[illegible]

ENGINEERING CHANGE NOTICE

Page 1 of 2

1. ECN 186116

Proj. ECN

2. ECN Category (mark one) Supplemental [] Direct Revision [X] Change ECN [] Temporary [] Standby [] Supersedure [] Cancel/Void []	3. Originator's Name, Organization, MSIN, and Telephone No. LL Carter/8M730/H0-35/376-3929	3a. USQ Required? [] Yes [X] No	4. Date March 12, 1996
	5. Project Title/No./Work Order No. Certification of MCNP Version 4A for WHC Computer Platforms	6. Bldg./Sys./Fac. No. Hanford Facilities	7. Approval Designator ~~Z~~SQ
	8. Document Numbers Changed by this ECN (includes sheet no. and rev.) SD-MP-SWD-30001, Rev. 7	9. Related ECN No(s). 605701	10. Related PO No. N/A

11a. Modification Work [] Yes (fill out Blk. 11b) [X] No (NA Blks. 11b, 11c, 11d)	11b. Work Package No. N/A	11c. Modification Work Complete N/A Cog. Engineer Signature & Date	11d. Restored to Original Condition (Temp. or Standby ECN only) N/A Cog. Engineer Signature & Date

12. Description of Change

A few minor changes were made including: changing the main public file directory on the UNIX machines from /p/mcnp/ to /apps/mcnp/ to align with the new WHC system configuration; fixing a history track writing error for optionally plotting tracks in the Monte Carlo simulation; and an upgrade in the Hanford mcnph version to allow additional optimization in selection of the direction-of-flight. The 25 test problems have been used to verify that the same answers are obtained here at WHC on each computer platform as at LANL.

13a. Justification (mark one)

Criteria Change	[]	Design Improvement	[X]	Environmental	[]	Facility Deactivation	[]
As-Found	[]	Facilitate Const	[]	Const. Error/Omission	[]	Design Error/Omission	[]

13b. Justification Details

Keep current with state-of-the-art for this important computer program that can be used to determine dose rates, criticality, and reactor physics parameters.

14. Distribution (include name, MSIN, and no. of copies)

See attached distribution list.

RELEASE STAMP

MAR [illegible] 1996

21

A-7900-013-2 (11/94) GEF095

ENGINEERING CHANGE NOTICE

Page 2 of 2

1. ECN (use no. from pg. 1) 186718

15. Design Verification Required	16. Cost Impact				17. Schedule Impact (days)	
	ENGINEERING		CONSTRUCTION			
[] Yes	Additional	[X] $2000	Additional	[] $	Improvement	[]
[X] No	Savings	[] $	Savings	[] $	Delay	[]

18. Change Impact Review: Indicate the related documents (other than the engineering documents identified on Side 1) that will be affected by the change described in Block 12. Enter the affected document number in Block 19.

SDD/DD	[]	Seismic/Stress Analysis	[]	Tank Calibration Manual	[]
Functional Design Criteria	[]	Stress/Design Report	[]	Health Physics Procedure	[]
Operating Specification	[]	Interface Control Drawing	[]	Spares Multiple Unit Listing	[]
Criticality Specification	[]	Calibration Procedure	[]	Test Procedures/Specification	[]
Conceptual Design Report	[]	Installation Procedure	[]	Component Index	[]
Equipment Spec.	[]	Maintenance Procedure	[]	ASME Coded Item	[]
Const. Spec.	[]	Engineering Procedure	[]	Human Factor Consideration	[]
Procurement Spec.	[]	Operating Instruction	[]	Computer Software	[]
Vendor Information	[]	Operating Procedure	[]	Electric Circuit Schedule	[]
OM Manual	[]	Operational Safety Requirement	[]	ICRS Procedure	[]
FSAR/SAR	[]	IEFD Drawing	[]	Process Control Manual/Plan	[]
Safety Equipment List	[]	Cell Arrangement Drawing	[]	Process Flow Chart	[]
Radiation Work Permit	[]	Essential Material Specification	[]	Purchase Requisition	[]
Environmental Impact Statement	[]	Fac. Proc. Samp. Schedule	[]	Tickler File	[]
Environmental Report	[]	Inspection Plan	[]		[]
Environmental Permit	[]	Inventory Adjustment Request	[]		[]

19. Other Affected Documents: (NOTE: Documents listed below will not be revised by this ECN.) Signatures below indicate that the signing organization has been notified of other affected documents listed below.

Document Number/Revision	Document Number/Revision	Document Number Revision
NA		

20. Approvals

	Signature	Date		Signature	Date
OPERATIONS AND ENGINEERING			ARCHITECT-ENGINEER		
Cog. Eng. LL Carter	LL Carter	3-12-96	PE		
Cog. Mgr. J Greenborg	J Greenborg	3-14-96	QA		
QA H Spanheimer	H Spanheimer	4/29/96	Safety		
Safety EJ Krejci	[illegible] EJK	4/5/96	Design		
Environ.			Environ.		
Technical Review RA Schwarz	RA Schwarz	3/12/96	Other		
			DEPARTMENT OF ENERGY		
			Signature or a Control Number that tracks the Approval Signature		
			ADDITIONAL		

Certification of MCNP Version 4A for WHC Computer Platforms

LL Carter
Westinghouse Hanford Company, Richland, WA 99352
U.S. Department of Energy Contract DE-AC06-87RL10930

EDT/ECN: 186718 UC: 2020
Org Code: 8M730 Charge Code: LF002
B&R Code: EW3135040 Total Pages: 78

Key Words: MCNP

Abstract: MCNP is a general-purpose Monte Carlo code that can be used for neutron, photon, or coupled neutron/photon 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 some special fourth-degree surfaces (elliptical tori).

[illegible] 5-7-96
Release Approval Date

Approved for Public Release

RECORD OF REVISION	(1) Document Number WHC-SD-MP-SWD-30001	Page 2

(2) Title
Certification of MCNP Version 4A for WHC Computer Platforms

CHANGE CONTROL RECORD

(3) Revision	(4) Description of Change - Replace, Add, and Delete Pages	Authorized for Release	
		(5) Cog. Engr.	(6) Cog. Mgr. Date
6 RS	Upgrade to Version 4A, ECN-186710	LL Carter	J Greenborg
		LL Carter	[illegible]
		1-4-94	1-4-94
7RS	Minor Upgrade of 4A, ECN-605701	LL Carter	J Greenborg
		LL Carter	[illegible]
		1-27-95	1/27/95
8 RS	Minor Upgrade of 4A, ECN-186718	LL Carter	J Greenborg
		LL Carter	[illegible]
		3-12-96	3/14/96

RECORD OF REVISION	(1) Document Number WHC-SD-MP-SWD-30001	Page 1

(2) Title

Certification of MCNP Version 4XE for WHC Computer Platforms

CHANGE CONTROL RECORD

(3) Revision	(4) Description of Change – Replace, Add, and Delete Pages	Authorized for Release		
		(5) Cog./Proj. Engr	(6) Cog./Proj. Mgr	Date
0	(7) EDT-456711			
0-A	Replace pages 16 and 17; pages A-13 through A-19; pages A-24 and A-25. Replace Appendices B and C in their entirety. Released per ECN 110604.	L. L. Carter L L Carter 10-9-89	J. W. Daughtry [illegible] 10-9-89	
0-B	Replace pages 9, 10, and 16. Remove page A-6; insert pages A-6.1, A-6.2, and A-6.3. Remove pages A-14, A-15, A-16, and A-17; insert pages A-14, A-15, A-16, A-17.1, A-17.2, and A-17.3. Replace Appendix B in its entirety. Release per ECN 110605.	L. L. Carter L L Carter 12-29-89	J. W. Daughtry [illegible] 12-29-89	
0-C	Replace pages 4, 6, 7, 8, and 16. Replace pages A-14, A-15, A-16, A-17.1, A-17.2, and A-17.3. Replace Appendices B and C in their entirety. Released per ECN 137351.	L. L. Carter L L Carter 4-17-90	J. W. Daughtry [illegible] 4-17-90	
1	Not used due to administrative error.			
2	Initial release of Revision 2 certifying MCNP on the Sun Workstations and Appendix G added. Released per ECN 137352.	R. A. Schwarz [illegible] 9-4-90	L. L. Carter L L Carter 9-4-90	
3 RS	Upgrade to new Version 4.2; all pages replaced. Appendix E was deleted since it is covered in Section 5.0 of the text. Hence, the upgraded Appendix F and Appendix G from Revision 2 become Appendix E and Appendix F in this Revision 3. Released per ECN 137355.	L. L. Carter L L Carter 9-4-91	L. L. Carter L L Carter 9-4-91	
4	Upgrade to new UNICOS 6.1 operating system on the Cray, along with minor additional upgrades; all pages replaced.	L. L. Carter L L Carter 8-5-92	J. Greenborg [illegible] 8/6/92	
5 RS	Upgrade to version 4XE and include HP and SGI Computer Platforms.	LL Carter L L Carter 3-22-93	J. Greenborg [illegible] 3/25/93	

BEST COPY AVAILABLE

A-7320-005 (10/88)

TABLE OF CONTENTS

APPENDICES

1.0 SCOPE

The Monte Carlo method for treating neutron and photon transport emerged from work at Los Alamos during World War II. Since that time, over 300 person-years have been invested at Los Alamos to obtain the current Monte Carlo techniques and data. The culmination of this effort is the Monte Carlo N-Particle (MCNP) computer code that has been used to solve tens of thousands of particle transport problems. Most of these are neutron or photon transport, but the code also has capabilities for electron transport.

Version 4A of the MCNP code has now been transferred from Los Alamos National Laboratory (LANL) to Sun[a], Silicon Graphics[b], HP[c] and IBM[d] platforms on the SECC Westinghouse Hanford UNIX Network: the acronyms Sun, SGI, HP, and IBM will be used in this ECN to designate these platforms. Version 4A has also been implemented on platforms under the DOS[e] operating system: the acronym DOS will be used to designate these platforms.

In transferring the code, a few minor system dependent changes were made to the source code to enable MCNP to work properly in the WHC computer environment. These changes were minor since LANL writes the code with a pre-processor that will generate compatible source code for specified hardware and software. Two executables of MCNP exist on each platform: one executable --mcnp-- is a direct translation of the Los Alamos code, while the other executable --mcnph-- is an enhanced version of MCNP that allows for a number of useful options not available in the main code.

The code custodian for MCNP at WHC is Lee Carter, and the backup code custodian is Randy Schwarz. Any questions regarding the code should be referred to them.

It is the intent of this document to show that execution of the MCNP code at Hanford gives the same results as the base MCNP code executed on the Los Alamos machines. It also will be demonstrated that the additional options available through the enhanced Hanford version of MCNP will produce the expected results.

This main body of this Engineering Change Notice (ECN) and Appendix A will focus on the general aspects of implementation on any of the UNIX[f] platforms with specific examples given for Sun Workstations. The changes made in the FORTRAN code to adjust to the WHC UNIX computing environment are given in Appendix B. Unique

[a]Sun is a registered trademark of Sun Microsystems, Inc.

[b]SGI is a registered trademark of Silicon Graphics, Inc.

[c]HP is a registered trademark of Hewlett Packard Company.

[d]IBM is a registered trademark of IBM Corporation.

[e]DOS is a registered trademark of Microsoft, Inc.

[f]UNIX is a registered trademark of AT&T.

considerations for the various UNIX platforms are given in Appendix C and in Appendix D for a DOS platform. Execution of the SABRINA pre- and post- processor code is discussed in Appendix E, where its most important use is to obtain three-dimensional perspective views of MCNP geometry models.

2.0 PROGRAM DESCRIPTION

Complete information on running MCNP, and the algorithms used in the code, are discussed in detail in the new MCNP manual (Briesmeister, editor, 1993). The MCNP manual is hundreds of pages long and will not be included as an appendix due to its large size. No attempt will be made to duplicate the LANL documentation in this ECN. Only the implementation at Hanford and differences due to the Hanford computing environment will be discussed in this ECN.

MCNP is a general-purpose Monte Carlo code that can be used for neutron, photon, or coupled neutron/photon transport, and has the capability to simulate electron transport also. Some of the more common applications of MCNP are:

1) Radiation assessments for waste management.
2) Shielding calculations.
3) Criticality calculations.
4) Particle streaming analysis calculations.
5) Nuclear heat deposition calculations.
6) Fusion neutronics calculations.
7) Neutron activation calculations.
8) Coupled neutron-photon calculations.

MCNP has the capability of modeling very complex three-dimensional configurations of materials in geometrical cells surrounded by first and second degree surfaces and some special fourth degree surfaces (elliptical tori). Included in the geometry package is a repeated structure (lattice geometry) capability using hexagonal prisms or hexahedra. It uses continuous energy or multigroup cross section sets, allowing for very accurate transport calculations to be made. A discussion of much of the physics and mathematics involved in a MCNP calculation is given in the Monte Carlo book by Carter and Cashwell of reference 2.

A user's manual for the executables of MCNP at WHC is given in Appendix A. This manual explains how to run both executables (mcnp and mcnph) on the various computer platforms, along with a description of the required additional input for the enhanced options in mcnph and its format.

3.0 TRANSFERRING MCNP FROM LANL TO WHC

3.1 THE MASTER PROGRAM LIBRARY FOR PRODUCING SOURCE CODE

MCNP has been maintained at Los Alamos using a FORTRAN code, prpr, that performs record keeping for MCNP. Prpr is a small FORTRAN 77 code[1] of only about 300 lines (including comment statements) that is easily implemented on computer platforms.

It is a software control system that enables the code developer to store and update a master program library while keeping track of all the changes made to that library. The master program library consists of a set of special commands embedded within a FORTRAN program, where these commands insert, delete, write lines to the compile file and in other ways operate on the FORTRAN program. Each line in the FORTRAN program is uniquely identified with a line identifier. When the program is modified, lines can be inserted or deleted by referring to the appropriate line identifier.

A list of commands to modify a master program library is contained in a file known as a patch file, and each new line that is inserted from this file is also assigned a unique identifier. A powerful feature of the master program file for MCNP is that it is written so that a source code for MCNP can be generated that is consistent with the computer characteristics (hardware, operating system, and graphics software) for a desired computer platform. In addition to any modifications, the patch file also contains a "*define" card specifying the computer platform characteristics. Execution of prpr with the appropriate patch file defining the computer platform, and any changes that need to be made for the local environment, then produces the source code that can be compiled and loaded on the platform.

Work stations typically have 32-bit words for floating point arithmetic so double precision is used in the MCNP source code for these platforms to obtain an accuracy equivalent to 64-bit words. However, all cross sections are stored in single precision to conserve memory, since this is more than adequate for the present accuracy of microscopic cross sections. However, the use of single precision for cross sections means that some problems will not track the same on each platform; i.e., in the random walk simulation, the calculations often differ from full 64-bit calculations due to round-off in randomly sampling from the cross section data base.

3.2 OBTAINING THE APPROPRIATE MCNP FILES FROM LANL

The appropriate files for version 4A of MCNP were obtained on October 8, 1993 from the directory /x6code/mcnp4a/ of cfs at LANL, except for the source files mcnp4a.id and mcnpc.id, which were separately provided on a temporary disk. The relevant files included:

- mcnp4a.id	--	the master program library for the FORTRAN portion of MCNP in prpr format -- the file mcnp4a.id is identical to the file mcnpf.id in the 4A package from RSIC;
- mcnpc.id	--	the master program library for the C portion of MCNP in prpr format;
- prpr.id	--	the FORTRAN program prpr;
- makxs.id	--	the master program library in prpr format for obtaining the FORTRAN program to convert card image cross sections to binary;
- makemcnp.sun .hp .sgi	--	the command file used by LANL to produce the MCNP source code with prpr, compile, and load on Sun, HP, SGI, IBM, or DOS platforms, respectively;

```
           .ibm
           .dos
  - patch.sys      --   a file showing the one line generic "*define" line to
                        obtain a compile file for various hardware, graphics,
                        and parallelism;
```

Future versions of MCNP will be released through the Radiation Shielding Information Center (RSIC) at Oak Ridge National Laboratory, so future versions will be obtained from there.

3.3 PRODUCING SOURCE CODE, COMPILING AND LOADING MCNP ON COMPUTER PLATFORMS

All baseline considerations for each platform are already included in the master program library files mcnp4a.id and mcnpc.id. The appropriate source code compile file is generated with prpr by simply specifying the hardware and software configuration on the "*define" card as discussed in section 3.1. We also include in our patch file descriptive information to identify the "QA" version of the code, which is written to the outp file during execution, and some miscellaneous changes to adapt to our particular computer environment, where very few such changes were needed.

The command file that was used to create the executable mcnp file on the Sun Workstations is shown in Figure 1. Line 12 in Figure 1 compiles and loads the prpr program that is subsequently executed to produce the compile file containing the C routines for MCNP at line 17 and another compile file containing the FORTRAN routines at line 26. The prpr program expects that the master program library will be on a file named codef so the C master program library mcnpc.id is copied to codef at line 16 and the FORTRAN master program library is copied to codef at line 24, prior to the two executions of prpr to produce the C source code and FORTRAN source code, respectively.

Each prpr execution produces an output file called compile that is the source code ready for compilation. The C portion of the source code for MCNP is switched to a file named mcnpc.c on line number 18 of Figure 1 and compiled on line 21. The FORTRAN portion of the source code for MCNP is separated into individual subroutines on line 29 with fsplit and compiled on line 35 after the source code for the individual subroutines has been moved to a sub-directory flib/ on line 32. The compilation of the C and FORTRAN subroutines on lines 21 and 35, respectively, produces machine code in *.o files. The executable file, mcnp, is produced in the load instruction on line 37 using these *.o file. Finally, in line 39 of Figure 1 the *.o files are moved to the subdirectory olib/, but typically these *.o files and the *.f files are no longer needed and so would be deleted.

After generating an executable with the control stream of Figure 1, the last few lines of the newf.id and newc.id files should be checked for any error messages relative to the prpr updates.

Figure 1. The Command File, makemcnp.sun, to Create the Executable File, mcnp.

```
#! /bin/sh
# shell script to make MCNP-4a on SUN Sparc Station
# local files (required):  mcnpc.id mcnp4a.id p.4a00c prpr.id
#                          pc4a
mkdir flib
mkdir olib
rm flib/*
rm olib/*
set -x
# compile and load prpr
cp prpr.id prpr.f
/apps/lang/f77 prpr.f -o prpr
rm -f prpr.f newid compile
# merge main C code with C patch file
cp pc4a patch
cp mcnpc.id codef
prpr
mv compile mcnpc.c
mv newid newc.id
# compile C code
/usr/bin/cc  -O3 -dalign -c -I/usr/openwin/include mcnpc.c
rm -f codef patch
# merge main FORTRAN code with patch file
cp mcnp4a.id codef
cp p.4a00c patch
prpr
mv newid newf.id
# split FORTRAN source file
/usr/bin/fsplit compile > clog
rm -f compile codef patch clog
# put FORTRAN source in flib/ and C source in clib/
mv *.f flib
mv *.c flib
# compile FORTRAN
/apps/lang/f77 -O2 -Nn6000 -Nq6000 -Ns6000 -Nx2000 -dalign -c flib/*.f
# load mcnp with both C and FORTRAN .o files
/apps/lang/f77 -o mcnp -Bstatic *.o -lX11
# move .o files to olib/
mv *.o olib
```

The FORTRAN change file, p.4a00c, used in Figure 1 is listed in Appendix B. The first line on this file "define unix, cheap,sun,pointer,plot,mcplot,gkssim,xlib" is used to communicate to prpr the hardware and software configuration. Many of these acronyms have obvious meanings. The acronym "cheap" denotes a 32 bit machine: with the use of this acronym prpr will produce MCNP source code that utilizes double precision for floating point variables except for the cross sections, which are treated with single precision to conserve memory. The acronym "pointer" will treat a variable portion of common using pointers so that machine memory can be dynamically allocated (using malloc and realloc) during execution. The acronyms "plot" and "mcplot" generate source code for the user to optionally obtain plots of the geometry or tallies, respectively. The acronym "xlib" will produce source code to do the plots in X windows[9]. The remainder of the p.4a00c file includes descriptive information to identify the "QA" version of the code and miscellaneous changes to adapt to our particular computer environment, where most of those changes involve creation of a postscript plot file to obtain hard copies of the plots. Essentially all of the changes to the C are for these postscript plots.

The corresponding command file that creates the mcnph executable file is shown in Figure 2. This command file is essentially identical to that shown in Figure 1 for obtaining the mcnp executable, except that "mcnph" replaces "mcnp" on the load line and the patch file ph.4a00c replaces p.4a00c. Refer to Appendix A for information concerning the capabilities of the enhanced Hanford version of MCNP.

Command files for generating the executables on the other UNIX platforms are very similar to those shown in Figures 1 and 2, and are specifically discussed in Appendix C. The command files for generating the executables on the DOS platform are given in Appendix D.

[9]X Window System is a trademark of the Massachusetts Institute of Technology.

Figure 2. The Command File, makemcnph.sun, to Create the Executable File, mcnph.

```
#! /bin/sh
# shell script to make MCNPH-4a on SUN Sparc Station
# local files (required):  mcnpc.id mcnp4a.id ph.4a00c prpr.id
#                          pc4a
mkdir flib
mkdir olib
rm flib/*
rm olib/*
set -x
# compile and load prpr
cp prpr.id prpr.f
/apps/lang/f77 prpr.f -o prpr
rm -f prpr.f newid compile
# merge main C code with C patch file
cp pc4a patch
cp mcnpc.id codef
prpr
mv compile mcnpc.c
mv newid newc.id
# compile C code
/usr/bin/cc  -O3 -dalign -c -I/usr/openwin/include mcnpc.c
rm -f codef patch
# merge main FORTRAN code with patch file
cp mcnp4a.id codef
cp ph.4a00c patch
prpr
mv newid newf.id
# split FORTRAN source file
/usr/bin/fsplit compile > clog
rm -f compile codef patch clog
# put FORTRAN source in flib/ and C source in clib/
mv *.f flib
mv *.c flib
# compile FORTRAN
/apps/lang/f77 -O2 -Nn6000 -Nq6000 -Ns6000 -Nx2000 -dalign -c flib/*.f
# load mcnp with both C and FORTRAN .o files
/apps/lang/f77 -o mcnph -Bstatic *.o -lX11
# move .o files to olib/
mv *.o olib
```

3.4 CREATING A PROGRAM LISTING

The FORTRAN portion of the MCNP compile file is about 100,000 lines long. This length is difficult to work with when creating patches or trying to follow the logic of the code. It turns out that if the common statements are removed from the program listing, its size is cut to about 38,000 lines. Such a listing is easily obtained by simply printing the newf.id file that is generated by the job stream of either Figure 1 or Figure 2 (or the newc.id file for a listing of the C). Simply listing the mcnp4a.id (or mcnpc.id) file produces a listing of the code as obtained from LANL. These listings are not included in this document due to their large size, but they are readily available from bluegate archival storage as described in a subsequent section.

4.0 THE WHC USER ENVIRONMENT FOR MCNP

4.1 EXECUTION OF MCNP

A detailed description of how to execute mcnp and mcnph is given in Appendix A. Presently, a command file for executing mcnp is available on all UNIX platforms in the disk directory /apps/mcnp/ for WHC UNIX computer platforms. This /apps/mcnp/ directory also contains a cross section directory file, xsdir, and a number of cross section files, where the use of these files is typically transparent to the user. Table 1 lists the files contained in the UNIX public file disk directory /apps/mcnp/.

The activation libraries, alvi to a8vi, have been generated[3] from the Fusion Evaluated Nuclear Library that is based upon the best evaluation from the world community. The user can refer to the ascii file actdir for information on the contents of these activation libraries. The file xsdir contains the specific identifiers, which end in *.66y for targets in the ground state and *.67y for those targets in an isomeric state.

MCNP can be executed from any of the platforms by simply entering the line:

```
        /apps/mcnp/mcnp inp=inpfile outp=outfile runtpe=runfile
```

which optionally renames the files inp, outp, and runtpe, and tells MCNP to initiate and run the problem (default). Simply entering

```
    /apps/mcnp/mcnp
```

means that the user will supply an input file named inp describing the problem, and that the generated output and problem dump files will be named the default outp and runtpe, respectively. Including the entry "ip" in the above execute lines will initiate and allow the user to ask for cross sectional views of the geometry, or including the entry "z" will allow the user to ask for plots of tallies from a previous run. See Appendix A for a more detailed discussion. Utilizing /apps/mcnp/mcnph in the above execute lines, rather than /apps/mcnp/mcnp, will execute the extended Hanford version of the code rather than the LANL version.

These mcnp and mcnph files are actually command files. These command files check to see what platform is being used and then initiate the requested execution on that platform.

Table 1. MCNP Public Files for UNIX Platforms.

dir: /apps/mcnp/

mcnp	command file for execution of MCNP
mcnph	command file for execution of extended Hanford version of MCNP
532dos	dosimetry cross section library
alvi to a8vi	activation libraries from master fusion data base
actdir	descriptive information for alvi to a8vi
buflib	buildup (ISOSHLD) factor library for point kernel
csfish53	code to generate a cross section print file for an isotope
d91	discrete library
drmccsl	discrete library
ell	electron transport library
endf5p	neutron cross section library (ENDF/B-V)
endf5t	neutron cross section library (ENDF/B-V)
endf5u	neutron cross section library (ENDF/B-V)
endl85	neutron cross section library
eprixs	neutron cross section library (ENDF/B-V)
fishinst	instructions for executing fishoutp
fishoutp	code to scan outp file in x windows
fishread	code to scan cross section file
gpbuf	buildup factor library from ORNL
irnatl	natural (ENDF?B-V) iridium
kidmanl	neutron cross section library (ENDF/B-V)
llldos	dosimetry cross section library
mcnped	command file for execution of visual editor
mcnp.log	log file showing user executions of mcnp
mcnph.log	log file showing user executions of mcnph
mcnped.log	log file showing user executions of mcnped
mcplib	gamma-ray cross section library
newxs	neutron cross section library (ENDF/B-V)
pvmgetarch	script file to determine machine architecture
rmccs	neutron cross section library
rmccsa	neutron cross section library
rmccsb	neutron cross section library
rlvi to r5vi	neutron ENDF/B-VI cross sections
sabrina	command file for execution of SABRINA
super	neutron cross section library (EMDF/B-V)
tmccs	neutron cross section library, thermal
wl84xsl	multigroup library for a test problem
xsdir	cross section directory
tcard.f	FORTRAN program to convert binary to card image

4.2 ARCHIVAL STORAGE OF MCNP FILES

All the files used in creating Version 4A of MCNP and the files used by MCNP are stored in various subdirectories under the main bluegate directory /v93586/. The following listing shows the location in bluegate of the pertinent MCNP files.

/v93586/mcnp4a/current/ Directory for current "QA" version
[The subdirectories sun/, sgi/, hp/, ibm/, and dos/ also contain executables for mcnp and mcnph for these platforms.]

makemcnp.sun	makemcnph.sun	makemcnp.sgi	makemcnph.sgi
makemcnp.hp	makemcnph.hp	makemcnp.ibm	makemcnph.ibm
p.4a00c	ph.4a00c	pc4a	makxs.id
mcnp4a.id	mcnpc.id	prpr.id	tcard.f
dos/linkmcnp.bat	dos/patchs.dos	dos/f7713.eer	

/v93586/mcnp4a/old/ Previous "QA" old Version 4A's

p.4a00a	ph.4a00a	pc4a
p.4a00b	ph.4a00b	

/v93586/mcnp4a/test/ Directory for testing new versions

p.4a0uu	ph.4a0uu

/v93586/crossl/ Directory for ascii cross section files

532dosl	endf5pl	endf5tl	endf5ul
endl851	llldosl	newxsl	rmccsl
rmccsal	rmccsbl	superl	tmccsl
eprixsl	mcplibl	ell	d91
drmccsl	wl84xsl	irnatl	kidmanl
endf6/	dos/mcplibl	dos/testlibl	

/v93586/crossl/convrt/ Files to create binary cross sections

buflib	xsdirl.d	specs	xsdir
dos/xsdir			

/v93586/mcnp4a/prob/ Directory of test problems and associated command files

probrun.sun	probrunh.sun	probrun.sgi	probrunh.sgi
probrun.hp	probrunh.hp	probrun.ibm	probrunh.ibm
hanhrun.sun	testinp.tar	ihan.tar	testliba
testlibb	testlibc	xsdir	
testmctl.sun	testmctlh.sun	testoutp.sun	testoutph.sun
	testmctlh.sgi	testoutp.sgi	testoutph.sgi
testmctl.hp	testmctlh.hp	testoutp.hp	testoutph.hp
testmctl.aix	testmctlh.aix	testoutp.aix	testoutph.aix
chl	chml	wcolc	wsrcc
dos/runprob.bat	dos/tinp_dos.exe	dos/tmct_dos.exe	dos/tout_dos.exe

The official current unix executable versions of MCNP will reside in the subdirectories sun/, hp/, sgi/, ibm/ of the bluegate directory /v93586/mcnp4a/current/. If there is any question as to the validity of the public MCNP executable files, the executable files can be retrieved and executed from this directory.

The directory /v93586/mcnp4a/old/ will contain any old (Version 4A vintage) versions of MCNP that have been under Quality Assurance (QA).

Most of the files shown above are discussed in this Engineering Data Transmittal (EDT). The directory /v93586/mcnp4a/test/ is for testing new versions of the code. The files p.4a0uu and ph.4a0uu in this directory are identical to the patch files p.4a00c and ph.4a00c for the current versions, except the print statements for the output files have been changed from giving the QA information for the official version to a statement that this is a test version. Hence, these patch files can be revised to test new versions, and additional files will be added to this directory as such testing proceeds.

4.3 PLOTTING MCNP GEOMETRIES AND TALLIES

As described in the MCNP manual (Briesmeister 1993) and Appendix A, MCNP geometries can be plotted by entering "ip" on the mcnp or mcnph execute line. Another option is to plot MCNP tallies. This can be done by entering "z" on the mcnp or mcnph execute line. However, the tally plotting routine requires that either the runtpe file or the mctal file exist since all the plots are made using the information contained in these files from running the problem.

5.0 MCNP CROSS SECTION DATA BASE

MCNP has a number of different cross section treatments available to the user. The most commonly used cross section treatment is the continuous energy neutron interaction cross section treatment which contains energy grids (unique to each isotope) which are sufficiently dense to allow for linear-linear interpolation within a tolerance of about 1%. The resulting energy grid may contain from as few as 250 points (^{1}H) to as many as 22,500 points (^{197}Au). The continuous energy cross section set is primarily based on the ENDF/B-V or ENDF/B-VI national data base. Other available cross section treatments include 262 group discrete neutron interaction cross sections, photon reaction cross sections, neutron dosimetry cross sections, neutron thermal cross sections, and electron transport cross sections.

The cross section sets contain cross section information for the various neutron, photon, and electron interactions along with cross sections defining the angular distribution of the scattered particles, photon production cross sections, and energy deposition microscopic KERMA factors.

Only the most commonly used cross section sets have been transferred from Los Alamos to WHC. For a quick listing of the cross sections available under UNIX for a particular isotope on these cross section files, the user should refer to the file xsdir contained in the /apps/mcnp/ disk directory. For example, to obtain a listing of cross section files for ^{6}LI, do "grep 3006 /apps/mcnp/xsdir".

The ascii cross section files obtained from LANL are usually converted to a binary format in order to reduce the disk storage requirements and increase the I/O speed. To convert an ascii card image file to binary, the following steps are taken:

- The master program file, makxs.id (see under listing of bluegate directory /v93586/mcnp4a/current in section 4.2), is used with prpr to generate the FORTRAN source code and this is compiled and loaded to obtain the executable makxs.

- The makxs program is executed with an input file, specs, that tells makxs what to do, an old directory that tells where cross sections for each isotope are located on the card image library, and the card image library. Makxs then reads the card image cross section file, creates a new binary cross section file, and a directory that describes the data in the binary file. This directory is appended onto the master directory file xsdir.

The input file, specs, contains the old directory file name and the name for the new directory file. Also contained in the specs file is the name of the ascii cross section file to be converted and the corresponding name of the new binary cross section library that is being created and the number "2", indicating the conversion to binary. An example specs file for converting the electron cross section file ell from ascii to binary is as follows, assuming that the conversion is being done in the disk directory /t_sunl/llc/:

Specs File for Converting Electron Library to Binary

```
xsdirl.d xsdir2
/t_sunl/llc/ell /t_sunl/llc/el 2 2048 512
```

where 2048 is the record length of the binary records and 512 is the number of entries per record.

The file xsdirl.d gives information about the ascii cross sections, including where they reside, and can be obtained from the bluegate directory /v93586/crossl/convrt/. An editor must be used to change the directory to /t_sunl/llc/ell for the location of the ascii cross sections in the above example. This ascii file is brought in from the bluegate directory /v93586/crossl/. The new binary cross sections will reside in the disk file /t_sunl/llc/el and the directory file xsdir2 will give information about this file. This xsdir2 file will be modified to point to the /apps/mcnp/ directory when the el file is moved into the public file space, and then the xsdir2 file must be appended to the xsdir file in /apps/mcnp/. See the MCNP Manual[1] for a more detailed discussion of card image to binary conversion of cross sections.

The files mcnp or mcnph for Version 4a may be executed without copying the xsdir file to the local directory from which the code is being executed. The code will look for xsdir in the local directory, and if it is not there, it will use the xsdir file in /apps/mcnp/.

6.0 VERIFICATION OF MCNP EXECUTABLES

One purpose of this document is to show that the version of MCNP at WHC performs identically to the Los Alamos version. The code development group for MCNP has 25 test problems that are used to assure that the code produces the same answers after code changes or adaptation to new computers. These test problems were executed with both mcnp and mcnph on each computer platform. Then files of the output tallies (mctal files) from LANL are compared to the files produced by mcnp and mcnph, and similar comparisons are made for the corresponding outp files.

A highly reliable indicator of the repeatability of a problem is the number of pseudo random numbers generated during the execution of a problem. The number of random numbers generated is dependent on the physics of the problem; thus, if an error exists in an executable code, the number of random numbers generated for a specific problem will not match up with the number obtained from a previous version. This technique was used in 11 test problems to provide a further check on the extended Hanford version of MCNP, (MCNPH), and to provide a problem test base for future versions of MCNPH.

The accuracy of the MCNP and MCNPH codes on the WHC computer platforms is verified by rerunning the test problems for each new executable. Figure 3 gives a listing of the command file probrun.sun which will run the 25 LANL test problems and check the results on a Sun Workstation, where the corresponding command file for MCNPH executes with mcnph rather than mcnp. The testinp.tar file used in Figure 3 contains the 25 test problems in tar format. The corresponding testoutp.sun and testmctl.sun files contain the outp and mctal (tally) files for the 25 test problems.

Figure 4 shows the command file hanhrun.sun that will run the 11 test problems to test the MCNPH options. This allows for a method of verifying the codes at any time, if there is reason to believe it is not producing correct results. These test problems will be used for each new version of MCNP that is transferred from Los Alamos to the Hanford Cray. The "t" at the end of the executable name in Figures 3 and 4 is to avoid any confusion with the corresponding script file name that might be in the path.

Figure 3. Command File probrun.sun to Execute 25 Test Problems for MCNP.

```
#! /bin/sh
# script to test ./mcnp.
tar -xvf testmctl.sun
tar -xvf testoutp.sun
tar -xvf testinp.tar
mcnpt name=inp01
diff inp01m mctl01 > difm01
diff inp01o outp01 > difo01
rm -f inp01r
[repeat above 4 card sequence for cases 02 to 25 with following exceptions:
  mcnpt name=inp08 rssa=inp07w          mcnpt name=inp22 rs=inp21w
  mcnpt name=inp25 rss=inp09w ]
rm diffm
cat difm* >diffm
rm diffo
cat difo* >diffo
```

Figure 4. Command File hanhrun.sun to Execute 11 Additional Problems for MCNPH.

```
tar -xvf ihan.tar
mcfunc()
  {
  cp $name$num inp
  rm outp
  mcnpht>l$name$num
  cat  l$name$num  >> results
  mv outp o$name$num
  rm runt*
  }
 name=ih
 rm wssa
 num=1;mcfunc
 cp wssa rssa
 num=2;mcfunc
 rm wssa
 num=3;mcfunc
 cp wssa rssa
 num=4;mcfunc
 num=5;mcfunc
 num=6;mcfunc
 num=7;mcfunc
 num=8;mcfunc
 num=9;mcfunc
 num=10;mcfunc
 rm wcol wsrc
 num=11;mcfunc
```

7.0 CONFIGURATION CONTROL PLAN/VALIDATION PROCEDURE

Below is a procedure for creating and verifying MCNP on the WHC computer platforms. This procedure is followed by the code custodian, or backup code custodian, every time a new version of MCNP is created. [See Appendix D for minor differences with DOS.]

1) Obtain the source master files (C and FORTRAN) for the current version of MCNP from Los Alamos or RSIC (see Sections 3.1 and 3.2).

2) Create two (patch) files for adapting the prpr compatible FORTRAN files to the WHC computer platforms for mcnp and mcnph, respectively. The first file contains the system dependent changes needed, mostly graphics adaptations, for creating the mcnp executable. The second file contains these system dependent changes plus the physics enhancement changes for the extended Hanford Version for creating the mcnph executable. Then create two (patch) files for adapting the prpr compatible C files to the WHC computer platforms.

3) Use the command file of Figure 1 to produce source code with prpr, and to compile and load mcnp on the Sun Workstation. Repeat this for the other platforms.

4) Use the command file of Figure 2 to produce source code with prpr, and to compile and load mcnph on the Sun Workstation. Repeat this for the other platforms.

5) Run the 25 LANL test problems for MCNP and MCNPH (see command file of Figure 3), and the additional 11 Hanford test problems for MCNPH (see command file of Figure 4), on each computer platform. Use these problems to verify that the executables perform as expected (see Section 6.0). The title card for each of the 25 LANL test problems briefly describes the problem. Figure 5 shows this title card for each of the 25 LANL test problems. The primary test for the LANL problems is a UNIX "diff" command, after execution of each of the 25 problems, to display any differences between the LANL output tallies and those generated with the new executables. The 25 difference files from the diff command are stored in bluegate in the directory /v93586/mcnp4a/prob/diff/ in tar format with file names d.sun, d.sgi, d.hp, and d.ibm for the four platforms , and with file names dh.sun, dh.sgi, dh.hp, and dh.ibm for the corresponding mcnph execution. After execution of the 11 Hanford test problems with MCNPH, a UNIX "grep random results" command is utilized. This is to verify that the same number of random numbers were obtained with the new executable as were obtained with our old executable (from previous ECN) for each problem. These compare files for the eleven Hanford test problems are stored in the same bluegate directory as stated above with file names gh.sun, gh.sgi, gh.hp and gh.ibm for archival reference. The title of each of the 11 Hanford test problems is given in Figure 6.

6) Issue an ECN for comments and sign-off.

7) After the ECN sign-off, notify users and replace the old version in Bluegate and in the public file directory /apps/mcnp/. Each new version is identified with a unique five digit identifier; i.e., 4a00m, where 4a denotes Version 4A, 00 is reserved for any LANL revisions of that version, and m is the Hanford change number that is typically a, but could progress to b,c... if changes were necessary in the Hanford code. Such new versions or changes would always be implemented by ECN. In addition to this five digit identifier being printed on each MCNP output file, the EDT number and the latest ECN number will also be printed.

8) The five digit identifier will be 4a00c for the files for this current ECN. The executable files for the defunct revisions will be stored with their appended five digit identifiers under the bluegate directory /v93586/mcnp4a/old/.

Figure 5. Title for Each of the 25 LANL Test Problems.

```
prob1 -- simple neutron problem to test some basic operations of mcnp.
prob2 -- three different tallies of the same physical quantity.
prob3 -- many features of the general source
prob4 -- photons
prob5 -- toroidal tokamak
prob6 -- cutoffs, flagging, and variance reduction features
prob7 -- generate surface source for prob8
prob8 -- use surface source from prob7
prob9 -- kcode in complicated cells and sdef
prob10 -- general test problem     /x6code/gtprob
prob11 -- intertwined super pretzels with s(a,b), mode n p
prob12 -- porosity tool model
prob13 -- check of the volume calculator, rotational symmetry case
prob14 -- test general source in repeated structures.
prob15 -- test filled lattice and skewed lattice.
prob16 -- test general source in a lattice.
prob17 -- kcode in a rectangular finite lattice.
prob18 -- kcode in a hexagonal prism lattice.
prob19 -- multigroup boltzman-fokker-planck version of electron problem prob20.
prob20 -- continuous energy electron version of problem prob19.
prob21 -- electron-photon problem - generates surface source for prob22.
prob22 -- electron-photon surface source read from prob 21
prob23 -- forward 80 group electron-photon detector chip problem
prob24 -- reflecting lattice.  15x15 at 3.75 w/o u-235 enrichment.
prob25 -- fission surface source from prob9
```

Figure 6. Title for Each of the 11 Hanford Test Problems.

```
prob01 -- test of write to wssa file with importances
prob02 -- test of read rssa file and check tally against prob01
prob03 -- test of write to wssa file with weight windows
prob04 -- test of read rssa file and check tally against prob03
prob05 -- test like prob01, but no delayed gamma-rays from fission
prob06 -- test for sampling direction of flight inward -- cylinder
prob07 -- test like prob01, but delayed gamma-rays from fission, no prompt
prob08 -- test of point kernel calculation
prob09 -- test of minimum and maximum z in cells
prob10 -- test of fission energy bias
prob11 -- test of writing random walk coordinates for tally
```

7.1 PROCEDURE TO FOLLOW IN CASE AN ERROR IS DISCOVERED IN THE CODE OR CROSS SECTION DATA BASE

In the event that an error is discovered in the executable version of mcnp and/or one of the cross section sets, and the code custodian agrees that an error exists, the appropriate changes will be made in the public version of mcnp in the /apps/mcnp/ disk file space and users will be notified that unapproved changes have been made. However, the official version of mcnp contained on the common file system (Bluegate) directory /v93586/mcnp4a/current/ will not be altered until an ECN has been filed and approved. At this time users will be notified of the approval.

By doing this, a user can use the validated version of the code at all times by asking the code custodian to retrieve it from archival storage. However, if an error is found to exist in the code, it can also be corrected quickly for the executables on the /apps/mcnp/ disk, allowing the user the option of using a code which is believed to be correct, but has not yet been fully approved.

8.0 HARDWARE REQUIREMENTS

One million to five million words of memory are typically required to accommodate the code and required cross sections. MCNP presently uses an expandable memory algorithm that will expand the amount of fast memory used by the code until all the cross sections have been stored in memory. Sufficient disk space is required to store the executable file, the required cross sections, the output file, and the problem dump file.

9.0 REFERENCES

1. Briesmeister, J. F., Editor, MCNP - A General Monte Carlo N-Particle Transport Code, Version 4A, LA-12625, Los Alamos National Laboratory, Los Alamos, New Mexico, 1993. The Radiation Shielding Information Center will send users this manual packaged under the RSIC COMPUTER CODE COLLECTION with RSIC number CCC-200, Radiation Shielding Information Center, Oak Ridge National Laboratory, 1993.
2. L. L. Carter and E. D. Cashwell, "Particle Transport Simulation with the Monte Carlo Method," TID-26607, ERDA Critical Review Series, U. S. Energy Research and Development Administration, Technical Information Center, Oak Ridge, Tennessee (1975).
3. Mann, F. M., Lessor, D. E., and Carter, L. L., "Processing of FENDL_PA/1.1, WHC-EP-0727, 1993.

APPENDIX A

THE MCNP HANFORD USER'S MANUAL FOR VERSION 4A.

I. INTRODUCTION

The MCNP Monte Carlo code (Briesmeister 1993) was originally developed for the simulation of neutron and photon transport utilizing a generalized-geometry along with a continuous-energy treatment of the cross sections. The code solves neutral particle transport including time dependence and may be used in any of three modes: neutron transport only; photon transport only; or combined neutron/photon transport, where the photons are produced by neutron interactions. The capability to calculate k-effective eigenvalues for fissile systems is also a standard feature. The upgrade to Versions 4, and later 4.2, included the option to do electron transport or to produce the approximate distribution of photons from the Thick-Target Bremsstrahlung (TTB) approximation. The upgrade to Version 4a concentrated on fixing a few problems with version 4.2 and adjusting to the increased emphasis on work stations. This has included:

- Allowing X-LIB graphics in Open Windows on UNIX computer platforms.

- Dynamic memory allocation at problem execution is now available on all UNIX computer platforms.

- MCNP can now be executed in parallel on a cluster of UNIX workstations using PVM (Parallel Virtual Machine) software.

- Releasing a DOS source code.

The generalized-geometry treatment was previously extended with Version 3B of the code to include a repeated structure capability. This makes it possible to describe only once some cells and surfaces of any structure that appear more than once in a geometry. This repeated structure capability allows the treatment of very complex lattice-type geometries with a minimum of input data.

MCNP also optionally allows for a multigroup treatment of the neutrons and photons. This is sometimes advantageous because of a dramatic reduction in cross section fast memory requirements, but does carry the potential for a sacrifice in accuracy due to self-shielding. The multigroup treatment is somewhat simpler and has the additional feature in MCNP of allowing the user an option of an adjoint treatment.

Visual display of the results of calculations and of the geometry is an important aspect. The code will optionally allow on-line graphical displays of the results of the calculations. This is in addition to the graphical package that will display any two-dimensional slice of the geometry the user may want to see.

The manual for MCNP (Briesmeister 1993) extensively discusses the physics and optimization biasing options that are available in MCNP. It also includes a complete description of the input data required to run a problem and the cross section data sets that are available. Some example problems are displayed in detail. This general manual is the main source of information for users at Hanford. The intent of the Hanford User's Manual in this Appendix is to only discuss items that differ from, or are extensions to, those described in the general manual.

Execution of mcnp, both interactively and in batch mode on computer platforms at WHC are discussed in Section II of this Appendix. Section III gives instructions for obtaining graphical displays of the geometry, or tallies on the various platforms. Section IV summarizes the additional input data required to exercise options allowed with mcnph, the extended Hanford Version of MCNP. Section V summarizes the changes that were made to obtain this extended Hanford version upgrade--if you are not interested in using the extended Hanford version, Sections IV and V will not be pertinent. This extended Hanford version optionally includes: the production of delayed photons from fission; allows for varying source weights in kcode calculations that are controlled by weight windows to enable a consistent biasing within the core; allows the user to optionally bias fission neutrons toward higher energies to reduce statistical uncertainties in tallies of threshold reactions; allows the user to optionally use a point kernel treatment for the photon transport; has some advanced angle-biasing "dxtran" optimization options for cylindrical and rectangular slots; and allows the source, collision, surface crossing, and tally coordinates during the random walk to be written to card image files -- typically for graphical display purposes. There is also a special feature on the "fs" card that allows the user to specify a three dimensional grid in rectangular, cylindrical, or spherical coordinates for a tally type 4.

II. EXECUTION OF MCNP

Interactive Mode

Most of the cross section database is in binary format in the disk public file directory /apps/mcnp/, and the mcnp executable reads the cross sections from this directory during initialization of a problem. The mcnp and mcnph command files for execution and a file named xsdir also reside in the /apps/mcnp/ directory. This xsdir file is used to identify the file name that contains each cross section set in a manner that is usually transparent to the user. Normal execution of Version 4A does not require that xsdir be in the user's local file. However, if there is a file called xsdir in the user's local file, it will be the xsdir that is used during the calculation.

The user's local directory must include a card image file that contains the problem input [see the general MCNP manual (Briesmeister 1993) particularly CHAPTER 3, DESCRIPTION OF MCNP INPUT]. The execution of mcnp then simply involves the following interactive command:

```
  /apps/mcnp/mcnp       (or /apps/mcnp/mcnph for the extended Hanford version)
```

The execution of mcnp will generate an output file with default name outp which can be printed or interrogated with a text editor. The execution of mcnp will also typically create a problem dump file runtpe and, depending upon the problem, may create (Briesmeister 1993) other files. The problem dump file runtpe typically requires a moderate amount of disk space, and the user should delete this file unless he intends to use it to make a continue run or to display plots using the mcplot module of mcnp. A subsequent continue run is rarely made in the interactive mode, but is often necessary in the batch mode for reducing statistical uncertainties.

The above execution line assumes that the input file in the user's local directory is named inp. If the user wants to use a name other than this default, or to change the names of any of the other default files, the following type of execute line can be used:

```
/apps/mcnp/mcnp inp=itest outp=otest
```

where itest is the name of the input file and otest is the desired name of the output print file. The above execute line will result in a dump file with the default name runtpe since it was not specified otherwise on the execution line. The name runtpe could have been changed to rtest, for example, by including "runtpe=rtest" on the execution line.

If an output file name is specified, and a file with that name already exists, the code will create a new file name by incrementing the last letter of the filename. Hence, if outp is specified and already exists, it will call the new output print file outq. If outq also exists, it will call it outr, etc.

The "/apps/mcnp/mcnp" execution command actually runs a command file called "mcnp". This script file checks to see what platform is involved [Sun, HP, SGI, or IBM] and then selects the executable for that machine. This is all transparent to the user.

During interactive execution the status of the calculation and certain on-line changes can be implemented with the <control-c> commands as described in the MCNP manual. However, on the HP platform the <control-c> commands only work at the HP console.

Batch Mode

Execution in batch mode is essentially no different than interactively as far as MCNP is concerned. A few details often take on increased importance in the batch mode, such as, maximum memory requirements for execution of the batch command file, the problem dump tape is often required and its name is typically included on the mcnp (or mcnph) execute line, and it is important to terminate the calculation in time to gracefully produce an outp file and to do any housekeeping chores.

MCNP utilizes dynamic dimensions. The code initializes the problem by reading the input data, which consists of the information in the inp file, and setting up arrays. This initialization is typically accomplished while using only about a million words of fast memory. Then, fast memory is expanded to the problem requirements for reading the cross sections one at a time and expanded again, if necessary, to run the problem. This expansion of memory is transparent to the user, but the user should conservatively request the maximum amount of memory that the job will require in the batch command file. The maximum memory required is approximately equal to the problem requirements for dynamic dimensions (see the "total" under "dynamically allocated storage" at the end of Table 1 in the output outp file) plus 800,000 words.

It is important that the execution of the random walks terminate with sufficient time remaining for a graceful conclusion of the problem. This conclusion includes: writing the final problem tally information on the output summary file, writing the problem information to a problem dump file, and performing whatever additional tasks are requested in the batch command file. A number of methods are available to assure that there is time left in the batch job for these housekeeping tasks. On some platforms, the code periodically checks to see how much time remains in the batch job and will terminate the job with ample time for a graceful exit in most cases. The user can also specify the total running time in minutes on the ctme card. With this option, the code will terminate by periodically checking the time used so far compared to the total running time requested on the ctme card, after some additional allowance for housekeeping tasks,.

The user can also specify the total number of random walks to be sampled on the nps card, for all problems except those involving the kcode card option. Then the problem will terminate after this specified number of histories, or by the previously discussed time termination, whichever comes first. For eigenvalue problems with a kcode card, the user can input on the 4th entry of the kcode card the number of cycles to run before termination.

Occasionally the user may desire to have a batch job (or background job) terminate gracefully before it reaches any of the above discussed time limits. For example, the batch job may not have finished overnight, but the user wants to look at the outp file for the number of histories that have been ran. The user can trigger graceful termination by going into the disk directory where the job is running and creating a dummy file called qmcnpnow -- might for example copy the input file to qmcnpnow. The code periodically checks for a file by this name, and will terminate when it finds that it exists. This qmcnpnow file should then be removed or the next mcnp execution within this directory will immediately terminate.

Continue Run

MCNP includes an option to read the information obtained from a previous problem and to execute additional random walk histories. This requires that the user has saved the runtpe problem dump file from the previous MCNP (or MCNPH) calculation. The continue run is then implemented with an inp input file like the example shown in Figure A.1. In a continue run, only certain data cards are allowed as discussed in Chapter 3 of the MCNP manual. In the example of Figure A.1, the only data card is a ctme card to tell the code to execute the problem for another 115 minutes.

Figure A.1. Example of an inp File for a Continue Run.

```
    continue
    ctme    115.
```

The entry c. must appear on the execute line for a continue run. For example, the line

```
  /apps/mcnp/mcnph c inp=icont outp=ocont runtpe=runa
```

would execute a continue run with an input file like that of Figure A.1 named icont, the runtpe file from a previous execution being named runa, and the output file for the continue run named ocont. It is the user's responsibility to furnish a batch command file that makes sure that the previous runtpe file is available, includes a "c" on the execute line, and renames files on the execute line as desired.

QA of MCNP and MCNPH on Multiple Machines Using PVM

MCNP can be executed in parallel on a cluster of workstations, and these need not be the same type of platform. For example, we have demonstrated that a predecessor to version 4A executed a problem satisfactorily on 2 Sun's, an HP, and a SGI. However, the public file version of 4A in disk directory /apps/mcnp/ will not have the pvm capability. The executables to utilize pvm on multiple machines will be directly under the control of the code custodian. This ECN will be the QA of these executables, where they will be tested by the code custodian on the standard test problems, as described in Section 6.0, and then the code custodian will work with individual users on a problem-by-problem basis.

Default Print of Statistical Analysis of Tallies

The default with no "print" card is to include the new statistical information on each tally in the outp file. Since this can lead to very large output files, this default has been changed in the Hanford version of mcnp and mcnph to not print with the exception of the generation-to-generation kcode information. This statistical information can be included on the outp file by including "160" on the print card. If a print card is part of your input file, and you want to suppress the statistical information, simply include "-160" on the print card. The following examples are given to help clarify:

Print Card in inp	Result on outp file
"no print card"	Only summary of statistical diagnostics
"print"	Full statistical diagnostics
"print 110"	Only summary of statistical diagnostics
"print 160 110"	Full statistical diagnostics
"print -110	Full statistical diagnostics
"print -160 -110	Only summary of statistical diagnostics

III. GRAPHICAL DISPLAY OF GEOMETRY AND TALLIES

Black and White Plots

Graphical plots may be obtained on any platform in an x window environment. However, for the HP, the user needs to be at an HP console. We give examples here for the mcnp executable, but the same examples hold for mcnph by simply replacing the mcnp executable (script file name) with mcnph.

The execute lines are simply

```
  /apps/mcnp/mcnp ip inp=inputfilename
```

to obtain geometry plots or

```
  /apps/mcnp/mcnp z runtpe=runtpefilename
```

to obtain plots of the tally information from a previous execution.

The screen will then respond with a "?" asking for input data for the plot. For a geometry plot, the user responds as in previous versions with input defining origin, basis, extent, etc., for the plot: see Appendix B of the new manual.[1] Since version 4A supports both color and black-and-white, the user can also specify which mode is desired, where "color on" is the default. An example input line for geometry plotting would be:

```
  ? or=10 10 200 b=1 0 0 0 0 1 ext=300 la=1 1 color off
```

for an x,z plot with the center of the plot at the (x,y,z) coordinates of (10,10,200), an extent (measured from the center) of 300 in both the horizontal and the vertical, and a plot showing both surface and cell numbers. In addition to the plot being displayed on the screen in a separate window after entering the above line, a postscript file called out.ps is produced that can be sent to the printer to obtain a hard copy. The typical command to print the black-and-white postscript file is simply

```
    lpr out.ps
```

This file will be overwritten when the next plot is generated.

Color Plots

Color plots will fill in the cells with color chosen according to the material being used in the cell. Such plots are particularly useful for generating transparencies or for identifying geometry errors. They are also helpful in quickly showing the materials being used in the cross sectional view. The above discussion for black-and-white plots also applies to color plots, except you can switch back to color from black-and-white by including "color on" as part of the instructions after the "?" for the new plot. Color is the default for the first plots and subsequent plots until a "color off" is included.

A postscript file called out.ps is also generated when the plot is in color, except the postscript file should now be printed on a color printer with an instruction like

```
   lpr -Pcps2 out.ps
```

Geometry Plots Under Batch

Plots can also be created in a batch mode. Batch plotting under UNIX can be quite useful for plots of very complex geometries, where it may take minutes or even hours to obtain the plot. To create geometry plots in a batch mode, begin with the usual shell script giving the batch output and error file names, memory requirements, cpu time, etc.. Then typically the following three cards would be included,

```
  cd /t_../yourdirectory
  /apps/mcnp/mcnp ip inp=inputfilename com=plotfilename
  lpr out.ps
```

where the first card switches to your local directory containing the inputfilename and the plotfilename. The second card executes mcnp to obtain the plot and the last line outputs the postscript file to the printer. Additional plots can be made by repeating these last two lines with a different plotfilename each time. When generating multiple plots in one batch job, the out.ps file should also written to another file before printing so that the out.ps file is not overwritten by the next plot before it gets printed.

Here inputfilename is the name of your MCNP input file and plotfilename contains two lines, where the first line is the instructions for the plot and the second line is "end". For example, this file could contain the two lines

```
  or=10 10 200 b=1 0 0 0 0 1 ext=300 la=1 1 color off
  end
```

Note that the "?" is not included in this file.

IV. ADDITIONAL INPUT FOR THE EXTENDED HANFORD VERSION, MCNPH

Compatibility With the LANL MCNP Manual

The Hanford Version of MCNP, aside from minor operating system differences, is compatible with the MCNP manual. This comprehensive manual (Briesmeister 1993) should be utilized for general information on physics and using the code. The detailed discussions in the MCNP manual will not be repeated here. We only include in this section short physics discussions, and the additional input that is required to utilize optional features in the extended Hanford Version of MCNP.

Additional Input for Extended Hanford Version of MCNP

The extended Hanford Version of MCNP optionally includes: the production of delayed photons from fission applicable to fast reactors; varying source weights in kcode calculations that are controlled by weight windows to enable a consistent biasing within the core; allows the user to optionally bias fission neutrons toward higher energies to reduce statistical uncertainties in tallies of threshold reactions; allows the user to optionally utilize a point kernel treatment for the photons; allows for special biasing of the direction of the source toward a cylindrical axis or toward a point; has some advanced angle-biasing "dxtran" optimization options for cylindrical and rectangular slots, and allows an option to write out the coordinates during the random walk for visualization purposes with commercial plotting packages. There is also a special feature on the "fs" card that allows the user to specify a three dimensional grid in rectangular, cylindrical, or spherical coordinates for a tally type 4.

The extended Hanford version recognizes all the input data of the standard code, but also optionally allows additional input in order to exercise these options. The additional input allowed is summarized in Figure A.2. These tend to be very specialized options that are only used in certain types of calculations. They are discussed in more detail later in Section V.

Some of the options will require more than one card type from the input described in Figure A.2. The point kernel option is implemented with the 1st entry on the idum card and, optionally, the 1st entry on the rdum card. The special source option for dynamically calculating the direction-of-flight reference vector from each sampled (x,y,z) point toward some central axis requires two entries on the rdum card in addition to the 4th entry on the idum card. The special source option for dynamically calculating the direction-of-flight from each sampled (x,y,z) point toward a fixed (x',y',z') point requires the 4th entry on the idum card and the 4th, 5th, and 6th entries on the rdum card. The high energy biasing of the fission source requires the 2nd and 3rd entries on the rdum card and at least two energy groups of neutron weight windows.

The option to write out coordinates of the random walk to files has been expanded to allow the option to only write those random walks that contribute to a tally. The coordinates of the random walks are now written to the files in binary rather than card image. A FORTRAN program, tcard.f, is available in /apps/mcnp/ that will read one of the binary files and write out the corresponding card image file on a (i2,1p6e12.5,5i12) format, if a card image file is needed. The binary files have the advantage that they require less disk space.

The default for flux-to-dose conversion factors for the photon point kernel option has been changed from the 1977 ANSI standard to the 1991 ANSI standard. However, the old 1977 conversion factors can be used with a "1" as the 8th entry on the idum card.

The location numbers on the idum and rdum cards may have changed since versions prior to 4A, so please check old input files carefully against Figure A.2.

Figure A.2. Optional Input for Extended Hanford Version.

Card Name	Location of Entry on Card	Default of Entry	Description of Entry and Option
idum	1	0	The default of zero is the normal treatment of photon transport. An entry of 1 will treat the photon transport with a point kernel using the ANSI/ANS-6.1.1-1991 fluence-to-dose conversion factors. An entry of -1 will treat the photon transport with a point kernel using the fluence-to-dose conversion factors from ISOSHLD. With the point kernel option the user must request f5 point detector tallies at the x,y,z points of interest, but should not usually include fluence-to-dose conversion factors. The code internally applies fluence-to-dose conversion factors to obtain mrem/hr dose rates at the detectors for a source strength in photons/second. Note: with the point kernel option the user should typically use an nps card to terminate the calculation and non-zero importances for all cells between the source and detector. An entry of 2 or -2 is just like the above description except the new buildup factors of ANS-6.4.3 are used. This utilizes the library file gpbuf from /apps/mcnp/ instead of buflib for the buildup factors. Effective z values of -1, -2, or -3 on the first entry of the rdum card will utilize buildup factors for water, concrete, or air, respectively, in ANS-6.4.3. The dimensions have been increased to allow for up to 2,000 point detectors.

idum	2	0	This entry controls the production of gamma rays by neutrons, and allows an approximate inclusion of delayed gamma-rays from fission products at fission events typical of a fast reactor. An entry of 1 reverts to their production as with the standard code. With an entry of 0, delayed gamma rays from fission events will also be included. An entry of 2 suppresses the production of all gamma rays from fission -- both prompt and delayed. An entry of -1 produces delayed gamma rays from the fission event, but suppresses gamma-ray production from all other sources. Typically the default of 0 is used for steady-state problems.
idum	3	0	Material-in-material option no longer allowed.
idum	4	0	An entry of 1, 2, or 3 implements a special option for selecting the direction-of-flight of the source, where the axis of the cylindrically symmetric source is in the x, y, or z direction, respectively. The (y,z), (x,z), or (x,y) coordinates of the center of the cylinder are input as the (5th and 6th), (4th and 6th), or (4th and 5th) entries on the rdum card, respectively. These, coordinates are used to determine a new vec for each source particle that points to this coordinate location from the sampled source position, where the axial coordinate is unchanged; i.e., points directly toward the center of the cylinder from the source point. The direction-of-flight is then sampled from the user specified dir distribution using this reference vector. The inp file must define an initial dummy vec with the axial component of zero or the code gives a fatal error. An entry of -1 implements a special option that calculates vec as the direction from each source coordinate sampled to the (x,y,z) point input as the 4th, 5th, and 6th entries on the rdum card, respectively. This is useful for sampling the direction-of-flight with respect to a vector that dynamically points to a fixed coordinate location in space. The user must input a dummy vec that is never used.

idum	5	0	Non-zero entries will optionally write coordinates during the random walk to binary files, where wsrc is the file for source coordinates and wcol is the file for more general information. These files can be used to make graphical displays of sources, collisions, etc. Each binary record contains the following 12 parameters in the order given: the random walk particle nps number; the x,y,z coordinates; the energy; the weight; the time; the problem cell number [leaving if a surface crossing or entering if a source surface]; the problem surface number [zero if a collision]; the problem material number in the cell; the problem tally number; and the tally segment number. The last two entries are zero unless a tally is being made, in which case the tally contribution rather than the particle weight is written. The floating point variables are written in single precision. The wsrc and wcol files become very large so the following Figure A.3 shows various options available for limiting the data written to these files.
idum	6	0	Write fission source points [x,y,z,E,cell #,wgt, parent nps #] to outp file in 7e15.7 format for a cylce equal to this entry. Default of 0 does not write.
idum	9	0	A non-zero entry only influences geometry plotting. Will plot the "n" value of the importance expressed as 2**n, with n the next lowest integer if it is not an integer value already. If the importance is zero sets n to -99 for the plot. This option is sometimes convenience for plotting large importance values since it has a smaller number of digits to put in each cell.
idum	10	0	=0, default, is normal dxtran. =1, Dxtran sphere within angle and without attenuation. =3, Dxtran cylindrical slot-- see dxt:n card. =5, Dxtran rectangular slot-- see dxt:n card. See idum(11), (12), rdum(8) for options when >0, and see dxc:n card discussion and Figure A.4 as well as the main "OPTIONAL DXTRAN..." discussion.
idum	11	0	used only if idum(10)>0. =0, default, do not play ww or importance games with dxtran particle until its 1st collision. =1, Play ww or importance games.

idum	12	0	used only if idum(10)=3 or 5. =0, default, play roulette to account for angle between vector from collision and normal to slot vector. =1, reduce particle weight instead.
rdum	1	0	This entry is used with the point kernel treatment [see the 1st entry on the idum card], and is the effective Z value to use between source and detector for the point kernel option. The default of zero on this rdum card means that the code will calculate the effective Z internally to obtain the buildup factor.
rdum	2	0	The neutron energy from fission events is biased above the energy of this 2nd entry on the rdum card (if non-zero) in order to reduce statistical precision for high energy neutron reactions. No high energy biasing is used for the default entry of zero. See the 3rd entry on the rdum card.
rdum	3	0	This entry is only used if the 2nd entry on the rdum card is non-zero. This 3rd entry is the probability ($0 < p < 1$) to be used to sample neutrons from fission above the energy specified on the 2nd entry of the rdum card. This biasing of the neutron energy from fission also requires the use of at least two energy groups of weight windows in the calculation. If this third entry is negative, a probability consistent with the weight windows being used is calculated by the code at each fission event, but this probability is subject to the requirement that the probability also be constrained to a value less than the absolute value of this third entry: the default of zero sets the entry to -0.5.
rdum	4	0	See the 4th entry on the idum card for special source option. This is the x value for the reference vector direction -- not used if the 4th entry of idum is 1.
rdum	5	0	See the 4th entry on the idum card for special source option. This is the y value for the reference vector direction -- not used if the 4th entry of idum is 2.
rdum	6	0	See the 4th entry on the idum card for special source option. This is the z value for the reference vector direction -- not used if the 4th entry of idum is 3.

rdum	7	0	An entry greater than zero implements a special option to determine the approximate z-min and z-max value for each cell from the random walk particles. Will print all cells for dz of the cell greater than this entry.
rdum	8	0	Probability for playing roulette with dxtran particle if its 1st collision is in cell where it was born. Default of 0.0 is to let code use dxtran weight multiplier as this probability for roulette.

Figure A.3. Information Written to src and wcol Files with Idum(5) Non-zero.

idum(5)	Source Coordinates to wsrc File	Coordinates Written to wcol File			
		Source	Surface	Collision	Tally
1[a]	yes	no	no	no	no
2[a]	yes	yes	yes	yes	no
3[a]	yes	no	no	yes	no
4[a]	no	no	no	yes	no
5[a]	yes	yes	yes	yes	yes
6[a]	yes	no	no	yes	yes
7[a]	no	no	no	yes	yes
8[a]	no	no	no	no	yes
9[a]	no	no	yes	no	yes
10[a]	no	yes	no	yes	no

a) The negative of this entry will only write the coordinates of those random walks that contribute to tallies. Normally this is used with only one tally in the input file. For idum(5)=-5 to -8, only the coordinates at entrance to the tally cell will be written with the weight [6th entry written to wcol] replaced with the tally contribution [weight * t * N * sigma, with t the track length to a collision or the cell boundary, N the first entry norm factor on the fs card, and sigma the microscopic cross section specified on the fs card]. If a collision occurs within the tally cell, the coordinates at the collision site will be written with the weight [6th entry written to wcol] replaced with the tally contribution [weight * t * N * sigma, with t the track length to the next collision or the cell boundary, N the first entry norm factor on the fs card, and sigma the microscopic cross section specified on the fs card], etc.

If idum(10) is zero (normal dxtran) or 1 (dxtran sphere without attenuation), the dxt card has the meaning described in the MCNP manual. If idum(10) is 3 or 5, the required entries on the dxt card are given in Figure A.4. If idum(10)>0, the entries on the dxc card are interpreted a little differently. If an entry on the dxc card is zero, dxtran particles will not be created for collisions within the cell. If an entry is non-zero, dxtran particles will be created unless checks against other conditions (see section on Optional DXTRAN Angle Biasing Treatments) indicate otherwise. For entries of p with p>0 but <1, the DXTRAN particle is further created with probability p and the weight of the dxtran particle is the divided by p. For idum(10)>0 and more than one dxtran sphere (or slot), only one probability on the dxc cards can be >0 for a given cell. Also for idum(10)>0, assuming dxtran particle creation is occurring at the collision, the normal particle from the collision is not allowed to enter the solid angle of the dxtran sphere (or slot); i.e., its history is terminated if it does and normal particles are not terminated when they reach the dxtran sphere (or slot) in order to keep a fair game.

Figure A.4. Information Required on dxt Card for Cylindrical or Rectangular Slot (idum(10)=3 or 5).

idum(10)	Entries #'s on dxt card	Description
3 or 5	1 to 3	(x,y,z) center of dxtran bias slot.
3 or 5	4 to 6	(u,v,w) positive direction of current -- must be normal to slot.
3 or 5	7	Minimum value of cosine of angle between vector from collision to center of slot and (u,v,w) of normal to slot -- entries 4 to 6. Default is 0.17305 (80 degrees), although typically the cosine of a few degrees would be more representative. <=0 not allowed.
3 or 5	8	Minimum distance axially below slot for collisions for dxtran (default = 1 cm) -- may not be zero. If <0, interprete as moving the center of the slot a distance up or down so that the vertical distance of the collision below the slot is the absolute value of this entry; i.e., the solid angle for the dxtran biasing is constant for collisions a given distance radially from slot axis.
3 or 5	9	Maximum distance axially below slot for dxtran -- must be greater than the 8th entry unless the 8th entry <0 in which case this entry is not used.
3	10 and 11	Radii from the slot axis for the inner dxtran portion of the slot with higher probability density. Inner and outer radii, respectively, where the 10th entry may be zero (like a sphere) and the 11th entry must not be less than the 10th.
3	12 and 13	Radii from the slot axis for the outer dxtran portion of the slot -- inner and outer radii, respectively. The 12th entry must be less than or equal to the 10th and the 13th entry must be greater than or equal to the 11th.
3	14	The maximum azimuthal angle in radians for the inner dxtran portion of the slot -- must be >0 and < or = π.

3	15	The maximum azimuthal angle in radians for the outer dxtran portion of the slot -- must be more than or equal to entry 14 and < or = π.
5	10	Half thickness of inner rectangular slot in "thin" direction -- must be >0.
5	11	Half thickness of inner rectangular slot in "thick" direction -- must be >0.
5	12	Half thickness of outer rectangular slot in "thin" direction -- must be more than or equal to entry 10.
5	13	Half thickness of outer rectangular slot in "thick" direction -- must be more than or equal to entry 11.
5	14 to 16	(u,v,w) of thin direction of rectangular slot -- must be normal to entries 4 to 6. Code computes (u,v,w) for thick direction.

SPECIAL FS TALLY IN RECTANGULAR, CYLINDRICAL, OR SPHERICAL MESH:

A special fs tally is available in mcnph for tally type 4 that allows one to divide space into a three dimensional grid -- the standard fs tally is more restricted. In rectangular coordinates, for example, specifying kx surfaces normal to x, ky normal to y, and kz normal to z defines a grid of (kx-1)*(ky-1)*(kz-1) parallelepipeds. Such a grid may be much more efficient to obtain dose rate contours than using hundreds or thousands of point detectors.

The allowed entries on the special fs card are summarized in Figure A.5, and there is an elaboration of the meaning of the input in Figure A.6. The first entry on this special fs card specifies the type of coordinate system so all other information is for this type of coordinate system. The 2nd, 3rd, and 4th entries specify the number of surfaces being used along each coordinate to define the mesh. The surfaces numbers are then entered in increasing order of the relevant variable [increasing values of planes for rectangular coordinates or radii of spheres in spherical coordinates, for example]. All coordinates must be defined with respect to the x,y, or z axes; i.e., planes parallel to these axes for rectangular coordinates, or the axial of cylindrical coordinates, or the polar angle in spherical coordinates must be defined with respect to an x,y, or z axes, and the azimuth angle in either spherical or cylindrical coordinates with respect to one of these axes that is consistent with the other definitions.

The surfaces numbers on the fs card for the 2nd (polar) coordinate in spherical coordinates must be entered so that the cosine of the polar angle increases from something greater than -1 to something less than +1. This is accomplished with conical surfaces, except for the required surface at 90 degrees, which will be a simple plane. Each surface number of these cones is required to appear twice on the fs card, where one leg of the cone splits up the mesh beyond 90 degrees and the other leg less than 90 degrees. The angle is actually entered on the MCNP surface card as the square of the tangent of the angle.

The surfaces for the 3rd (azimuth) coordinate in either cylindrical or spherical coordinates is entered with plane surfaces, where this will be a "p" surface if the angle is not a multiple of pi/2 and it will be a "px", "py", or "pz" surface otherwise -- one of these surface types will not be allowed since it would be normal to the azimuth direction.

Figure A.5. Definition of Entries on fs Card for Special Tally Mesh.

Entry Number on fs Card	Coordinate System for Mesh		
	Rectangular	Cylindrical	Spherical
1	-2000000	-3000000	-4000000
2	kx	kx	kx
3	ky	ky	ky
4	kz	kz	kz
5 to (4+kx)	Mesh problem surface numbers for 1st coor.	Mesh problem surface numbers for 1st coor.	Mesh problem surface numbers for 1st coor.
(5+kx) to (4+kx+ky)	Mesh problem surface numbers for 2nd coor.	Mesh problem surface numbers for 2nd coor.	Mesh problem surface numbers for 2nd coor.
(5+kx+ky) to (4+kx+ky+kz)	Mesh problem surface numbers for 3rd coor.	Mesh problem surface numbers for 3rd coor.	Mesh problem surface numbers for 3rd coor.

Figure A.6. Details Regarding fs Card for Special Tally Mesh.

	Coordinate System for Mesh		
	Rectangular	Cylindrical	Spherical
Surface signs	+	+	+
Coordinate order	Any order	(axial,radial, azimuth angle)	(radial,cos[polar angle],azimuth angle) -- radial may have ellipses
Surface order for a coordinate	Most negative to most positive	Most negative to most positive for axial; increasing radius for radial; increasing radians in counter clockwise direction for azimuth (1st surface >0 radians)	Increasing radius for radial; increasing cos() for polar; increasing radians in counter clockwise direction for azimuth
Reference vector for angles	--	+y if axial is x; +z if axial is y; +x if axial is z. First surface must have azimuth angle >0 radians. Last surface may be zero.	Polar angle reference is major axis of ellipses. Arbitrary if all spheres. Azimuth is like cylindrical except polar angle reference is like axial in cyl. coor.
Constraints	kx>1 ky>1 kz>1	kx>1 ky>1 kz=even (may be 0) Each azimuth surface number must appear twice on fs card (a mesh surface at <=180 degrees and at >180 degrees)	kx>1 ky>2 and odd kz=even (may be 0) There must be one surface plane at 90 degrees, and each of the other polar surface numbers must appear twice on fs card. Each azimuth surface must appear twice on fs card (a mesh surface at <=180 degrees and at >180 degrees)

Number mesh intervals	(kx-1)*(ky-1)*(kz-1)	(kx-1)*ky*kz or (kx-1)*ky if kz=0	kx*(ky+1)*kz or kx*(ky+1) if kz=0

The origin of the coordinate system must be consistently defined for the cylindrical or spherical coordinate systems. The cones and plane defining the polar angle mesh in spherical coordinates must all pass through the center of the spherical radial surfaces. Similarly, the planes defining the azimuth boundaries must all cross the center of the cylindrical radial surfaces for cylindrical coordinates or the spherical radial surfaces for spherical coordinates. For a cylindrical coordinate system with z-axis (or a spherical coordinate system with z-cones to define the polar angles) that has radial center at (x_o,y_o), we require that the first entry on the "p" card for an azimuth plane be sin(s), where s is the counterclockwise angle between the x-axis and the first-crossing of the plane. The second entry on the "p" card is -cos(s), the third entry is zero [plane normal to z-axis], and the fourth entry is given by

$$D=x_o \sin(s) - y_o \cos(s)$$

For x being the reference axis rather than z, x-->y and y-->z in the above equation for D, the first entry on the "p" card is zero, the second entry is sin(s), and the third entry is -cos(s). For y being the reference axis rather than z, x-->z and y-->x in the above equation for D, the first entry on the "p" card is -cos(s), the second entry is zero, and the third entry is sin(s).
The order of the surface numbers for these azimuth planes on the fs card must be in increasing angle s from the smallest s>0 degrees to the s<=pi. Then these surface numbers are repeated in the same order for angles beyond pi, where the last surface number may optionally be a simple plane for 2pi.

Ellipses, rather than spheres, or a combination of ellipses and spheres can be used in the spherical coordinate system definition of the mesh. This may be used to define a mesh for a central cylindrical source by using ellipses near the source that have a large major axis and a small minor axis. Then there could be a transition at larger radii from the ellipses to spheres to obtain the desired spherical solid angles at large spherical radii for the mesh at large distances from the source. The use of ellipses must follow the rules:

- The major axis must always be along the same axis, and the polar angles of the mesh must be defined with respect to this same axis -- i.e., cones with respect to this same axis.
- The ellipses and spheres must not cross each other.
- Each ellipse must be defined with the SQ card, where one of the coefficients (A, B, or C) defines the major axis of the ellipse and is equal to the inverse square of the major radius -- the other two coefficients define the minor axis of the ellipse and are equal to the inverse square of the minor radius. The D, E, and F coefficients must be zero and the G coefficient must be -1. The (x,y,z)-bar origin values for the ellipse must be consistent with the origin of any cones that define the polar angles and with the origin of any spheres.

The output of the tally on the outp file is in order of the 1st coordinate of the mesh varying most rapidly and the 2nd coordinate varying next, where the 3rd coordinate mesh layer number is given before the corresponding tally values. The tally beyond the mesh is also printed in the outp file, but not in the mctal file. The data for this special fs tally in the mctal file starts with a "vals" card as with all other tallies. However, included on the "vals" card are 6 additional integers: the number of tally (volume) elements, the coordinate system used for the tally (such as -2000000), the number of surfaces in each of the 3 coordinates entered on the fs card to define the mesh, and the number of energy bins (usually 1 which includes the usual flux-to-dose conversion factors on de and df cards). Again the order is with the 1st coordinate of the mesh varying the most rapidly, but only the numbers are given on the mctal file -- not the detail that is on the outp file.

The volumes of the mesh elements are also given in the outp file at the first of the tally print in the same order with the 1st coordinate varying most rapidly. Also printed in the initiation is the total volume of the entire mesh for checking purposes -- this can be obtained from the outp file with the grep command, grep "Volume" outp. An output file called qmesh is generated during the MCNP initiation. It contains approximate average coordinate values for each mesh element, which may be useful for displaying contour plots.

The input required for the special fs card is best illustrated by simple examples. MCNP input file examples are shown in Figures A.7 to A.9 for rectangular, cylindrical, and spherical mesh definitions, respectively, for tally 4. All of these are void calculations for simple testing of the fs card with known sources for computing volumes, so the answer is also known. A series of 14 of these kind of test problems were used to check the implementation of this special fs option. These 14 test problems are executed with the script file fsrun.sun.

The code will calculate the volumes of the mesh intervals, under the assumption that the tally is being specified to include the volume of all of each mesh interval. You can optionally input the volumes on the sd card to override the code calculation. The order on the sd card is the same as discussed above for the tally output; i.e., 1st coordinate of the mesh varying most rapidly. You can use "j" entries for mesh volumes where you want to use the code computation of the volume element. The total number of entries on the sd card must exactly equal the number of mesh elements -- see the bottom of Figure A.6.

Unique surface numbers should be used to specify the mesh; i.e., **do not use surfaces that are used to specify the cells**. If you want the same value entered on a surface for the cells, change it slightly in the 5th place or something. If universes are involved, the fs surfaces are in the local coordinate system of the universe the particle is in. Hence, for an outside volume of cells, inside

Figure A.7. Example Input File for Special fs Tally in Rectangular Coordinates.

```
fs test, spr source on box, with special fs tally, ifs4
    1     0  -1
    9     0  1

    1        so  88.0000
    2        px  -50.0000
    3        px  -10.0000
    4        px  10.0000
    5        px  50.0000
    6        py  -50.0000
    7        py  -10.0000
    8        py  10.0000
    9        py  50.0000
   10        pz  -50.0000
   11        pz  -10.0000
   12        pz  10.0000
   13        pz  50.0000

mode  p
imp:p  1          0          $ 1, 10
phys:p   j 1
nps     10000
prdmp    j j 1
print
sdef     sur=1  nrm=-1 dir=d1 erg=d7 wgt=24328.5
si1    0 1
sp1    -21 1
si7      1  4.8  5.2
sp7         .20  .80
fc4      flux within special fs rectangular mesh
f4:p    1
fs4    -2000000 4 4 4   2 3 4 5  6 7 8 9  10 11 12 13
e4      4.9 5.5
```

Figure A.8. Example Input File for Special fs Tally in Cylindrical Coordinates.

```
fs test, cyl in x, plane source , with special fs tally, ifsl4x
    1     0  -1
    9     0  1

    1      s    0. 35. 60.   100.0000
   21     c/x  35. 60.    8.
   22     c/x  35. 60.   16.
   23     c/x  35. 60.   20.
   24     c/x  35. 60.   25.
   31     p  0.  .8660254  -.50000  0.310889
   33     p  0. .50000   .8660254 69.461524
   34     pz   60.
   10        px  -50.0000
   11        px  -10.0000
   12        px  30.0000
   13        px  50.0000

mode  p
imp:p  1          0          $ 1, 10
phys:p   j 1
nps     10000
prdmp    j j 1
print
c       cylindrical source along cylinder
sdef     pos=0. 35. 60. rad=dl ext=d2 wgt=3927.0 vec=1. 0. 0. dir=d4 erg=d7
         axs   1. 0. 0.
sil      0. 25.
spl     -21 1
si2     -55. 55.
si4     1 -1. 1.
sp4        .5 .5
si7      1  4.8 5.2
sp7         .2 .8
fc4      flux within special fs cylindrical mesh
f4:p    1
fs4      -3000000  4 4 6   10 11 12 13  21 22 23 24
                31 33 34 31 33 34
e4      4.9 5.5
```

Figure A.9. Example Input File for Special fs Tally in Spherical Coordinates.

```
fs test, ell in z, source inward , with special fs tally, ir34z
   1     0  -1
   9     0  1

   1          s  35.0000 60.0000 0.0000 52.0000
  11        k/z  35.0000 60.0000 0.0000 3.0000 0.0000
  12         pz  0.0000
  31          p  0.8660254 -0.5000000 0.0000000 0.3108890
  32         px  35.000000
  33          p  0.8660254 0.5000000 0.0000000 60.3108890
  34         py  60.000000
  21         sq  0.015625 0.015625 0.00390625 0.0000 0.0000 0.0000 -1.0000
                 35.0  60.0000 0.0000
  22         sq  0.00390625 0.00390625 0.00097656 0.00000000 0.00000000
                 0.00000000 -1.00000000 35.00000000 60.00000000 0.00000000
  23     s   35.0 60.0 0.0  40.0
  24     s   35.0 60.0 0.0  50.0

mode  p
imp:p  1          0          $ 1, 10
phys:p    j 1
sdef      sur=1  nrm=-1 dir=d1 wgt=8494.86
si1    0 1
sp1    -21 1
nps    1500000
prdmp     j j 1
print
fc4       flux within special fs spherical mesh
f4:p     1
fs4       -4000000  4 3 8   21 22 23 24  11 12 11
                 31 32 33 34 31 32 33 34
fc14       standard volume calculation inside sphere = 73,622.2 cm
f14:p      1
sd14       1.
```

of which is a universe, the usual procedure would be to do the calculation for a tally over that portion of the mesh beyond the universe where the mesh boundaries exactly (to within small deltas for making the mesh surfaces unique) match at the universe boundary. Then the calculation is made with the tally inside the universe, where the appropriate transformations and rotations are made of the mesh so that each mesh interval location corresponds to the desired mesh location in the global coordinates. Then the tallies from the two calculations must be merged using the tally from the first calculation for the mesh intervals outside of the universe, and the tally from the second calculation for the mesh intervals inside the universe.

The code does a certain amount of error checking, but the user should enter the fs card and surface coefficients with care and confirm with geometry plots -- the MCNP visual editor can be used to obtain plots of the mesh. A series of

V. UPGRADE TO OBTAIN THE EXTENDED HANFORD VERSION OF MCNP

The patch (see Section 3.3, Producing Source Code, Compiling, and Loading MCNP on Computer Platforms) file, ph.4a00c, for the extended Hanford version includes both the changes that make Version 4A of MCNP operational on the Hanford Cray, and the additional coding to implement the options for the extended Hanford Version of MCNP. The implementation of each option with the information in the bluegate file /v93586/mcnp4a/current/ph.4a00c will be discussed briefly. The prpr processor that merges these upgrade changes with the base code requires that the change cards be in sequence with the main program file so that a given option may be implemented at a number of different places in the patch file. To improve readability, the beginning of each set of changes for a different option in the patch file is signaled with a comment card having the form "*/ message regarding the changes immediately following". Comment cards within the file further explain the implementation of the options.

Variable Source Weights for kcode Calculations

The eigenfunction (kcode card) iteration for the fission source of a critical or near-critical system in MCNP utilizes constant weights of source neutrons during each generation. This can lead to problem optimization inconsistencies if weight windows are being used to bias histories into one portion of the reactor. The constant source weights tend to partially cancel out what one is trying to do with the weight windows. The extended Hanford version of MCNP allows for variable source weights whenever weight windows are being used.

Card insertions for this option start with the comment "*/ allow for variable source weights for kcode". These changes involve including the neutron weight in the next-generation parameters that are saved for each source neutron. If importance are used rather than neutron weight windows, all of these weights will be constant and the treatment is essentially unchanged from the standard code. If neutron weight windows are used in the kcode calculation, fission sources for the next-generation are created proportional to the frequency of fission events in the random walk and the weight assigned is proportional to the weight of the random walk particle involved in the fission event. The proportionality factor is chosen to stabilize the number of source fission neutrons per generation to approximately the user specified number.

Users are cautioned to not bias too heavily with weight windows in eigenvalue problems since the biasing can influence the reliability of the fission source distribution for the full core.

Optionally Create Delayed Gamma Rays from Fission

The standard code only produces the prompt gamma rays at fission events in coupled neutron-photon problems. The extended Hanford version includes the delayed gamma rays from fission products as the default and provides options for the user to selectively revert to the old treatment, suppress all gamma-ray production from fission events, or suppress the prompt gamma rays from fission events but produce the delayed gamma rays. Gamma-ray production from events other than fission are treated as in the standard code; i.e., only prompt gamma rays are included and even prompt gamma rays are not included for certain isotopes for which the prompt gamma-ray production information is not in the cross section data base (see Appendix G of the manual). For some isotopes, ENDF/B-V lumps photon production above about 1 MeV together rather than having an individual treatment for each type of reaction. For these cross section sets the option to suppress prompt gamma rays from fission events will not work at these high energies.

Card insertions for this option start with the comment "*/ delayed gamma from fission option". Most of this modification involves the subroutine dgamf, which produces the delayed gamma rays at fission events and writes them to the bank. This subroutine includes data statements that give the intensity and energy distribution of the delayed gamma rays from fission products integrated over all subsequent time as given in Table A.1. This information was obtained from the ADENA computer code (George, et al. 1982), which uses aggregate data derived (where possible) from experiments rather than relying entirely upon the information in the ENDF/B-V fission product files.

Tally 7 Changed to Print Average Energy in MW Rather than MeV

For most of our applications, Tally 7 is useless since it does not include delayed contributions from fission, and is also inconvenient since the units are MeV rather than MW. Tally 7 has been changed, by modifying the appropriate data statements to give MW of energy from the fission events under the assumption that all fission events ultimately lead to an average of 212 MeV/fission. This is the correct conversion factor for the FFTF mixed oxide core, and will only require a small adjustment of the printed output for this tally for other cores. By using Tally 7 with a sum over all cells, the fission power for the reactor is readily available.

Card insertions for this option start with the comment "*/ norm f7:n tally with 212 mev/fission and print in MW".

Table A.1. Delayed Gamma Rays From Fission Events Integrated Over All Time.

Group	Upper Group Energy (MeV)	Delayed Photons (Photons/Fission)						
		^{235}U[a]	^{238}U[a]	^{239}Pu[a]	^{240}Pu[a]	^{241}Pu, ^{241}Am[b]	^{232}Th[a]	^{233}U[b]
1	15.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2	10.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
3	9.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
4	8.0	0.0001	0.0001	0.0001	0.0001	0.0001	0.0003	0.0
5	7.0	0.0003	0.0004	0.0002	0.0002	0.0002	0.0007	0.0
6	6.0	0.0044	0.0040	0.0019	0.0021	0.0022	0.0109	0.0080
7	5.0	0.0206	0.0199	0.0091	0.0098	0.0108	0.0373	0.0296
8	4.0	0.0383	0.0421	0.0235	0.0254	0.0281	0.0573	0.0417
9	3.5	0.0442	0.0485	0.0271	0.0294	0.0325	0.0662	0.0481
10	3.0	0.0966	0.1087	0.0686	0.0736	0.0814	0.1441	0.3301
11	2.5	0.1873	0.1924	0.1323	0.1411	0.1520	0.2802	0.3108
12	2.0	0.2241	0.2333	0.1800	0.1861	0.1979	0.2671	0.3169
13	1.6	0.7270	0.8040	0.5280	0.5670	0.6289	1.0180	0.4737
14	1.2	0.7560	0.8567	0.6202	0.6398	0.6963	0.8169	0.4739
15	0.90	1.7320	2.0210	1.4780	1.5240	1.6480	1.8090	0.6608
16	0.60	1.2520	1.6470	1.2000	1.2910	1.4780	1.2540	0.6173
17	0.40	1.3900	2.0790	1.4450	1.6240	1.9990	1.6600	0.8260
18	0.21	0.7502	1.3580	0.8586	0.9473	1.2600	0.8887	1.1040
19	0.12	0.6575	1.1210	0.8633	0.9018	1.1170	0.6417	0.6070
20	0.07	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Total Number		7.8850	10.5410	7.4360	7.9630	9.3320	8.9530	5.8480
Total Energy (MeV)		5.8440	6.9110	4.8030	5.0950	5.7060	7.1530	5.1240

[a]Based on fast (fission energy) neutron induced fission.
[b]Based on thermal neutron induced fission.

Optionally use a Point Kernel Treatment for the Gamma Rays

Point Kernel with ISOSHLD Dose Buildup Factors

Card insertions for this option start with the comment "*/ point kernel option". A mathematical description of this option follows.

A source point is sampled and assigned a weight W, which is typically the source weight entry on the sdef card. This sampling of the source is with the standard MCNP source routines that are unchanged from the standard code. When using the point kernel option for photons, the dose contribution T at a point detector from this point photon source is calculated as

$$T=W*U*B*d$$

(A.1)

where d is the flux-to-dose conversion factor in (mrem/hr)/(photon/cm2/sec) for the energy of the source photon under the assumption that the weight W is the total source strength in photons/second, B is the ISOSHLD dose buildup factor, and U is the uncollided photon flux,

$$U = \frac{\exp(-L)}{(4*\pi*r^2)}$$

(A·2)

from a unit source with L the number of mean free paths between the source point and the point detector and r the distance from the source point to the detector point.

The dose buildup factor is a function of the effective Z of the material between the source point and the point detector, the number of mean free paths L between source and detector, and the energy of the source photon. It is calculated as in ISOSHLD by interpolation in the buildup factor tables.

The above formulation is consistent if the buildup factor is defined as

$$B = \frac{[\int \Phi d(E') \, dE']}{[U * d(E)]}$$

(A.3)

where E is the energy of the source, d(E) is the flux-to-dose conversion factor at energy E, U is the uncollided flux due to a unit source, and Ø(E') is the flux at the point detector due to this unit source.

By substituting the B defined in Equation A.3 into Equation A.1 the U and d cancel out so we obtain

(A.4)

$$T = W * [\int \Phi (E') \, d(E') \, dE']$$

i.e., the dose rate from a point source is the total source strength multiplied by the integral of the product of the energy dependent flux at the position of the point detector and the energy dependent flux-to-dose conversion factor.

Transport calculations are utilized to evaluate the dose buildup factor from Equation A.3 at a sufficient number of values for E, L, and Z so that it can be determined to the required accuracy by interpolation. Note in the above formulation, one could choose to define U by different ways as long as its use in the evaluation of the buildup factor with Equation A.3 is the same as the final utilization in Equation A.1. Goldstein appears to not include coherent scattering in the computation of L; i.e., treats it as straight ahead at the higher energies and it is comparatively small at lower energies. The point kernel treatment in MCNP includes the coherent in the evaluation of U, which is consistent with the treatment in ISOSHLD.

The above discussion assumed an effective Z for use with the buildup factor. The point kernel treatment in MCNP will automatically compute an effective Z for each segment of the flight path from source to detector as the sum of the products of the Z for each element in the material and the macroscopic total cross section of that material. Then for layers of different materials, an effective Z for an entire flight path is computed as

(A.5)

$$Z = \frac{\sum_i Z_i \ [\{L_{i-1}\}^{0.5} - \{L_i\}^{0.5}]}{\sum_i \ [\{L_{i-1}\}^{0.5} - \{L_i\}^{0.5}]}$$

where Z_i is the Z of the ith segment and L_i is the number of mean free paths from the end of the ith segment to the point detector, and the sum is over the last ten segments or the last ten mean free paths, whichever occurs first. An alternative is for the user to input his own effective Z as the first entry on the rdum card in which case the computation of Equation A.5 will not be made.

The point kernel treatment described above can be replaced with a transport treatment by simply changing the 1st entry on the idum card back to a zero and changing to whatever tally is desired for the transport calculation.

The source often contains a small high energy component that dominates the contribution through the shield. ISOSHLD automatically takes care of this with the fixed energy mesh. For such problems with MCNP it will typically be necessary to bias the selection of the source energy in the point kernel treatment in order to reduce statistical uncertainties from these higher source energies. You can get information on a near optimal biasing by making a short point kernel execution using your guess on an sb card regarding an optimum biased sampling in energy. This short execution should include an e5 card to define energy tally bins corresponding to the source energy bins. These tallies then become the entries for the sb card for the final execution. If the short execution still has large uncertainties, more than one iteration may be required to obtain a near-optimum sb card.

This discussion of the point kernel option has assumed a photon problem. For a coupled neutron-photon problem the same considerations apply only the photon source in Equation A.1 is from the photons produced at a neutron collision. The neutron source strength on the sdef card is in neutrons/second in order to obtain the correct units of mrem/hr on the photon point detector tally.

Point Kernel with ANS-6.4.3 Dose Buildup Factors

This point kernel treatment is just like that described previously except that the buildup factors are taken from ANS-6.4.3. These buildup factors are tabulated[3] for Be, B, C, N, O, Na, Mg, Al, Si, P, S, Ar, K, Ca, Fe, Cu, Mo, Sn, La, Gd, W, Pb, and U using geometric progression coefficients. These buildup factors are interpolated in Z, where the buildup factors for Be are used for Z<4 and the buildup factors for U are used for Z>92. ANS-6.4.3 also has buildup factors for the materials water, concrete, and air. The user can utilize these material buildup factors for a calculation by specifying -1, -2, or -3, respectively, as the Z-bar input entry [1st entry on rdum card]. The buildup factors for this material will then be used in the point kernel treatment irrespective of the materials in the geometry.

The buildup factors do not include the extra energy points to provide detail near absorption edges where buildup factors rise rapidly to very large values; i.e., they are based upon the subroutines of Daniel[3] that utilize a standard energy grid. An upgrade to include extra energy points may eventually be incorporated, especially if the RSIC library is upgraded with subroutines that do this.

<u>Test Problems for Point Kernel</u>

The problems that were used to check the point kernel option are summarized in Table A.2. Shown in Table A.2 are comparisons for the point kernel calculations based upon both ISOSHLD and ANS-6.4.3 buildup factors as well as comparisons to transport with either version 3b or version 4. The version 4 calculations all included straight ahead Bremsstrahlung except for the one footnote case, where a separate calculation was also made without Bremsstrahlung.

Additional comparisons between MCNP transport and point kernel are made in a paper to be given at the 8th International Conference on Radiation Shielding to be held in April, 1994.

Table A.2. Point Kernel Treatment Compared To Transport Calculations. (Point Source in Spherical Geometry.)

Material	Source Energy (MeV)	Uncollided Mean Free Path Thickness	Dose Rate (mrem/hr)			
			Isoshld Point Kernel	ANS-6.4.3 Point Kernel	Transport Version 3b	Transport Version 4
Lead	0.15	15.0	1.93E-4	3.54E-4	5.94E-4 (2.5%)[a]	5.59E-4 (3.8%)
	2.0	12.5	0.0844	0.0912	0.109 (1.9%)	0.104 (1.9%)
	8.0	12.5	0.305	0.549	0.248 (5.3%)	0.462(3.9%) 0.273(4.9%)[b]
Iron	0.15	25.7	1.75E-12 1.23E-12[c]	5.41E-14 3.79E-14[c]	7.69E-13 (18%)	8.19E-13 (21%)
	2.0	--	6.74	6.78	6.22 (6.5%)	5.94 (6.7%)
	8.0	6.0	77.3 82.3[c]	84.0 89.4[c]	74.6 (6.0%)	76.9 (7.7%)
Mixture[d]	8.0	6.7	38.6	54.5	36.2 (2.7%)	43.4 (2.5%)
Ordinary Concrete	8.0	10.8	0.928	0.947 0.922[e]	0.871 (2.9%)	0.869 (3.9%)
Ordinary Concrete	2.0[f]	--	3.02 (7.8%)	3.23(3.0%) 3.16(3.0%)[e]	2.87 (5.7%)	2.88 (6.0%)

[a] One standard deviation statistical uncertainty.
[b] Transport calculation with Version 4 without Bremsstrahlung.
[c] Results with ISOSHLD dose conversion factors rather than ANSI/ANS-6.1.1-1977.
[d] Five equal gram thicknesses: 1st and 2nd regions are iron; 3rd region 12.5% hydrogen and 87.5% lead; 4th region 50% iron and 50% lead; and 5th region 67% iron and 33% oxygen--all given in weight percent.
[e] Results based on buildup factors for concrete rather than calculated z-bar.
[f] Neutron source energy where the dose rate is from the photons produced at the neutron collision. Concrete thickness is 868 g/cm^2.

Optionally Bias Fission Neutrons toward Higher Energies

The uncertainties of high energy reaction rates can be reduced by setting weight windows low at high energies. However, this only increases the frequency of random walks at high energies after a high energy neutron has been produced at a fission event. It does not increase the frequency of producing high energy neutrons from fission events and hence uncertainties may typically be further reduced by biasing the energy selection at the fission event. We have included an option to allow the user to bias fission events above some input energy. The code computes the analog probability of sampling above the energy, and the user inputs a biased probability p for sampling above this energy: p is typically from ten to one hundred times the analog probability. The code then samples above the input energy with probability p and multiples the weight by 1/p, or samples below the energy with probability (1-p) and multiplies the weight by 1/(1-p).

There should be some consistency between the probability p and the ratio of the low energy to high energy weight windows. The code will optionally multiply the analog probability of sampling each fission event above the input energy by the ratio of the lowest energy to highest energy weight window for the cell where the fission event occurs. This sampling probability is not allowed to be greater than an input probability furnished by the user.

The weight windows being used in a problem along with the fission biasing may not lead to the desired number of source neutrons in a subsequent generation. Once a generation has gone by, the code will begin adjusting the proportionality constant, being used at fission events for producing neutrons for the next generation, in order to obtain approximately the desired number of source neutrons. However, when starting with an srctp file, it doesn't have appropriate information to make an adjustment during the first generation. The user can input a k as the second entry on the kcode card. Then the number of neutrons generated for the second generation will be inversely proportional to this k. If the average weight window for the source neutrons is near unity, this k entry may not be required. If it differs substantially from unity, the entry may be imperative and it also may be necessary to skip a few generations depending upon the accuracy of the k entry--it is a good idea to skip one generation even for a well converged srctp file and the appropriate k entry. The user should always check to see that the number of source neutrons begins and continues at about the number desired--to within statistical fluctuations. A pertinent printout is obtained with a "grep sampling outp" command.

The user should also be careful to skip a sufficient number of generations before accumulating tally information by using the third entry on the kcode card. Changes in the weight windows will typically mean that more generations should be skipped even if the srctp file represents a converged source for the problem prior to the weight window changes. The resulting srctp file at the end of the calculation should be saved if similar calculations will be made in the future.

Two other words of caution will be noted for kcode calculations. Monte Carlo calculations tend to have a bias that is inversely proportional to the number of neutrons per generation. This bias is likely increased when using strong biasing with weight windows, and calculations for some systems have indicated that a few thousand neutrons per generation may be required to effectively remove the bias. The other word of caution is that the MCNP estimation of uncertainties is non-conservative since it neglects fission source correlations. Preliminary indications are that the MCNP estimated uncertainties could be low by as much as a factor of two in some kcode problems. Efforts are being made to quantify this.

Special fs Tally Option

Checking of the input data for this option is in subroutine nxtitl. Locators involving the total length of this special tally are set in subroutine itally along with a specification of the order of the printing of the tally in the outp file. Information regarding this special tally is printed in the outp file with subroutine italpr. However, most of the additional coding to implement this option are in subroutine volume. Here the volume elements are computed for any of the three coordinate systems, and the mid-mesh values of the coordinates are determined and written to the file qmesh. Considerable input error checking is performed in subroutine volume, which in most cases result in a fatal error so random walks will not be processed until the input error is corrected. The mesh volumes are optionally printed to the outp file in subroutine italpr.

A subroutine named chkcelv was added to the code that is used only for this special fs tally option. This subroutine determines which mesh interval the particle is in at the beginning of each flight path that is across some, or all, of the tally cells. This subroutine was obtained by making a few modifications to subroutine chkcel to do the same type of thing for the special tally. Subroutine tally was modified to call chkcelv at the beginning of the flight path. Subroutine tally then calls subroutine track, as usual, to obtain the "fs" surfaces crossed during the flight path. Then additional modifications to subroutine tally determine which mesh segments the flight crosses, the distance across each segment, and makes the tally for each segment.

Subroutine tallyp was modified to write the special fs tally information to the outp file, and subroutine tallyq was modified to write out the volumes for the mesh. Finally, subroutine mctalw was modified to write out the tallies for the mesh to the mctal file.

Optional DXTRAN Angle Biasing Treatments

Optional DXTRAN treatments are implemented with the tenth entry on the "idum" card, where the eleventh and twelfth entries on the "idum" card and the eighth entry on the "rdum" card can also be used to change some defaults -- see Figure A.2 for a discussion of these options and Figure A.4 for a description of entries on the dxt card for a cylindrical or rectangular slot. With the tenth entry on the "idum" card equal to zero, the standard mcnp DXTRAN treatment is unchanged.

The optional DXTRAN treatments can be thought of as an angle-bias implementation, where the DXTRAN particles are created within a small solid angle that is described by a sphere, concentric cylindrical gap, or rectangular gap. The DXTRAN particles are not attenuated in this optional treatement, as in the standard DXTRAN, but continue in a random walk from the collision point where they were created. Particles are not deleted in these optional treatments when they reach the DXTRAN vicinity. The probabilities on the DXCm card are also treated a little differently in that if a cell probability is zero, DXTRAN is not operational and the normal particle can also emerge in the "non-active" DXTRAN solid angle (when DXTRAN is used at a collision the normal particles are terminated if they emerge in the DXTRAN solid angle). For more than one DXTRAN sphere, only one probability on the DXCm cards is allowed to be greater than zero for a given cell in order to preserve an unbiased game. This has the advantage that you can angle bias when far away in distance (although usually relatively close in mean free paths) and just do normal sampling for closer distances.

To avoid a proliferation of particles or playing roulette with the DXTRAN particles you actually want, usually a lower weight cutoff is not used; i.e., the third and fourth entries on the CUT card are set to zero. To control the number of DXTRAN particles, by default mcnph with the new options will play roulette (proportional to the weight decrease that occured at the creation of the DXTRAN particle) with a DXTRAN particle if it collides while in the same cell where it was created, or you can use your own probability with the eighth entry on the RDUM card. You can further control the number of DXTRAN particles by increasing or decreasing the probabilities on the DXCm card.

Typically these DXTRAN options will be used for problems where scattering into a very small solid angle is important, such as scattering down a duct or scattering from the surface of waste near the top of a tank into the small solid angle of a riser at the ceiling. The optional choice of "DXTRAN Sphere", "DXTRAN Cylindrical Slot", or "DXTRAN Rectangular Slot" will depend upon the shape of the desired target region.

DXTRAN Sphere Without Attenuation

This is like the standard DXTRAN except the DXTRAN particles are not attenuated, the normal random walk particles are rejected if they scatter (or are born from the source) within the solid angle of the "active" DXTRAN sphere, and neither the random walk particles nor the DXTRAN particles are rejected as they enter a

DXTRAN sphere. Probabilities are required on the DXT card as for the standard MCNP code.

DXTRAN Cylindrical Slot Without Attenuation

This option is like the "DXTRAN Sphere Without Attenuation" except the DXTRAN solid angle is between two cylindrical gaps and between two azimuthal angles defined on some plane surface as shown in the plan view of Figure A.10 -- there is actually an inner and outer gap defined where the density function per unit area of the inner is five times the outer. The DXTRAN angle biasing can be used for a range of distances of collisions below the slot or mcnph will optionally move this plane surface along its axis so that it is always a fixed distance above the collision point.

At each collision subroutine dxtran.f is first called to set up parameters for the collision and to decide if a DXTRAN treatment will be used at the collision. This involves the following steps:

1. The cell probabilities for each DXTRAN cylindrical slot are examined and the probability, p, for the DXTRAN slot with a positive entry is stored in memory. If all probabilities are zero, a variable is set to indicate that DXTRAN will not be used in this collision and an exit is made from subroutine dxtran.f.

2. The collision point is first transformed to global coordinates, if transformations are involved, and all the subsequent mathematics are in the global coordinate system. The distance ß (See elevation view of Fig. A.10) from the collision to the center of the cylinders at the slot elevation is computed. Then the direction vector components (u_{10}, v_{10}, w_{10}) from the collision to the center of the slot are computed. Then the cosine of the angle, μ, between this vector and the positive axial slot vector is computed as:

   (A.6) $\mu = u_o u_{10} + v_o v_{10} + w_o w_{10}$

   If μ is less than the user input minimum (usually the desired angle bias is nearly along the axis vector so wide angles are not of interest), a variable is set to not use DXTRAN for this collision and an exit is made from subroutine dxtran.f. The distances l and α are evaluated from the geometry of Fig. A.10. If l is not within the user specified range, a variable is set to not use DXTRAN for this collision (or optionally the user can specify a fixed distance between the collision and the plane for l and the slot is moved to obtain this).

3. The areas, a_i and a_o, for the inner and outer DXTRAN regions are computed and b is computed as the ratio of the approximate outer density function in direction of flight compared to isotropic

(A.7) $b = (5a_i + a_o)/(4\pi \beta^2)$

If b is greater than 0.25, DXTRAN is not used for this collision since it will be doing very little biasing anyway.

4. The (x_2,y_2,z_2) points for the collision site projected onto the slot plane (see Fig. A.10) are computed by advancing the collision coordinates a distance l along the slot axis direction. Then the (u_{o2},v_{o2},w_{o2}) direction vector from the center of the slot plane to this projection are computed using the two spatial locations. This direction vector is the reference for the azimuth of the slot; i.e., it is the center of the azimuth for this DXTRAN slot and collision.

(A.8) $u_{o2} = (x_1 + u_o l - x_o)/\alpha$

$v_{o2} = (y_1 + v_o l - y_o)/\alpha$

$w_{o2} = (z_1 + w_o l - z_o)/\alpha$

5. If the cell probability p is between zero and one, roulette is used to detemine whether a DXTRAN particle will be generated. If it is to be generated, the weight is increased by 1/p. In either case, the normal particle will be terminated if it emerges within the DXTRAN solid angle.

During the collision, subroutine dxtran.f is again called to potentially create a DXTRAN particle and put it in the bank based on the information from steps 1 to 5 above. If a DXTRAN particle is not to be created, the following steps are omitted.

6. Radial and azimuthal values are randomly sampled proportional to the area on the slot plane using the relevant input DXTRAN parameters. The inner covering area is sampled with probability $5a_i/(5a_i + a_o)$, where the density function of the inner is assumed to be five times that of the outer. For the inner, the radius is sampled proportional to r^2 between the two radial bounds of the inner and the azimuth angle is sampled uniformly between the negative of its maximum specified input value and the positive. The sampling is more complicated for the outer, but the same ideas are involved.

7. Determine the (u',v',w') direction from the slot axis center to the sampled position. This involes using Fig. 4.1 of Carter and Cashwell with $\mu_{lab}=0$ along with double angle trigonometry formulas for expressing the azimuth as the sum of the base azimuth angle determined by the collision and the sampled azimuth angle with respect to the base. The result is

(A.9) $u' = u_{o2}c + s(v_o w_{o2} - w_o v_{o2})$

$$v' = v_{02}c + s(-u_o w_{o2} + w_o u_{o2})$$

$$w' = w_{02}c + s(u_o v_{o2} + v_o u_{o2})$$

where s and c are the sine and cosine values of the sampled azimuth. If abs(w_o) is >0.999999, limit equations are used -- see bottom of Fig. 4.1 of Carter and Cashwell. The above result was obtained using the following for the cosine and sine of the base azimuth angle for the collision -- repeating the use of Fig. 4.1 for the base direction on the plane (u_{o2}, v_{o2}, w_{o2}):

$$\cos = -w_{o2}/q$$

$$\sin = (u_o v_{o2} - v_o u_{o2})/q$$

$$q = [1-w_o^2]^{0.5}$$

Then the (x',y',z') value of the sampled location on the plane is obtained by simply advancing a distance equal to the sampled radius r from the reference axis center in the (u',v',w') direction

$$\text{(A.10)} \quad x' = x_o + ru'$$

$$y' = y_o + rv'$$

$$z' = z_o + rw'$$

8. The direction of flight is evaluated as the direction from the collision to the (x',y',z') of Eq. (A.6) and this distance, g, is also determined along with the cosine, Δ, of the angle between the new direction-of-flight (u",v",w") and the (u_o, v_o, w_o) of the positive normal to the slot. The new direction-of-flight is transformed to the local coordinate system if there is one.

9. Standard calls in subroutine dxtran.f are used to evaluate the probability density function, f, for the particle scatting through the appropriate angle to have the desired final direction of flight from step 8. Then the weight is multiplied by the factor

$$f(5a_i + a_o)/(2p\pi g^2)$$

if the sampling was within the outer DXTRAN or 1/5th of this weight if in the sampling was within the inner DXTRAN. Note that for isotropic scattering (f=0.5) and if we had used a factor of 1 instead of 5 for the inner DXTRAN and if p=1, the above factor reduces to

$$(\text{Total DXTRAN Area of slot})/(4\pi g^2)$$

This is the fraction of the 4π solid angle subtended by the DXTRAN area if the area is always normal to the direction-of-flight. This expected result is gratifying. The issue that Δ will not always be normal (=1) has been addressed by 1986 notes that J.E. Booth from LANL kindly supplied to me. The correct expected value is obtained if the weight is also multiplied by Δ -- we allow either the multiplication or roulette on whether to bank the DXTRAN particle. If the DXTRAN particle is created (banked), the weight reduction is remembered and this DXTRAN particle is (optionally) terminated with twice that probability if the next collision is in the same cell where it was created. This is done to reduce the prolifferation of DXTRAN particles -- if it has a collision in the same cell, it typically is not reaching the region of interest.

Finally, subroutine dxtran.f is called for the normal random walk particle to see if its direction-of-flight is within the solid angle of the DXTRAN particle (this check is not made if DXTRAN was not used at the collision -- see steps 1, 2, and 3 above). This is accomplished with the following steps:

10. Transform the normal particle to the global coordinate system and determine where this normal particle will cross the DXTRAN plane.

11. Determine the radius on this plane. If the radius is outside of the radial bounds of the DXTRAN cylindrical slot, return and accept this normal particle. If it is inside the radial bounds, go to step 12 to examine the azimuth.

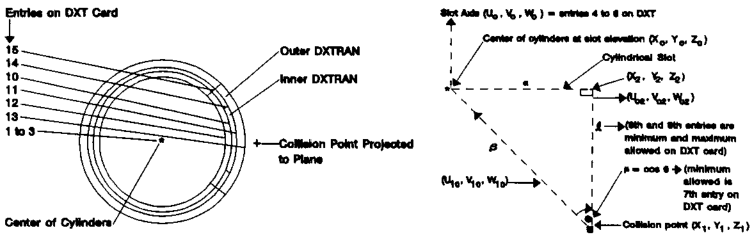
12. Compute the direction of flight (u,v,w) from the center reference point to this crossing and compute the relative azimuth, δ, as (see Eq. (A.8) for the reference azimuthal direction):

    $$\delta = \cos^{-1}[uu_{02}+vv_{02}+ww_{02}]$$

    If the azimuth angle δ is within that specified for the outer DXTRAN, this normal particle history is terminated. Otherwise it continues.

Tallies are made for both the normal and DXTRAN particles. There is no such thing as being "inside" the DXTRAN sphere since this is an area on a plane. The tallies from the normal particles represent all contributions from scattering outside of the DXTRAN solid angle (or for all angles for those collisions where DXTRAN was not used). The tallies from the DXTRAN particles are made after each random walk has had at least one of the scattering events produce a DXTRAN particle. How much of the tally came from the two different contributions will generally not be known to the user. If the DXTRAN parameters were defined wisely, an increased figure of merit should result.

Figure A.10. Plan and Elevation Views of DXTRAN Cylidrical Slot.


Entries on DXT Card
15
14
10
11
12
13
1 to 3
Outer DXTRAN
Inner DXTRAN
Collision Point Projected to Plane
Center of Cylinders
Slot Axis (U_o, V_o, W_o) = entries 4 to 6 on DXT
Center of cylinders at slot elevation (X_o, Y_o, Z_o)
Cylindrical Slot
(X_2, Y_2, Z_2)
α
(U_{o2}, V_{o2}, W_{o2})
β
ℓ (8th and 9th entries are minimum and maximum allowed on DXT card)
$\mu = \cos\theta$ (minimum allowed is 7th entry on DXT card)
(U_{1o}, V_{1o}, W_{1o})
θ
Collision point (X_1, Y_1, Z_1)

DXTRAN Rectangular Slot Without Attenuation

This option is essentially identical to the "DXTRAN Cylindrical Slot Without Attenuation" except the DXTRAN solid angles are defined by rectangles as shown in Figure A.11 -- again there is an inner and outer area defined where the density function per unit area of the inner is five times the outer. The DXTRAN angle biasing can be used for a range of distances of collisions below the slot or mcnph will optionally move this plane surface along its axis so that it is always a fixed distance above the collision point.

At each collision subroutine dxtran.f is first called to set up parameters for the collision and to decide if a DXTRAN treatment will be used at the collision. This involves the steps 1, 2, 3, and 5 exactly like was done for the DXTRAN Cylindrical Slot except the areas in step 3 now involve rectangles instead of azimuthal portions of cylindrical gaps. Things are somewhat simpler for the rectangle so step 4 is not needed.

During the collision, subroutine dxtran.f is again called to potentially create a DXTRAN particle and put it in the bank based on the information from steps 1, 2, 3, and 5. If a DXTRAN particle is not to be created, the following steps are omitted.

6. The inner covering area is sampled with probability $5a_i/(5a_i + a_o)$, where the density function of the inner is assumed to be five times that of the outer. For the inner, the "thin" distance from the center of the rectangular slot is sampled uniformly between zero and the maximum entry specified in the input. A random number is compared to 0.5 to switch the sign of this distance half of the time. This is then repeated in the thick direction for the inner. The sampling is more complicated for the outer, but the same ideas are involved.

7. The (x',y',z') value of the sampled location on the DXTRAN plane is obtained by simply advancing vectorially the three coordinates of the particle from the center (x_o,y_o,z_o) location the appropriate sampled distances from step 7 in the "thin" and "thick" directions.

8. The direction of flight is evaluated as the direction from the collision to the (x',y',z') and this distance, g, is also determined along with the cosine, Δ, of the angle between the new direction-of-flight (u",v",w") and the (u_o,v_o,w_o) of the positive normal to the slot. The new direction-of-flight is transformed to the local coordinate system if there is one.

9. Standard calls in subroutine dxtran.f are used to evaluate the probability density function, f, for the particle scatting through the appropriate angle to have the desired final direction of flight from step 8. Then the weight is multiplied by the factor

$$f(5a_i + a_o)/(2p\pi g^2)$$

if the sampling was within the outer DXTRAN or 1/5th of this weight if in the sampling was within the inner DXTRAN. Note that for isotropic scattering (f=0.5) and if we had used a factor of 1 instead of 5 for the inner DXTRAN and if p=1, the above factor reduces to

(Total DXTRAN Area of slot)/($4\pi g^2$)

This is the fraction of the 4π solid angle subtended by the DXTRAN area if the area is always normal to the direction-of-flight. This expected result is gratifying. The issue that Δ will not always be normal (=1) has been addressed by 1986 notes that J.E. Booth from LANL kindly supplied to me. The correct expected value is obtained if the weight is also multiplied by Δ -- we allow either the multiplication or roulette on whether to bank the DXTRAN particle. If the DXTRAN particle is created (banked), the weight reduction is remembered and this DXTRAN particle is (optionally) terminated with twice that probability if the next collision is in the same cell where it was created. This is done to reduce the prolifferation of DXTRAN particles -- if it has a collision in the same cell, it typically is not reaching the region of interest.

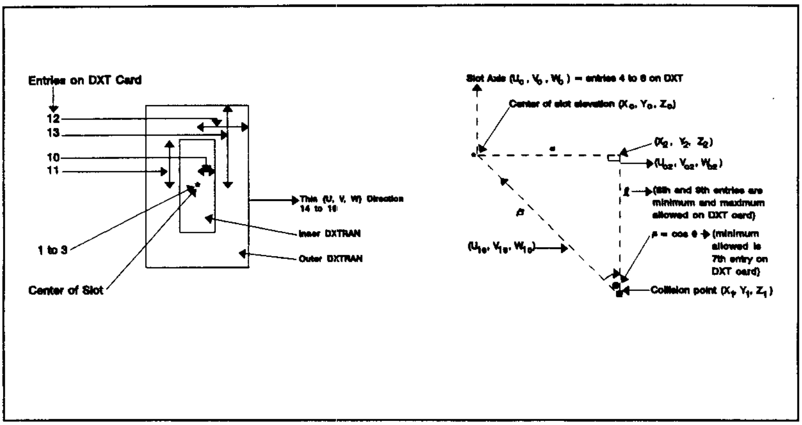
Finally, subroutine dxtran.f is called for the normal random walk particle to see if its direction-of-flight is within the solid angle of the DXTRAN particle (this check is not made if DXTRAN was not used at the collision -- see steps 1, 2, and 3 above). This is accomplished with the following steps:

10. Transform the normal particle to the global coordinate system and determine where this normal particle will cross the DXTRAN plane.

11. Determine how far this crossing of the plane is from the slot center in the "thin" and "thick" directions. If the absolute value of either of these distances exceeds the input slot dimensions, return and accept this normal particle. Otherwise, this normal particle is terminated because it is within the DXTRAN angle and is being treated by the DXTRAN particle.

Tallies are made for both the normal and DXTRAN particles. There is no such thing as being "inside" the DXTRAN sphere since this is an area on a plane. The tallies from the normal particles represent all contributions from scattering outside of the DXTRAN solid angle (or for all angles for those collisions where DXTRAN was not used). The tallies from the DXTRAN particles are made after each random walk has had at least one of the scattering events produce a DXTRAN particle. How much of the tally came from the two different contributions will generally not be known to the user. If the DXTRAN parameters were defined wisely, an increased figure of merit should result.

Figure A.11. Plan and Elevation Views of DXTRAN Rectangular Slot.


Entries on DXT Card
12
13
10
11
1 to 3
Center of Slot
This (U, V, W) Direction 14 to 16
Inner DXTRAN
Outer DXTRAN
Slot Axis (U_0, V_0, W_0) = entries 4 to 6 on DXT
Center of slot elevation (X_0, Y_0, Z_0)
(X_2, Y_2, Z_2)
(U_{02}, V_{02}, W_{02})
ℓ (8th and 9th entries are minimum and maximum allowed on DXT card)
ρ
(U_{10}, V_{10}, W_{10})
μ = cos θ (minimum allowed is 7th entry on DXT card)
Collision point (X_1, Y_1, Z_1)

VI. REFERENCES

1. Briesmeister, J. F., Editor, MCNP - A General Monte Carlo N-Particle Transport Code, Version 4A, LA-12625, Los Alamos National Laboratory, Los Alamos, New Mexico, 1993. The Radiation Shielding Information Center will send users this manual packaged under the RSIC COMPUTER CODE COLLECTION with RSIC number CCC-200, Radiation Shielding Information Center, Oak Ridge National Laboratory, 1993.

2. George, D. C., R. J. LaBauve, and T. R. England, June 1982, Application of Adjusted Data in Calculating Fission-Product Decay Energies and Spectra, LA-9362-MS, Los Alamos National Laboratory, Los Alamos, NM.

3. D.K. Truby and Y. Harima, "RSIC DATA LIBRARY COLLECTION ANS643 [DLC-129], ANS-6.4.3 Geometric Progression Gamma-Ray Buildup Factor Coefficients," Radiation Shielding Information Center, Oak Ridge National Laboratory, July 1990.

APPENDIX B

LISTING OF THE PRPR PATCH FILE p.4a00c FOR MAKING MCNP COMPATIBLE WITH WHC COMPUTER PLATFORMS

APPENDIX B. Listing of the PRPR Patch File, p.4a00c, for making MCNP Compatible with WHC Computer Platforms

```
*define unix,cheap,sun,pointer,plot,mcplot,gkssim,xlib
*ident whc
*/ common changes for max entries per cell
*d zc4a.7
     1 mink=200,mipt=3,mjsf=9,mkft=9,mktc=22,mlgc=310,mpb=5,mpng=21,
*d cor4-2.24
     3 hdpath/'/apps/mcnp'/,
*i mc.202
      write(iuo,162)
  162 format(49h edt 465711, sd-mp-swd-30001, rev 0, certified    )
      write(iuo,163)
  163 format(" ecn 186718 with prpr upgrade p.4a00c"        )
*i mc.204
*if def,cheap
*if def,hpux,1
      write(iuo,'(a20))')'HP platform         '
*if def,sgix,1
      write(iuo,'(a20))')' SGI platform       '
*if def,aix,1
      write(iuo,'(a20))')' IBM platform       '
*if -def,hpux.and.sgix.and.aix,1
      write(iuo,'(a20))')'Sun platform        '
*endif
*i im4a.6
      if(mnk.eq.0)ink(160)=0
*i ch.107
      if(m2-m1.ge.100)call erprnt(1,2,3,ncl(lncl+ic),m2-m1+1,mlgc,0,0,
     1 '19hdescription of cell,i5,5h uses,i4,9h words.  ,i5,8h is max.')
*i pl.148
*if def,xlib,2
c     open post-script file
      call openfile(1)
*i pl.154
*if def,xlib,2
c     close post-script file out.ps
      call closefile()
*i cor4-2.174
*if def,sgix,4
c     adjust sgi to read random access cross sections
   93 format(8h ly(4)=     ,i10)
      if(ly(1).eq.2)ly(4)=ly(4)/4
c     write(iuo,93) ly(4)
*i tn.73
c     quit if file qmcnpnow exists
      if(inqire('qmcnpnow').ne.0)nst=nst+32
*d vp4a.126
```

```
*/ fix of LANL history track write error
      write(iupw)(iw(j),j=1,1),(7,8,(j,j=id(1,n),id(2,n)),
*i mz.106
       call openfile(2)
*i mz.126
       if(koplot.le.1)call closefile()
*d gi4a.3
*if def,sgi2,3
      call idate(jd(1),jd(2),jd(3))
      call itime(jt)
      write(hi,10)jd(2),jd(1),mod(jd(3),100),jt
*if -def,aix.and.hpux.and.dec.and.sgi2,3
```

APPENDIX C

UNIQUE CONSIDERATIONS FOR HP, SGI, AND IBM PLATFORMS.

APPENDIX C. UNIQUE CONSIDERATIONS FOR HP, SGI, and IBM PLATFORMS.

The discussions in the main body of this ECN and in Appendix A have been, for the most part, applicable to all the computers. Where there were differences, the discussion and examples were oriented to the Sun (4/260) Workstations. In this Appendix we discuss unique considerations for the HP, SGI and IBM platforms, including relevant command files. Most of the differences are already accounted for in the LANL method of generating the source code with prpr so this discussion will be brief.

The command files that were used to create the executable mcnp file on the HP, SGI, and IBM platforms are shown in Figures C.1, C.2 and C.3, respectively.

Command files identical to those of Figures C.1, C.2, and C.3 are used to generate the mcnph executables except:

- The patch file p.4a00c is replaced by ph.4a00c;

- The executable file name, mcnp, is replaced by mcnph on the load line.

Two-processor HP's and sgi's are becoming available (such as the smpl Challenge L Series SGI with 4400 boards). The standard executables compiled on the older sgi's works on smpl and all the test problems compared successfully so these are satisfactory executables on the two-processor machines. In the future it may be possible to obtain faster execution speeds by compiling and loading on the two-processor machines. If a decision is made to do this, the code custodian will be responsible for demonstrating that the resulting executable compares successfully for the test problems prior to release to users for QA applications. The next ECN for MCNP will then show the job stream for compiling and loading, and will give any other pertinent information.

Figure C.1. The Command File, makemcnp.hp, to Create the Executable File, mcnp on the HP Platform.

```
#! /bin/sh
# shell script to make MCNP-4a on HP-730 Platform
# local files (required):  mcnpc.id mcnp4a.id p.4a00c prpr.id
#                          pc4a
mkdir flib
mkdir olib
rm flib/*
rm olib/*
set -x
rm -f codef patch compile newid clog prpr mcnp *.f *.o
# compile and load prpr
cp prpr.id prpr.f
/usr/bin/f77 prpr.f -o prpr
rm -f codef newid *.f *.o
# merge main C code with C patch file
sed s/,sun,/,hpux,/ pc4a >patch
cp mcnpc.id codef
prpr
mv compile mcnpc.c
mv newid newc.id
# compile C code
/bin/cc -I/t/'hostname'/llc4a -c mcnpc.c
rm -f codef patch
# merge main FORTRAN code with patch file
cp mcnp4a.id codef
sed s/,sun,/,hpux,/ p.4a00c >patch
prpr
mv newid newf.id
# split FORTRAN source file
/usr/bin/fsplit compile > clog
rm -f compile codef clog
# put FORTRAN source in flib/ and C source in flib/
mv *.c flib
# compile FORTRAN
/usr/bin/f77 +T +E1 -O -c a*.f b*.f c*.f d*.f e*.f f*.f
# hp-730 bug fix:  remove optimization for addtfc.f and findel.f
rm -f addtfc.o dxtran.o findel.o
/usr/bin/f77 +T +E1 -c addtfc.f dxtran.f findel.f
/usr/bin/f77 +T +E1 -O -c g*.f h*.f i*.f j*.f k*.f l*.f m*.f
/usr/bin/f77 +T +E1 -O -c n*.f o*.f p*.f q*.f r*.f s*.f
/usr/bin/f77 +T +E1 -O -c t*.f u*.f v*.f w*.f x*.f y*.f z*.f
mv *.f flib
# The following line loads MCNP with XLIB graphics in general:
/usr/bin/fort77 -o mcnp *.o -L/usr/lib/X11R4 -lX11
# move .o files to olib/
mv *.o olib
```

Figure C.2. The Command File, makemcnp.sgi, to Create the Executable File, mcnp on the SGI Platform.

```
#! /bin/sh
# shell script to make MCNP-4a on SGI  Platform
# local files (required):  mcnpc.id mcnp4a.id p.4a00c prpr.id
#                          pc4a
mkdir flib
mkdir olib
rm flib/*
rm olib/*
set -x
# compile and load prpr
cp prpr.id prpr.f
/usr/bin/f77 prpr.f -o prpr
rm -f prpr.f newid compile
# merge main C code with C patch file
/usr/bin/sed s/,sun,/,sun,sgix,/ pc4a >patch
cp mcnpc.id codef
prpr
mv compile mcnpc.c
mv newid newc.id
# compile C code
/usr/bin/cc -O2 -c mcnpc.c
rm -f codef patch
# merge main FORTRAN code with patch file
cp mcnp4a.id codef
/usr/bin/sed s/,sun,/,sun,sgix,/ p.4a00c >patch
prpr
mv newid newf.id
# split FORTRAN source file
/usr/bin/fsplit compile > clog
rm -f compile codef clog
# put FORTRAN source in flib/ and C source in flib/
mv *.c flib
# compile FORTRAN
/usr/bin/f77 -static -O2 -Nn6000 -c a*.f b*.f c*.f d*.f e*.f f*.f
/usr/bin/f77 -static -O2 -Nn6000 -c g*.f h*.f i*.f j*.f k*.f l*.f m*.f
/usr/bin/f77 -static -O2 -Nn6000 -c n*.f o*.f p*.f q*.f r*.f s*.f
/usr/bin/f77 -static -O2 -Nn6000 -c t*.f u*.f v*.f w*.f x*.f y*.f z*.f
mv *.f flib
# recompile two routines with default optimization
/usr/bin/f77 -c flib/sing.f
rm findel.o
/usr/bin/f77 -static -c flib/findel.f
# load mcnp with both C and FORTRAN .o files
/usr/bin/f77 -static -o mcnp *.o -lX11
# move .o files to olib/
mv *.o olib
```

Figure C.3 **The Command File, makemcnp.ibm, to Create the Executable File, mcnp on the IBM Platform.**

```
#! /bin/sh
# shell script to make MCNP-4a on IBM RSIC 6000 Platform
# local files (required):  mcnpc.id mcnp4a.id p.4a00c prprr.id
#                          pc4a
mkdir flib
mkdir olib
rm flib/*
rm olib/*
set -x
# compile and load prpr
rm -f codef patch compile newid clog prpr makxsf mcnp *.f *.o
cp prpr.id prpr.f
/bin/xlf prpr.f -o prpr
rm -f patch codef newid *.f *.o
# Merge main C code with C patch file
/bin/sed s/,sun,/,aix,/ pc4a >patch
cp mcnpc.id codef
./prpr
mv compile mcnpc.c
mv newid newc.id
/bin/cc -O -c mcnpc.c
rm -f codef patch
# merge main FORTRAN code with patch file
cp mcnp4a.id codef
/bin/sed s/,sun,/,aix,/ p.4a00c >patch
./prpr
mv newid newf.id
rm -f codef patch
# split FORTRAN source file
/bin/fsplit compile > clog
rm -f compile codef clog
# put FORTRAN source in flib/ and C source in clib/
mv *.c flib
/bin/xlf -O -NQ20000 -NA16384 -c *.f
# The following line removes optimization for some compiler bugs.
/bin/xlf -NQ20000 -NA16384 -c brang.f tallyp.f nextit.f
mv *.f flib
# load mcnp with both C and FORTRAN .o files
/bin/xlf -o mcnp *.o -lX11
# move .o files to olib/
mv *.o olib
```

APPENDIX D

UNIQUE CONSIDERATIONS FOR DOS PLATFORMS.

APPENDIX D. UNIQUE CONSIDERATIONS FOR DOS PLATFORMS.

Obtaining the mcnp executable on a DOS machine was in many ways simpler than on the UNIX machines. The patch file, patchs.dos, for the mcnp executable only consisted of the following cards:

```
   *define pcdos,cheap,plot,mcplot,gkssim,lahey
   *d zc4a.5
         parameter (mdas=4000000)
   *d cor4-2.24
        3 hdpath/'c:\mcnp4a\xs'/,
   *i mc.202
         write(iuo,162)
     162 format(49h edt 465711, sd-mp-swd-30001, rev 0, certified     )
         write(iuo,163)
     163 format(49h ecn 186718 with prpr upgrade patchs.dos           )
```

where the *define card defines the computer hardware and software system including the lahey compiler and graphics. The second and third cards set the length of the main data array to 4,000,000 words -- this is the code default, but is included here since this parameter would be changed to load an executable for smaller or bigger machines. The DOS platform does not currently have the dynamic memory capability that is on the UNIX platforms. The forth and fifth cards modify the default location for the cross sections to the directory \mcnp4a\xs\. The code will look for the xsdir file in this directory if it is not in the local directory of execution. Finally, the last four cards write out QA identification information to the output file.

The FORTRAN source code was then generated with the LANL specified commands:

```
   copy mcnpf.id codef
   copy patchs.dos patch
   prpr
   copy compile mcnp.for
```

The executable file, mcnp.exe, was then generated with the LANL commands:

```
f7713 mcnp.for
386link mcnp.obj -stub runb -exe mcnp.exe -Lib graph3
```

For this latter step it is important to have the file f7713.eer, from LANL, in the directory and the LACEY FORTRAN compiler must be available.

The cross section files (including xsdir) were copied into the directory \mcnp4a\xs\, and the executable mcnp.exe file was copied into the directory \mcnp4a\exec\ to enable execution of MCNP problems in this directory.

The 25 test problems were executed to verify the executable. This execution of the test problems was in the directory \mcnp4a\exec\test\, where the files needed in this directory included: the mcnp.exe execution file; the xsdir directory file; a file tinp_dos.exe to retrieve the 25 input files; a file tmct_dos.exe containing the mctal file output from the 25 corresponding calculations at LANL; a file tout_dos.exe containing the outp output from the 25 corresponding calculations at LANL, and a script file runprob.bat that will execute the 25 test problems.

The tinp_dos.exe, tmct_dos.exe, and tout_dos.exe files were all executed to strip out the 25 files from each. Then the runprob.bat file was executed to execute the 25 test problems. This script file compares the differences between the output files from our calculations to those made at LANL to verify the executable. There were no differences in the comparisons to the LANL mctal files.

This verification of the MCNP executable was made on an "AST PREMMIA 4/33" machine. Since this executable has been tested with the test problems, it can be used on other similar machines. Or the above compilation with testing procedure can be exercised on other machines to verify a new executable, where the mdas= entry on the 3rd card of the patch file may be changed to some other number than 4000000 to increase or decrease the maximum allowed problem size, and other minor changes may be made to adjust to a new platform. This compilation and testing of a new executable must be done by the code custodian or the alternate code custodian. This compilation and testing procedure will also be followed for mcnph by the code custodian or the alternate code custodian, where all the test problems must be utilized.

The geometry plotting works on the DOS platforms, but there is presently no hardcopy capability.

APPENDIX E

THE SABRINA PRE- AND POST- PROCESSOR.

APPENDIX E. THE SABRINA PRE- AND POST- PROCESSOR.

SABRINA[1] is a three-dimensional interactive geometry modeling program primarily used in conjunction with MCNP. The user should refer to the manual[1] for a complete description of input required and output examples. SABRINA can be executed on UNIX platforms at WHC with the command

```
  /apps/mcnp/sabrina
```

This command file will determine the platform from which you are executing and will then utilize the appropriate executable for SABRINA. If an executable does not exist on your platform, it will write out a message so stating. Presently the executable on the IBM UNIX platform is recommended for running SABRINA.

REFERENCES

1. Riper, K. A., SABRINA User's Guide for SABRINA Version 3.54, LA-UR-93-3696, Los Alamos National Laboratory, Los Alamos, New Mexico, 1993. The Radiation Shielding Information Center will send users this manual packaged under the RSIC COMPUTER CODE COLLECTION with RSIC number PSR-242, Radiation Shielding Information Center, Oak Ridge National Laboratory, 1993.

DISTRIBUTION SHEET

To Distribution	From Nuclear Physics and Shielding	Page 1 of 1
		Date March 1996
Project Title/Work Order Certification of MCNP Version 4A for WHC Computer Platforms		EDT No.
		ECN No. 186718

Name	MSIN	Text With All Attach.	Text Only	Attach./ Appendix Only	EDT/ECN Only
G. Akabani	K3-55	X			
L. L. Carter (10)	H0-35	X			
T. Chaio	A3-34	X			
D. G. Erickson	H0-38	X			
S. H. Finfrock	H0-38	X			
S. R. Gedeon	H0-35	X			
H. J. Goldberg	H0-35	X			
J. Greenborg/File (2)	H0-35	X			
M. D. Harris	H0-38	X			
D. C. Hetzer	S6-31	X			
B. E. Hey	A3-34	X			
K. E. Hillesland	H0-35	X			
S. F. Kessler	H0-35	X			
J. S. Lan	H0-35	X			
T. L. Miles	K8-34	X			
E. M. Miller	A3-34	X			
R. J. Morford	E6-35	X			
J. V. Nelson	H0-35	X			
V. E. Roetman	H0-35	X			
C. A. Rogers	A3-34	X			
F. A. Schmittroth	H0-35	X			
R. A. Schwarz	H0-35	X			
K. N. Schwinkendorf	H0-38	X			
E. R. Selle	S3-90	X			
R. L. Simons	H0-35	X			
T. M. Watson	H0-38	X			
W. D. Wittekind	H0-38	X			
D. W. Wootan	H0-38	X			
Central Files (Original + 3)	A3-88	X			
Tech. Ref. Center	N2-12	X			