosines. The cumulative probability corresponding to the final discrete cosine is defined to be 1.0.

Table F.53
SIGMA Block*

Location in XSS	Parameter	Description
JXS(18)	$SCAT_{gg}(1)$	Location of the within–group scattering cross section for Group 1 within the P0 Block
.	.	.
.	.	.
JXS(18)+NXS(5)−1	$SCAT_{gg}(NXS(5))$	Location of the within–group scattering cross section for Group NXS(5) in the P0 Block

Table F.54
SIGSC Block*

Location in XSS	Parameter	Description
JXS(19)	SIGSC(1)	Total P0 scattering cross section for Group 1 excluding scattering to secondary particle
.	.	.
.	.	.
JXS(19)+NXS(5)−1	SIGSC(NXS(5))	Total P0 scattering cross section for group NXS(5) excluding scattering to secondary particle

Table F.55
SIGSCS Block*

Location in XSS	Parameter	Description
JXS(20)	SIGSCS(1)	Total P0 scattering cross section to a secondary particle for Group 1
.	.	.
.	.	.
JXS(20)+NXS(5)−1	SIGSCS(NXS(5))	Total P0 scattering cross section to a secondary particle for Group NXS(5)

*The SIGMA, SIGSC and SIGSCS Blocks are calculated and used internally within MCNP and do not actually appear on the cross-section file.

Table F.56
Additional Information for P0 and XPN Blocks

1. Ordering

Entries in these blocks always start with data for scattering from the highest energy group of the incident particle to the highest energy group of the exiting particle.The last entry is always data for scattering from the lowest energy group of the exiting particle. The remaining entries are ordered according to the following prescription:

X(1→J), J=I1(1), I2(1),
X(2→J), J=I1(2), I2(2),
.
.
.
X(NXS(5)→J), J=I1(NXS(5)), I2(NXS(5)).

If the incident and exiting particles are the same:

I1(K)=MAX(1,K–NXS(6)),
I2(K)=MIN(NXS(5),K+NXS(7)).

If the incident and exiting particles are different:

I1(K)=1,
I2(K)=NERG(i) for the appropriate secondary particle from Table F.47.

2. Length

If the incident and exiting particles are the same:

$$L=NXS(5)*(1+NXS(7)+NXS(6)) - \frac{(NXS(7) \bullet (NXS(7)+1)) + (NXS(6) \bullet (NSX(6)+1))}{2}$$

If the incident and exiting particles are different:

L = NXS(5)*NERG(i), where NERG(i) is for the appropriate secondary particle from Table F.47.

IX. FORMAT FOR ELECTRON TRANSPORT TABLES

This section has not yet been written (see Reference 4).

X. FORMAT FOR PHOTONUCLEAR TRANSPORT TABLES

The JXS Block format deviates from the traditional style in that all secondary-particle emission data is referenced through the IXS construct. Also, the locators TOT, NON, ELS and THN have been added. In neutron continuous-energy tables, the energy grid, the total, absorption and elastic cross sections, and the heating numbers are referenced through the ESZ locator.

Table F.57
Definition of the NXS Array

Entry	Parameter	Fixed Numeric Descriptive
NXS(1)	LXS	Length of the XSS data block
NXS(2)	ZA	Atomic and mass number of the target isotope ZA = Z*1000 + A
NXS(3)	NES	Number of energy entries in the main energy grid
NXS(4)	NTR	Number of MT entries in the reaction-type listing
NXS(5)	NTYPE	Number of secondary particle types with IXS information
NXS(6)	NPIXS	Number of parameter entries (fixed values) in the IXS array per secondary particle
NXS(7)	NEIXS	Number of entries (fixed values and locators) in IXS array per secondary particle
NXS(8-15)		Unused (Fill with zeros)
NXS(16)	TVN	Table Format Version

Table F.58
Definition of the JXS Array

Entry	Locator	Description
JXS(1)	ESZ	Main energy grid
JXS(2)	TOT	Total cross-section data
JXS(3)	NON	Total nonelastic cross-section data
JXS(4)	ELS	Elastic cross-section data
JXS(5)	THN	Total heating number data
JXS(6)	MTR	MT reaction numbers
JXS(7)	LQR	Q-value reaction energy data

Table F.58 (Cont.)
Definition of the JXS Array

Entry	Locator	Description
JXS(8)	LSIG	Cross-section locators (relative to SIG)
JXS(9)	SIG	Primary locator for cross-section data
JXS(10)	IXSA	First word of IXS array
JXS(11)	IXS	First word of IXS block
JXS(12-32)		Unused (Fill with zeros)

A. Data Blocks for Photonuclear Transport Tables

1. ESZ Block—contains the main energy grid and consists of a series of monotonically increasing, positive values located at (XSS(I): I=ESZ, ... , ESZ+NES-1).
2. TOT Block—contains the total cross section. There is an entry in this block corresponding to each entry in the ESZ array. It is located at (XSS(I): I=TOT, ... , TOT+NES-1), and it MUST be present.
3. NON Block—contains the total nonelastic cross section. There is an entry in this block corresponding to each entry in the ESZ Block. It is located at (XSS(I): I=NON, ... , NON+NES-1). The NON Block must exist if any nonelastic cross-section data are present. If ELS is zero, NON=TOT.
4. ELS Block—contains the elastic cross section. There is an entry in this block corresponding to each entry in the ESZ array. It is located at (XSS(I): I=ELS, ... , ELS+NES-1). This cross section is negligible and typically is not included in the original evaluation data file. If it is not included, ELS is set to zero and no entries are included in the XSS array.
5. THN Block—contains the average heating numbers. There is an entry in this block corresponding to each entry in the ESZ array. It is located at (XSS(I): I=THN, ... , THN+NES-1). If no data have been calculated for heating numbers, JXS(5) is zero and no entries are made in the XSS array. For photonuclear data, it is assumed in the calculation of the total heating number that all secondary particles deposit the energy locally and instantaneously, including neutrons, photons, protons, alphas, etc. Based on which particles are transported, this value can then be modified by subtracting the average energy a given secondary particle contributes to the total. This particle average heating number is stored in the appropriate PHN(J) Block.
6. MTR Block—contains a list of ENDF–6 MT numbers for every reaction cross section given. It is located at (XSS(I): I=MTR, ... , MTR+NTR-1). Production cross sections for reaction products of interest can be listed in the MTR Block by using the ZA number in place of the ENDF MT number. Production cross sections are not valid for transport and are used only as FM tally multipliers.
7. LQR Block—contains the Q-value associated with each reaction. There is one entry in this block corresponding to each MTR Block entry. It is located at (XSS(I): I=LQR, ... , LQR+NTR-1). Reactions that are not physical events have an entry of zero.

8. LSIG Block—contains the cross-section locators, the array index to the first word of the corresponding MT reaction data relative to the SIG locator. There is one entry in this block for each MTR Block entry. It is located at (XSS(I): I=LSIG, ... , LSIG+NTR-1).

9. SIG Block—the locator for finding the reaction cross-section data. The data follow a similar IE, NE, VALUES format as described in Table F.10.

10. IXS Block— emulates the parameter/locator concept of NXS/JXS for secondary particle information.[5] Because a full set of IXS elements is needed for each secondary particle, there are typically multiple IXS arrays in a table. They are listed sequentially starting at ((XSS(I): I=(IXSA+NEIXS*(J-1)) , ... , (IXSA+NEIXS*(J-1))+(NEIXS-1): J=1, NTYPE). The elements of the IXS array are described in Table F.59 for each secondary particle type J.

Table F.59
Description of the IXS Array

Entry	Parameter	Description
IXS(1,J)	IPT(J)	Particle IPT number
IXS(2,J)	NTRP(J)	Number of MT reactions producing this particle
IXS(3,J)	PXS(J)	Location of total particle production cross-section data
IXS(4,J)	PHN(J)	Location of particle average heating number data
IXS(5,J)	MTRP(J)	Location of particle production MT reaction numbers
IXS(6,J)	TYRP(J)	Location of reaction coordinate system data
IXS(7,J)	LSIGP(J)	Reaction yield locators (relative to SIGP)
IXS(8,J)	SIGP(J)	Primary locator for reaction yield data
IXS(9,J)	LANDP(J)	Reaction angular distribution locators (relative to ANDP)
IXS(10,J)	ANDP(J)	Primary locator for angular distribution data
IXS(11,J)	LDLWP(J)	Reaction energy distribution locators (relative to DLWP)
IXS(12,J)	DLWP(J)	Primary locator for energy distribution data

11. IPT Block—contains a single value to designate the secondary particle type J information as defined in Table F.60. In preparation for the expansion of MCNP to handle other particles, these tables may contain information for other secondary particles including protons (IPT=9), deuterons (31), tritons (32), helium-3 (33) and alpha particles (34).

Table F.60
Definition of IPT Values by Particle Type

Particle Name	Symbol (from mode card)	IPT
neutron	n	1
photon	p	2
electron	e	3

12. NTRP Block—contains a single value that indicates the number of reactions that produce the secondary particle type J.
13. PXS Block—contains the total secondary particle-production cross section. The data follow a similar IE, NE, VALUES format as described in Table F.10. It is located at ((XSS(I): I=PXS(J), ... , PXS(J)+NE+1): J=1, NTYPE).
14. PHN Block—contains the particle average heating numbers. The data follow a similar IE, NE, VALUES format as described in Table F.10 and are located at ((XSS(I): I=PHN(J), ... , PHN(J)+NE+1): J=1, NTYPE). As described in the THN Block above, these values are the contribution to the total average-heating number by this particle type if the particle's average emission energy is deposited locally.
15. MTRP Block—contains the ENDF/B MT reaction numbers that produce this secondary particle. They are located at ((XSS(I): I=MTRP(J), ... , MTRP(J)+NTRP(J)-1): J=1, NTYPE).
16. TYRP Block—contains the coordinate system of the reaction producing the secondary particle, either the lab system (value = 1) or the center-of-mass system (value = –1). The entries are located at ((XSS(I): I=TYRP(J), ... , TYRP(J)+NTRP(J)-1): J=1, NTYPE). Multiplicity data are not included in TYRP but instead use the SIGP Block.
17. LSIGP Block—contains the reaction yield locators, the relative locations of the corresponding MT reaction data in the SIGP Block. There is one entry (K) in this block corresponding to each MTRP Block entry. It is located at ((XSS(I): I=LSIGP(J), ... , LSIGP(J)+NTRP-1): J=1, NTYPE). The notation LSIGP(K,J) indicates the K^{th} entry (XSS(LSIGP(J)+K-1)) for the J^{th} secondary particle.
18. SIGP Block—the locator for finding the reaction yield data, given either as production cross sections (Table F.61) or as multiplicity data (Table F.62). There is one set of reaction cross-section data for each reaction specified in the MTRP array.

Table F.61
Reaction Yield Data as a Form of Production Cross-section

Location in XSS	Parameter	Description
SIGP(J)+LSIGP(K,J)-1	MFTYPE	13 – Production cross-section
SIGP(J)+LSIGP(K,J)	IE	Starting index on main energy grid
SIGP(J)+LSIGP(K,J)+1	NE	Number of consecutive entries
SIGP(J)+LSIGP(K,J)+2	PXS(I), I=1, NE	Production cross-section values for corresponding MT reaction (linear-linear interpolation)

Table F.62
Reaction Yield Data in Form of Reaction Multiplicity

Location in XSS	Parameter	Description
IXS+SIGP(J)+LSIG(K,J)-1	MFTYPE	6,12 or 16 – Reaction multiplicity
IXS+SIGP(J)+LSIG(K,J)	MTMULT	MT whose cross section should multiply the yield
IXS+SIGP(J)+LSIG(K,J)+1	NR	Number of interpolation regions (If NR = 0, NBT and INT are omitted and linear-linear interpolation is assumed)
IXS+SIGP(J)+LSIG(K,J)+2	NBT(I), I=1, NR	Starting index to which the corresponding interpolation parameter applies
IXS+SIGP(J)+LSIG(K,J)+2+NR	INT(I), I=1, NR	ENDF defined interpolation parameters
IXS+SIGP(J)+LSIG(K,J)+2+2*NR	NE	Number of energies at which the yield is defined
IXS+SIGP(J)+LSIG(K,J)+3+2*NR	E(I), I=1, NE	Energy grid on which yields are defined
IXS+SIGP(J)+LSIG(K,J)+3+2*NR+NE	Y(I), I=1, NE	Multiplicity (production cross-section = reaction MT cross-section * yield)

19. LANDP Block—contains the location of the angular distribution data for the corresponding MT reaction relative to the ANDP locator. There is one entry (K) in this block corresponding to each MTRP Block entry. It is located at ((XSS(I): I=LANDP(J), ... , LANDP(J)+NTRP-1): J=1, NTYPE). LANDP(K,J) is the K^{th} entry for the J^{th} secondary particle type.

    Several LANDP Block values have special meanings. A “0” indicates a reaction where all particles are emitted isotropically in the reference frame defined by the corresponding entry in the TYRP array. A “–1” indicates correlated energy/angle data where the angular distribution data are included with the energy emission distribution data in the DLWP array. For both 0 and –1, no angular data are entered in the ANDP array. Positive integer values indicate that the angular distribution data are contained in the ANDP array. Emission distributions are included explicitly for elastic scattering if such data are included in the ENDF evaluation.

20. ANDP Block—contains angular distributions for secondary particles. If all reactions are isotropic or correlated energy/angle, i.e. there are no positive values in the LANDP array, ANDP is set to zero. Otherwise, ANDP is the offset to the angular distribution block.

    Three types of angular distribution tables are allowed: isotropic, 32 equiprobable bin, and tabulated angular-bin data. The distributions are located using the angular distribution header information described in Table F.63 and are comprised of the average-emission angles for the J^{th} emission particle having the K^{th} reaction.

Table F.63
Angular Distribution Header Information

Location in XSS	Parameter	Description
ANDP(J)+LANDP(K,J)-1	NE	Number of energies at which angular distributions are tabulated
ANDP(J)+LANDP(K,J)	E(I), I=1, NE	Energy grid for the K^{th} reaction angular distribution
ANDP(J)+LANDP(K,J)+NE	LC(I), I=1, NE	Locators for the angular data corresponding to energy grid

LC(1)>0 indicates 32 equiprobable binned data described in Table F.64. LC(1)<0 indicates tabulated angular data described in Table F.65. LC(1)=0 indicates isotropic distributions, and there are no further data entries in the LC(1) array.

Table F.64
Description of 32-Equiprobable Bin Angular Distributions

Location in XSS	Parameter	Description
ANDP(J)+LC(I)-1	CAB(I,M), M=1, 33	32 equiprobable cosine bins for scattering at energy E(I)

Table F.65
Description of Tabulated Angular Distributions

Location in XSS	Parameter	Description
ANDP(J)+|LC(I)|-1	JJ	Interpolation parameter for cosine distribution: 1= histogram or 2= linear-linear
ANDP(J)+|LC(I)|	NP	Number of points in the distribution
ANDP(J)+|LC(I)|+1	CSOUT(I,M), M=1, NP	Cosine of the scattering angle
ANDP(J)+|LC(I)|+1+NP	PDF(I,M), M=1, NP	Probability density function
ANDP(J)+|LC(I)|+2+2*NP	CDF(I,M), M=1, NP	Cumulative density function

21. LDLWP Block—contains the energy distribution locators. These locators are the locations of the emission law data for the corresponding MT reaction. There is one positive integer entry in this block corresponding to each MTRP array entry. It is located at ((XSS(I): I=LDLWP(J), ..., LDLWP(J)+NTRP-1): J=1, NTYPE). Emission distributions are included explicitly for elastic scattering if such data are included in the ENDF evaluation.

22. DLWP Block— emission distribution data for each secondary particle (J). Typically, the emission data described here are the energy spectra for the secondary particle. However, many new data evaluations are taking advantage of the correlated, energy and angle, emission distributions. If the angular distribution data are contained in the emission distribution, the

corresponding LANDP entry must be negative one (–1). For all other cases, there must be a corresponding set of entries, as located by LANDP and ANDP, to describe the appropriate angular distribution.

Law Header. Each reaction has at least one emission distribution associated with it as given in Table F.66.

Table F.66
Emission Parameter Law Header

Location in XSS	Parameter	Description
DLWP(J)+LDLWP(K,J)-1	LNW_i	Location of next law header relative to DLWP(J). If $LNW_i = 0$, then LAW_1 is used regardless of other circumstances.
DLWP(J)+LDLWP(K,J)	LAW_i	Name (number) of this law
DLWP(J)+LDLWP(K,J)+1	$IDAT_i$	Location of law-dependent data relative to DLWP(J)
DLWP(J)+LDLWP(K,J)+2	NR	Number of interpolation regions; if NR = 0, NBT and INT are omitted and linear-linear interpolation is assumed for (E,P) pairs
DLWP(J)+LDLWP(K,J)+3	NBT(I), I=1, NR	Starting index to which the corresponding interpolation parameter applies
DLWP(J)+LDLWP(K,J)+3+NR	INT(I), I=1, NR	ENDF defined interpolation parameter in each region
DLWP(J)+LDLWP(K,J)+3+2*NR	NE	Number of energies
DLWP(J)+LDLWP(K,J)+4+2*NR	E(I), I=1, NE	Tabular energy points
DLWP(J)+LDLWP(K,J)+4+2*NR+NE	P(I), I=1, NE	Probability of law validity
…	…	…
DLWP(J)+$IDAT_i$-1	LDAT	First word of law-dependent data for LAW_i
…	…	…
DLWP(J)+LNW_i-1	LNW_{i+1}	First word of next law header
…	…	…

We now define the format of the LDAT array for each law. In the following subtables, we provide relative locations of data in the LDAT array rather than absolute locations in the XSS array. The variable J always indicates the J^{th} emission particle and the variable K always indicates the K^{th} reaction producing that particle.

a. LAW_i=1 Tabular Equiprobable Energy Bins

Location in XSS	Parameter	Description
LDAT(1)	NR	Number of interpolation regions (If NR = 0, NBT and INT are omitted and linear-linear interpolation is assumed)
LDAT(2)	NBT(N), N=1, NR	Starting index to which the corresponding interpolation parameter applies
LDAT(2+NR)	INT(N), N=1, NR	ENDF defined interpolation parameter in each region Only histogram or linear-linear
LDAT(2+2*NR)	NE	Number of incident energies tabulated
LDAT(3+2*NR))	E_{in}(N), N=1, NE	List of incident energies for which E_{out} is tabulated
LDAT(3+2*NR+NE)	NET	Number of outgoing energies listed in each E_{out} table
LDAT(4+2*NR+NE)	E_{out1}(N), N=1, NET; E_{out2}(N), N=1, NET; … E_{outNE}(N), N=1, NET	E_{out} tables have NET energies listed comprising the boundaries of (NET-1) equiprobable bins. Sampling uses a linear-linear interpolation between bin boundaries.

b. LAW_i=2 Discrete Emission energy

Location in XSS	Parameter	Description
LDAT(1)	LP	Indicates whether the emission particle is primary or nonprimary
LDAT(2)	EG	Emission energy (if LP=0 or LP=1) Binding energy (LP=2)
If LP=0 or LP=1, E_{out} = EG		
If LP=2, E_{out} = EG + (AWR/(AWR+1)) * E_{in}. Its use is strongly discouraged as it assumes simple neutron kinematics for computing the emission energy.		

c. LAW$_i$= 3 and 33 Level Scattering

Location in XSS	Parameter	Description
LDAT(1)	MT	For neutron scattering ((A+1)/A) * |Q|
LDAT(2)	CR	For neutron scattering $(A/(A+1))^2$

Law 3 indicates neutron incident, neutron emission and it is not used for photonuclear data. Law 33 indicates any combination of particles, incident and emitted, is allowed for photonuclear interactions. The parameters should be chosen for photonuclear kinetics instead of neutron kinetics, but this has not been implemented to date and neutron kinematics are still used. Sampling of this law follows the simple formula of E_{out} = LDAT(2) * (E_{in} – LDAT(1)) in the center-of-mass system.

d. LAW$_i$=4, 44 and 61 Tabular Energy Distributions

The common portion of the data format for this set of laws is described below. The format of the tabular distribution is dependent on which law is specified.

Location in XSS	Parameter	Description
LDAT(1)	NR	Number of interpolation regions (If NR=0, NBT and INT are omitted and linear-linear interpolation is assumed)
LDAT(2)	NBT(N), N=1, NR	Starting index to which the corresponding interpolation parameter applies
LDAT(2+NR))	INT(N), N=1, NR	ENDF defined interpolation parameter in each region; If INT=1, histogram and INT=2 linear-linear interpolation are allowed.
LDAT(2+2*NR)	NE	Number of incident energies tabulated
LDAT(3+2*NR)	E(N), N=1, NE	List of incident energies
LDAT(3+2*NR+NE)	L(N), N=1, NE	Locators for tabular distributions relative to DLWP(J)

Energy Law 4:
Tabular Distribution Format containing only energy-emission information. Its data format is described below. Angular distribution data must be included using the ANDP array.

Location in XSS	Parameter	Description

LDAT(3+2*NR+2*NE)
 	
Let M=3+2*NR+2*NE
	INTT′	Combination of the number of discrete photon lines, ND, and the interpolation scheme for subsequent data,
		INTT=1 histogram distribution
		INTT=2 linear-linear distribution

LDAT(M+1)	NP	Number of points in the distribution
LDAT(M+2)	EOUT(I), I=1, NP	Emission energy grid
LDAT(M+2+NP)	PDF(I), I=1, NP	Probability density function
LDAT(M+2+2*NP)	CDF(I), I=1, NP	Cumulative density function

If the value of INTT′ > 0, then

$$\text{INTT}' = (\text{ND}*10)+\text{INTT}$$

where INTT is the interpolation scheme and the first *ND* values of NP points describe discrete particle energies. The remaining *NP* – *ND* values describe a continuous distribution. In this way the distribution may be discrete, continuous, or a discrete distribution superimposed upon a continuous background.

Energy Law 44:
Expands Law 4 format to include the Kalbach parameters for each emission energy. The parameters are used to compute the angular distribution based on the Kalbach-87 formalism.[6,7] For photonuclear reactions, the slope value must be computed at the time the table is produced according to Chadwick's modification[8] to Kalbach's original formalism. Sampling of Law 44 emission energy is analogous to Law 4.

Location in XSS	Parameter	Description

LDAT(3+2*NR+2*NE)
 	
Let M=3+2*NR+2*NE
	INTT′	Combination of the number of discrete photon lines, ND, and the interpolation scheme for subsequent data described above,
		INTT=1 histogram distribution
		INTT=2 linear-linear distribution

LDAT(M+1)	NP	Number of points in the distribution
LDAT(M+2)	EOUT(I), I=1, NP	Emission energy grid
LDAT(M+2)+NP	PDF(I), I=1, NP	Probability density function

Location in XSS	Parameter	Description
LDAT(M+2)+2*NP	CDF(I), I=1, NP	Cumulative density function
LDAT(M+2)+3*NP	R(I), I=1, NP	Kalbach precompound fraction ‘r’
LDAT(M+2)+4*NP	A(I), I=1, NP	Kalbach-Chadwick angular distribution slope value ‘a’

Energy Law 61:
Like Law 44 but tabular angular distributions instead of Kalbach-87.

Location in XSS	Parameter	Description

LDAT(3+2*NR+2*NE)
 	
Let M=3+2*NR+2*NE
	INTT′	Combination of the number of discrete photon lines, ND, and the interpolation scheme for subsequent data described above,
		INTT=1 histogram distribution
		INTT=2 linear-linear distribution

LDAT(M+1)	NP	Number of points in the distribution
LDAT(M+2)	EOUT(I), I=1, NP	Emission energy grid
LDAT(M+2)+NP	PDF(I), I=1, NP	Probability density function
LDAT(M+2)+2*NP	CDF(I), I=1, NP	Cumulative density function
LDAT(M+2)+3*NP	LC(I), I=1, NP	Location of angular distribution tables* associated with incident energies E(N)
If LC(I)>0, it points to a tabular angular distribution. If LC(I)=0, then distribution is isotropic and no further information is needed. 32 equiprobable bin distribution is not allowed.		

For tabulated angular distribution data, LC(I)>0, the following format is followed.

Location in XSS	Parameter	Description
LDAT(N+1) Let N=DLWP(J)+|LC(I)|-1 * for the i^{th} array	JJ	Interpolation parameter for cosine distribution (Only histogram or linear-linear allowed)
LDAT(N+2)	NP	Number of points in the distribution
LDAT(N+3)	CSOUT(P), P=1, NP	Cosine bin boundaries
LDAT(N+3+NP)	PDF(P), P=1, NP	Probability density function
LDAT(N+3+2*NP)	CDF(P), P=1, NP	Cumulative density function

e. LAW$_i$=5 General Evaporation Spectrum (From ENDF–6 File 5 LF=5)

Location	Parameter	Description
LDAT(1)	NR	Interpolation scheme between T's (If NR=0, NBT and INT are omitted and linear-linear interpolation is assumed)
LDAT(2)	NBT(I), I=1,NR	
LDAT(2+NR)	INT(I), I=1,NR	
LDAT(2+2*NR)	NE	Number of incident energies tabulated
LDAT(3+2*NR)	E(I), I=1,NE	Incident energy table
LDAT(3+2*NR+NE)	T(I), I=1,NE	Tabulated function of incident energies
LDAT(3+2*NR+2*NE)	NET	Number of X's tabulated
LDAT(4+2*NR+2*NE)	X(I), I=1,NET	Tabulated probabilistic function

$E_{out} = X(\xi)*T(E)$, where $X(\xi)$ is a randomly sampled table of X's, and E is the incident energy.

f. LAW$_i$=7 Simple Maxwell Fission Spectrum (From ENDF–6 File 5 Law 7)

Location	Parameter	Description
LDAT(1)	NR	Interpolation scheme between T's (If NR=0, NBT and INT are omitted and linear-linear interpolation is assumed)
LDAT(2)	NBT(I), I=1,NR	
LDAT(2+NR)	INT(I), I=1,NR	
LDAT(2+2*NR)	NE	Number of incident energies tabulated
LDAT(3+2*NR)	E(I), I=1,NE	Incident energy table
LDAT(3+2*NR+NE)	T(I), I=1,NE	Tabulated T's
LDAT(3+2*NR+2*NE)	U	Restriction energy

$$f(E \rightarrow E_{out}) = C\sqrt{E_{out}}\ e^{-E_{out}/T(E)}$$

with restriction $0 \le E_{out} \le E - U$

$$C = T^{-3/2}\left[\frac{\sqrt{\pi}}{2}erf(\sqrt{(E-U)/T}) + -\sqrt{(E-U)/T}\ \ e^{-(E-U)/T}\right]^{-1}$$

g. LAW$_i$=9 Evaporation Spectrum (From ENDF-6 File 5 LF=9)

Location	Parameter	Description
LDAT(1)	NR	Interpolation scheme between T's (If NR$_b$=0, NBT$_b$ and INT$_b$ are omitted and linear-linear interpolation is assumed)
LDAT(2)	NBT(I), I=1,NR	
LDAT(2+NR)	INT(I), I=1,NR	

Location	Parameter	Description
LDAT(2+2*NR)	NE	Number of incident energies tabulated
LDAT(3+2*NR)	E(I), I=1,NE	Incident energy table
LDAT(3+2*NR+NE)	T(I), I=1,NE	Tabulated T's
LDAT(3+2*NR+2*NE)	U	Restriction energy

$$f(E \rightarrow E_{out}) = CE_{out}e^{-E_{out}/T(E)}$$

with restriction $0 \le E_{out} \le E - U$

$$C = T^{-2}\left[1 - e^{(E-U)/T}\ (1 + (E-U)/T)\right]^{-1}$$

h. LAW$_i$=11 Energy Dependent Watt Spectrum (From ENDF-6 File 5 LF=11)

Location	Parameter	Description
LDAT(1)	NR_a	
LDAT(2)	NBT_a(I), I=1,NR_a	Interpolation scheme between a's
LDAT(2+NR_a)	INT_a(I), I=1,NR_a	(If NR_b=0, NBT_b and INT_b are omitted and linear-linear interpolation is assumed)
LDAT(2+2*NR_a)	NE_a	Number of incident energies tabulated for a(E_{in}) table
LDAT(3+2*NR_a)	E_a(I), I=1,NE_a	Incident energy table
LDAT(3+2*NR_a+NE_a)	a(I), I=1,NE_a	Tabulated a's
Let L=3+2*(NR_a+NE_a)		
LDAT(L)	NR_b	
LDAT(L+1)	NBT_b(I), I=1,NR_b	Interpolation scheme between b's
LDAT(L+1+NR_b)	INT_b(I), I=1,NR_b	
LDAT(L+1+2*NR_b)	NE_b	Number of incident energies tabulated for b(E_{in}) table
LDAT(L+2+2*NR_b)	E_b(I), I=1,NE_b	Incident energy table
LDAT(L+2+2*NR_b+NE_b)	b(I), I=1,NE_b	Tabulated b's
LDAT(L+2+2*NR_b+2*NE_b)	U	Rejection energy

$$f(E \rightarrow E_{out}) = C_o exp[-E_{out}/a(E)]sinh[b(E)E_{out}]^{1/2}$$

with restriction $0 \le E_{out} < E - U$

Law 11 is sampled by the rejection scheme in LA-9721-MS.[1]

i. LAW$_i$=22 Tabular Linear Function (From UK Law 2)

Location in XSS	Parameter	Description
LDAT(1)	NR	Interpolation parameters that are not used by
LDAT(2)	NBT(I), I=1,NR	MCNP
LDAT(2+NR)	INT(I), I=1,NR	(histogram interpolation is assumed)
LDAT(2+2*NR)	NE	
LDAT(3+2*NR)	E_{in}(I), I=1,NE	Number of incident energies tabulated
LDAT(3+2*NR+NE)	LOCE(I), I=1,NE	List of incident energies for E_{out} tables
Data for E_{in}(1) (Let L=3+2*NR+2*NE):		Locators of E_{out} tables (relative to JXS(11))
LDAT(L)	NF_1	if $E_{in}(I)_i$ E < E_{in}(I+1) and ξ is a
LDAT(L+1)	P_1(N), N=1,NF_1	random number [0,1] then if
LDAT(L+1+NF_1)	T_1(N), N=1,NF_1	$\sum_{k=1}^{k=N} P_I(k) < \xi \leq \sum_{k=1}^{k=N} P_I(k)$
LDAT(L+1+2*NF_1)	C_1(N), N=1,NF_1	
Data for E_{in}(2):		
.	.	$E_{out} = C_I(N)*(E-T_I(N))$

Law 22 is not recommended for use in photonuclear tables. It is similar to Law 1 and Law 4 in that an incident energy is used to sample a tabulated distribution. However, the table is always chosen as the next distribution under the incident energy and no interpolation is done.

j. LAW$_i$=24 Tabular energy multiplier distribution(From UK Law 6)

Location in XSS	Parameter	Description
LDAT(1)	NR	Interpolation parameters that are not used
LDAT(2)	NBT(I), I=1,NR	by MCNP
LDAT(2+NR)	INT(I), I=1,NR	(histogram interpolation is assumed)
LDAT(2+2*NR)	NE	Number of incident energies
LDAT(3+2*NR)	E_{in}(I), I=1,NE	List of incident energies for which T is tabulated
LDAT(3+2*NR+NE)	NET	Number of outgoing values in each table
LDAT(4+2*NR+NE)	T_1(I), I=1,NET	Tables are NET boundaries
	T_2(I), I=1,NET	of NET−1 equally likely
	.	intervals. Linear-linear
	.	interpolation is used between
	T_{NE}(I), I=1,NET	intervals.
$E_{out} = T_K(I)*E$		
where T_K(I) is sampled from the above tables		
E is the incident neutron energy		

Law 24 is not recommended for use by photonuclear tables. It is similar to Law 1 and Law 4 in that an incident energy is used to sample a tabulated distribution. However, the table is always chosen as the next distribution under the incident energy and no interpolation is done.

k. LAW$_i$=66 N-body phase-space distribution (From ENDF-6 File 6 Law 6)

Location in XSS	Parameter	Description
LDAT(1)	NPSX	Number of bodies in the phase space
LDAT(2)	A_p	Total mass ratio for the NPSX particles

$$E_{out} = T(\xi) * E_i^{max}$$

where

$$E_i^{max} = \frac{A_p - 1}{A_p}\left(\frac{A}{A+1}E_{in} + Q\right)$$

and $T(\xi)$ is sampled from

$$P_i(\mu, E_{in}, T) = C_n\sqrt{T}(E_i^{max} - T)^{3n/2-4}$$

where the sampling scheme is from R28 of LA-9721-MS[1] and is described on page 2-47 in Chapter 2.

Law 66 is not recommended for use with photonuclear reactions due to the non-Newtonian nature of photon interactions.

l. LAW$_i$=67 Laboratory Angle-Energy Law (From ENDF-6 File 6 Law 7)

This law is not recommended for photonuclear data.

Location	Parameter	Description
LDAT(1)	NR	Number of interpolation regions
LDAT(2)	NBT(I), I=1,NR	ENDF interpolation parameters. If NR=0,
LDAT(2+NR)	INT(I), I=1,NR	NBT and INT are omitted and linear-linear interpolation is used.
LDAT(2+2*NR)	NE	Number of energies at which distributions are tabulated
LDAT(3+2*NR)	E(I), I=1,NE	Incident neutron energies
LDAT(3+2*NR+NE)	L(I), I=1,NE	Locations of distributions (relative to DLWP(J))

Location	Parameter	Description
Data for E(1) (let K=3+2*NR+2*NE):		

LDAT(K)	INTMU	Interpolation scheme for secondary cosines
		INTMU=1 histogram distribution
		INTMU=2 linear-linear distribution

LDAT(K+1)	NMU	Number of secondary cosines
LDAT(K+2)	XMU(I), I=1,NMU	Secondary cosines
LDAT(K+2+NMU)	LMU(I), I=1,NMU)	Location of data for each secondary cosine (relative to DLWP(J))
Data for XMU(1) (let N=K+2+2*NMU):		

LDAT(N)	INTEP	Interpolation parameter between secondary energies
		INTEP=1 histogram distribution
		INTEP=2 linear-linear distribution

LDAT(N+1)	NPEP	Number of secondary energies
LDAT(N+2)	EP(I), I=1,NPEP	Secondary energy grid
LDAT(N+2+NPEP)	PDF(I), I=1,NPEP	Probability density function
LDAT(N+2+2*NPEP)	CDF(I), I=1,NPEP	Cumulative density function
Data for XMU(2): . .		
Data for XMU(NMU): . .		
Data for E(2): . .		
Data for E(NE): . .		

XI. REFERENCES

1. C. J. Everett and E. D. Cashwell, “A Third Monte Carlo Sampler,” Los Alamos National Laboratory report LA-9721-MS (March 1983).
2. E. D. Cashwell, C. J. Everett, et.al., “Monte Carlo Photon Codes: MCG and MCP,” Los Alamos National Laboratory report LA-5157-MS (March 1973).
3. E. D. Cashwell and C. J. Everett, “MCP Code Fluorescence Routine Revision,” Los Alamos National Laboratory report LA-5240-MS (May 1973).
4. K. J. Adams, “Electron Upgrade for MCNP4B,” Los Alamos National Laboratory internal memorandum, X-5-RN(U)-00-14 (available URL: http://www-xdiv.lanl.gov/PROJECTS/DATA/nuclear/pdf/X-5-RN-00-14.pdf) (May 2000).
5. S. C. Frankle, “Follow-up to XTM:SCF-96-200, Proposed APT Data Library Formats,” Los Alamos National Laboratory internal memorandum, XTM:SCF-96-312 (1996).
6. C. Kalbach, “Systematics of Continuum Angular Distributions: Extensions to Higher Energies,” Los Alamos National Laboratory report, LA-UR-87-4139 (1987).
7. C. Kalbach, “Systematics of Continuum Angular Distributions: Extensions to Higher Energies,” *Physical Review C*, Vol. **37**, No. 6, pp. 2350-2370 (1988).
8. M. B. Chadwick, P. G. Young, and S. Chibas, “Photonuclear Angular-Distribution Systematics in the Quasideuteron Regime,” *Journal of Nuclear Science and Technology*, Vol. **32**, No. 11, pp. 1154-1158 (1995).

MCNP MANUAL INDEX