LA-UR-16-20109
Approved for public release; distribution is unlimited.

Title: MCNP Surface Source Write/Read File Format Primer

Author(s): Trahan, Travis John

Intended for: Report

Issued: 2016-01-11

MCNP Surface Source Write/Read File Format Primer

LA-UR-16-XXXXX

Travis Trahan

Los Alamos National Laboratory: P.O. Box 1663, Los Alamos, NM, USA, 87544, tjtrahan@lanl.gov

ABSTRACT

This paper provides a partial documentation of the MCNP surface source write and read files, also known in MCNP parlance as wssa and rssa files. No formal documentation of these files is known to exist. This is not a complete documentation of the format, but rather documentation of the minimal set of information needed to generate a usable surface source file. Furthermore, the documentation only applies to the latest MCNP format, "SF_00001".

1. INTRODUCTION

A surface source write/read (SSW/SSR) capability is implemented in the MCNP6 code developed at Los Alamos National Laboratory [1,2]. The SSW capability allows users to record the state of any particle as it crosses specified surfaces. The MCNP capability can also be used to record fission source neutrons in specified cells during KCODE calculations. MCNP can then, in a separate simulation, read in the recorded particles and use them as sources.

Using the SSW/SSR capability, one can perform linked calculations. For instance, if one wants to model a single source but many detectors, the source can be simulated one time, and all particles that reach some surface between the source and detector can then be recorded using a surface source write. Users can then perform separate calculations for each detector using the particles on the surface source files using a surface source read without having to simulate the full problem domain each time.

No formal documentation of the MCNP wssa/rssa format exists. Because the files are written in binary format, the only way to learn their structure is to study the MCNP source code. The MCNP format is documented here only to the minimum extent necessary to generate surface source files that are readable by MCNP. *Thus, the documentation provided here does not fully describe the MCNP wssa/rssa format*. Some missing features that are not documented are the ability to shorten the output format for spherically symmetric problems, and the ability to record fission source neutrons during k-eigenvalue calculations. This document does, however, provide sufficient information to generate a surface source file with any code that can be read by MCNP, provided the aforementioned features are not desired.

2. MCNP SURFACE SOURCE FILE (WSSA/RSSA) FORMAT

A partial definition of the MCNP surface source file (wssa/rssa) format as MCNP version 6.1.1Beta is presented in this section. Recall that this documentation is incomplete, and is missing at least two significant variations of the format: 1) a shortened format that takes advantage of spherical symmetry and

2) a format for storing fission source particles in cells rather than particles from a fixed source problem on surfaces. Finally, *only the latest wssa/rssa format, "SF_00001", is documented here*. The older formats can only be obtained by reading through the MCNP source code. Most of the relevant code for reading and writing of surface source files is found in `source_files.f90`.

2.1 Summary

The overall file format is as follows:

```
Format Record:        id
First Record:         kods, vers, lods, idtms, probs, aids, knods
Second Record:        np1, nrss, nrcd, njsw, niss
Third Record:         niwr, mipts, kjaq      ! if np1 < 0, but it always is
Fourth Record:        do j=1,njss
                          if( kq==0 ) then
                              jss(j),kst(jx),n,(scf(lsc(jx)+i),i=1,n)
Summary Record:       zero, ((nsl(i,j),i=1,2+4*mipt),j=1,njss+nilw)
Particle Record:      a, b, pbl%r%wgt, pbl%r%erg, pbl%r%tme, pbl%r%x,
                      pbl%r%y, pbl%r%z, pbl%r%u, pbl%r%v, c
                            !!== One record for every particle
```

2.2 Variable Definitions

In the following documentation, most variable names are taken from MCNP source code. Note, however, that the same variable may have different names in different subroutines. Each variable is presented in the form of a Fortran declaration similar to its declaration in MCNP. Comments beginning with "`!`" or "`!=`" are taken directly from the MCNP source code, while comments beginning with "`!!==`" provide additional interpretation of the author and, for non-local variables, specify the module to which the variable belongs.

`2.2.a Format Record:`
`id`

```
    character(len=8) :: id
      !!== This is actually treated like an enumerator, and converted to the
      !!== integer values below
      !!== Local variable

    integer, parameter, public :: OLD_MCNPX_RSSA  =  -2
      !  MCNPX surface source.
      !!== From SOURCE_FILES Module
    integer, parameter, public :: OLD_MCNP_RSSA   =  -1
      !  old MCNP surface source.
      !!== From SOURCE_FILES Module
    integer, parameter, public :: NO_RSSA         =   0
      !  No RSSA file to be read (currently unused).
      !!== From SOURCE_FILES Module
    integer, parameter, public :: SF_00001        =   1
      !  New RSSA with format identifier, different second SSB item.
```

```
      !!== THIS IS THE ONLY FORMAT TYPE DOCUMENT HERE
      !!== From SOURCE_FILES Module
```

2.2.b First Record:

kods, vers, lods, idtms, probs, aids, knods

```
    character(len=8) :: kods = ' '
      != Name of the code that wrote surface source file.
      !!== From MCNP_DATA Module
    character(len=5) :: vers = ' '
      != Version of code that wrote surface source file.
      !!== From MCNP_DATA Module
    character(len=8) :: lods = ' '
      != LODDAT of code that wrote surface source file.
      !!== this is a date, defined in mcnp_env.pl
      !!== From MCNP_DATA Module
    character(len=19) :: idtms = ' '
      != IDTM of the surface source write run. (current date and time)
      !!== MM/DD/YY hh:mm:ss
      !!== From MCNP_DATA Module
    character(len=19) ::  probs = ' '
      != PROBID(problem identification string) of the surface source write
      != run.
      !!== From MCNP_DATA Module
    character(len=80) ::  aids = ' '
      != Title card of the surface source write run.
      !!== From MCNP_DATA Module
    integer :: knods
      != Last dump in the surface source write run.
      !!== From FIXCOM Module
```

2.2.c Second Record:

np1, nrss, nrcd, njsw, niss

```
    integer(i8knd) :: np1
      != Number of histories in surface source write run.
      !!== NOTE: THIS SHOULD BE NEGATIVE
      !!==       IT WILL BE CONVERTED TO A POSITIVE NUMBER AFTER READING
      !!== From FIXCOM Module
    integer(i8knd) :: nrss
      != Number of tracks on input surface source file.
      !!== From FIXCOM Module
    integer :: nrcd
      != Number of items to be read from each surface-source event record
      != in the current RSSA (actual items-per-record may be larger).
```

```
      !!== NOTE: THIS SHOULD BE NEGATIVE
      !!==       nrcd = -11 for particle records not assuming spherical
      !!==       symmetry
      !!== From FIXCOM Module
    integer, public :: njsw  = 0
      != Number of surfaces in JASW. (jasw(:),
      != Surfaces from surface source input file.)
      !!== From MCNP_INPUT Module
    integer(i8knd) :: niss
      != Number of histories in input surface source.
      !!== From FIXCOM Module
```

2.2.d Third Record (NOTE: np1 SHOULD be negative):

```
if(np1<0)
   niwr, mipts, kjaq

    integer, public :: niwr  = 0
      != Number of cells in RSSA file.
      !!== From MCNP_INPUT Module
    integer, public :: mipts = 0
      != Source particle type.
      !!== From MCNP_INPUT Module
    integer :: kjaq
      != Flag for macrobody facets on source tape.
      !!== From FIXCOM Module
```

2.2.e Fourth Record:

```
do j=1,njss
    if( kq==0 ) then
        jss(j),kst(jx),n,(scf(lsc(jx)+i),i=1,n)

NOTE: kq is 0 when there are no macrobodies (i.e., BOX)
NOTE: the other parts of this record are for cell sources and are not
presented here

    integer, public :: njss
      != Number of surfaces in JSS
      !!== From FIXCOM Module
    integer, public, ALLOCATABLE :: jss(:)
      != Surfaces for surface source output file.
      !!== Surface indexes from MCNP input file
      !!== From MCNP_GLOBAL Module
    integer :: jx
      !!== Index of surface in lsc and kst arrays
      !!== Not necessarily the same as the index given on the MCNP input file
```

```
  !!== Local variable
integer, public, ALLOCATABLE :: kst(:)
  != Surface-type numbers of all the surfaces.
  !!== From MCNP_GLOBAL Module
integer :: n
  !!== Number of coefficients needed to define the surface type
  !!== Local variable
integer, public, ALLOCATABLE :: lsc(:)
  != For each surface, a pointer into SCF.
  !!== From MCNP_GLOBAL Module
real(dknd), public, ALLOCATABLE :: scf(:)
  != Surface coefficients for all surfaces.
  !!== The coefficients defining the surface
  !!== For example, a general sphere is defined by a radius and three
  !!== coordinates specifying the origin
  !!== From MCNP_GLOBAL Module
```

2.2.f Summary Record:

```
zero,((nsl(i,j),i=1,2+4*mipts),j=1,njss+nilw)

integer, public :: njss
  != Number of surfaces in JSS
  !!== From FIXCOM Module
integer, public :: nilw
  != Number of cells on SSW card.
  !!== From FIXCOM Module
integer, public :: mipts
  != Source particle type.
  !!== Number of source particle types specified in MCNP input file
  !!== From MCNP_INPUT Module
integer, public, ALLOCATABLE :: nsl(:,:)
  != Summary information for surface source file.
  !!== First index is:
  !!==   1: total tracks (all particle types)
  !!==   2: number of independent histories (all particle types)
  !!==   (ipt-1)*4+3: tracks (particle type ipt)
  !!==   (ipt-1)*4+4: independent tracks (particle type ipt)
  !!==   (ipt-1)*4+5: uncollided tracks (particle type ipt)
  !!==   (ipt-1)*4+6: independent uncollided tracks (particle type ipt)
  !!==   mipt: number of particle types
  !!== Second index loops over all surfaces (njss) and cells (nilw)
  !!== From MCNP_GLOBAL Module
```

2.2.g Particle Record:

a, b, pbl%r%wgt, pbl%r%erg, pbl%r%tme, pbl%r%x, pbl%r%y, pbl%r%z, pbl%r%u, pbl%r%v, c

```
    type(particle), public :: pbl
      != Particle information.
      !!== Real components are grouped into the ‘r’ member
      !!== The individual real components are defined below
      !!== From PBLCOM Module
    real(dknd) :: x
      != X-coordinate of the particle position.
    real(dknd) :: y
      != Y-coordinate of the particle position.
    real(dknd) :: z
      != Z-coordinate of the particle position.
    real(dknd) :: u
      != Particle direction cosine with X-axis.
    real(dknd) :: v
      != Particle direction cosine with Y-axis.
    real(dknd) :: w
      != Particle direction cosine with Z-axis.
      !!== The third direction cosine is calculated from the first two, and
      !!== its sign is determined by b (see below)
    real(dknd) :: erg
      != Particle energy.
    real(dknd) :: wgt
      != Particle weight.
    real(dknd) :: tme
      != Time at the particle position.
    REAL(DKND) :: a
      !!== History number of the particle (minus number of inactive particles
      !!== for KCODE)
      !!== Negative if the particle is uncollided
      !!== Local variable
    REAL(DKND) :: b
      !!== Packed variable
      !!== Much information is stored within this variable, not all is
      !!== documented here
      !!== Assume all neutrons are surface source particles:
      !!==   b=pack_ipt*real(pbl%i%ipt,dknd)
      !!==    =8*1    neutron
      !!==    =8*2    photon
      !!== Add 1 for cell source particles
      !!== The sign of b is the sign of the third direction cosine
      !!== Local variable
    REAL(DKND) :: c
      !!== Packed variable
      !!== Problem name of the surface (assuming surface particle)
      !!== For a macrobody, the facet number is written as a decimal,
```

```
!!==   e.g. 4.2 could be surface 2 of a BOX named 4
!!== Local variable
```

3. HOW TO WRITE AN MCNP SURFACE SOURCE FILE FROM ANOTHER CODE

3.1 Binary Formatting

Note that the MCNP input file must be written in binary format, and that Fortran binary files include record lengths at the beginning and end of each record. The record lengths are written as integers. The records must be grouped exactly as shown in Section 2.2. The corresponding record lengths are summarized in Table I for a system with a default integer size of 4 bytes and a default character size of 1 byte. Actual data type sizes may vary across machines.

3.2 Filling in Missing Data

When generating surface source information with a code other than MCNP, there may be no direct analog of the variables expected by MCNP in the alternate code. This is particularly likely in the first record (MCNP run and compilation information), fourth record (surface definitions), and summary record (summary of all tracks on the file).

Fortunately, much if this information is not needed by MCNP. The first and summary records are informational only, and filling them with placeholders is sufficient for MCNP to be able to use the file. MCNP does echo some of this information to an output file, in which case meaningless information will be present in the output. So long as the user is not expecting this information to be correct when linking the two codes, this is a non-issue.

It is not believed that the fourth record need define legitimate surfaces that correspond to the actual surfaces that the particles lie on. Defining legitimate surfaces is beyond the scope of this paper, and may require access to MCNP source code to learn. It is necessary to give a surface number on the particle record, though it may be arbitrary if the concept of surface number does not exist in the alternate code. The user should specify that same surface number using the OLD keyword on the SSR input card (see the MCNP manual [1]). It is recommended that the MCNP surface source read input file define a surface corresponding to the location of the recorded surface source, and that the number of this surface be specified using the NEW keyword on the SSR input card. Whether or not legitimate surfaces are defined in the fourth record, the correct size and types of data should be used.

An example of filling in missing data is provided in Section 4, which details how to create minimally functional MCNP surface source files.

Table I. Summary of Record Lengths for MCNP Surface Source File

	Data Types	Record Length
Format Record	character(len=8)	8
First Record	character(len=8) character(len=5) character(len=8) character(len=19) character(len=19) character(len=80) integer	143
Second Record	integer(i8knd) integer(i8knd) integer integer integer(i8knd)	32
Third Record	integer integer integer	12
Fourth Record	njss* (integer integer integer n*real(dknd)) njss = number of surfaces on the file n=number of coefficients needed to define	njss*(12+n*8)
Summary Record	double njss*(2+4*npt)*integer njss = number of surfaces on the file npt = number of particle types on the file	8+njss*(2+4*npt)*4
Particle Record	11*real(dknd)	88

4. GENERATING SURFACE SOURCE FILES FOR MCNP

In this section, we show how to create a minimally functional surface source file that is readable by MCNP. This also serves as an example of how to fill in missing data, as discussed in Section 3.2.

The instructions are presented for each record. The MCNP variable names corresponding to those used in Section 2 are shown on the left, with generic variable names from an alternate code shown on the right. Where appropriate, default values are provided in place of variable names. Comments are provided as needed for clarity.

4.1 Format Record:

id

id = “SF_00001”

4.2 First Record:

kods, vers, lods, idtms, probs, aids, knods

kods = <*Code name*>
// Genuine code name is not necessary

vers = <*Code version*>
// Genuine code version is not necessary

lods = <*CompilationDate*>
// Genuine compilation date is not necessary

idtms = <*date-time*>
// Date and time that the file was generated
// Genuine run date is not necessary
// Should be a character string with length=19 and format:
// ‘ MM/DD/YY hh:mm:ss ‘

probs = <*filename*>
// User specified surface source filename, minus extension

aids = <problem description string>

knods = 1

4.3 Second Record:

np1, nrss, nrcd, njsw, niss

np1 = -<*nStartingParticles*>

nrss = <*nParticles*>

nrcd = -11

njsw = 1

niss = <*nParticles*>

4.4 Third Record:

```
niwr, mipts, kjaq

      niwr  = 0

      mipts = 1
            // 1 specifies neutrons

      kjaq = 0
```

4.5 Fourth Record:

```
do j=1,njss
    if( kq==0 ) then
        jss(j),kst(jx),n,(scf(lsc(jx)+i),i=1,n)

njss = 1

kq = 0

jss(1) = 99999

kst(1) = 5
      // 5 specifies an “SO” surface, i.e., a sphere centered at the
      // origin
      // Valid surface information must be provided, but it need not
      // correspond to a genuine surface in the problem

n = 1
      // Only radius is need to define a sphere centered at the origin

scf(1) = -1.0
      // Legitimate radius not needed
      // Actual radius of surface can be written here if applicabale
```

4.6 Summary Record:

```
zero,((nsl(i,j),i=1,2+4*mipt),j=1,njss+nilw)


nsl(:,1) = (<nParticles>, -999, <nParticles>, -999, -999, -999)
      // Legitimate summary information not needed
```

4.7 Particle Record:

```
a, b, pbl%r%wgt, pbl%r%erg, pbl%r%tme, pbl%r%x, pbl%r%y, pbl%r%z, pbl%r%u,
pbl%r%v, c


      a = <id>

      b = sign(<z-direction>)*8*<particle type>
            // 8*1 specifies neutrons
            // 8*2 specifies photons

      pbl%r%wgt = <weight>

      pbl%r%erg = <energy>

      pbl%r%tme = <time>

      pbl%r%x = <x-position>

      pbl%r%y = <y-position>

      pbl%r%z = <z-position>

      pbl%r%u = <x-direction>

      pbl%r%v = <y-direction>

      c = 99999
            // Fake surface number if alternate code does not use surface
            // numbers
```

REFERENCES

1. D.B. PELOWITZ et. al., *MCNP6™ User's Manual*, LA-CP-13-00634, Los Alamos National Laboratory (2013).
2. T. GOORLEY et. al., "Initial MCNP6 release overview," *Nuclear Technology*, **180**(3), 298 (2012).