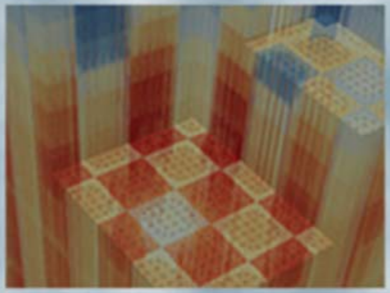
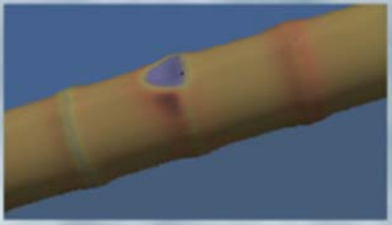
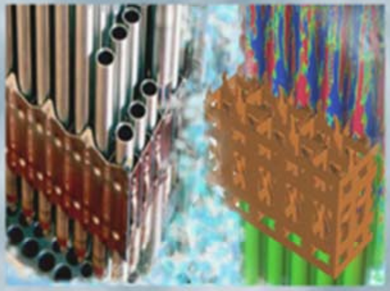
CASL-U-2016-1108-000

MPACT Standard Input User's Manual Version 2.2.0

June 9, 2016

REVISION LOG

Revision	Date	Affected Pages	Revision Description
0		All	Initial Version

Document pages that are:

Export Controlled ______None____

IP/Proprietary/NDA Controlled______None__________________________________

Sensitive Controlled______None____

Requested Distribution:

To: Unlimited distribution

Copy:

MPACT Standard Input User's Manual

Version 2.2.0

June 9, 2016

Contributors (in alphabetical order)

- Dr. Benjamin Collins (ORNL)
- Prof. Thomas J. Downar (UM)
- Andrew Fitzgerald (UM)
- Dr. Jess Gehin (ORNL)
- Andrew Godfrey (ORNL)
- Aaron Graham (UM)
- Daniel Jabaay (UM)
- Dr. Blake Kelley (formerly UM)
- Dr. Kang Seog Kim (ORNL)
- Dr. Brendan Kochunas (UM)
- Joel Kulesza (UM)
- Prof. Edward Larsen (UM)
- Dr. Yuxuan Liu (UM)
- Dr. Zhouyu Liu (formerly UM)
- Prof. William R. Martin (UM)
- Dr. Adam G. Nelson (formerly UM)
- Dr. Scott Palmtag (ORNL)
- Michael Rose (UM)
- Dr. Thomas Saller (formerly UM)
- Dr. Shane Stimpson (ORNL)
- Dr. Travis Trahan (formerly UM)
- Jipu Wang (UM)
- Dr. Will Wieselquist (ORNL)
- Mitchell T.H. Young (UM)
- Ang Zhu (UM)

Contents

1 Introduction **1**

2 Executing MPACT **3**
2.1 Standalone Serial Execution . . . 3
2.2 Standalone Parallel Execution . . . 3

3 Input File Structure **5**
3.1 Native MPACT Input . . . 5

4 Description of MPACT Native Input Blocks and Cards **7**
4.1 CASEID Block . . . 8
4.2 DEPL Block . . . 8
4.2.1 KERNEL Card . . . 8
4.2.2 SUBSTEP Card . . . 9
4.2.3 T_UNIT Card . . . 9
4.2.4 TIME_STEP_METHOD Card . . . 9
4.2.5 RESOXS_DT Card . . . 9
4.2.6 BURNUP_DT Card . . . 10
4.3 EDIT Block . . . 10
4.3.1 PNUM Card . . . 10
4.3.2 ISUM Card . . . 10
4.3.3 CHECKPOINT Card . . . 11
4.3.4 EDT_BURN Card . . . 11
4.4 GEOM Block . . . 11
4.4.1 FILE Card . . . 12
4.4.2 MOD_DIM Card . . . 12
4.4.3 PINMESH Card . . . 13
4.4.4 PIN Card . . . 15
4.4.5 MODULE Card . . . 16
4.4.6 LATTICE Card . . . 16
4.4.7 ASSEMBLY Card . . . 17
4.4.8 CORE Card . . . 17
4.5 MATERIAL Block . . . 17
4.5.1 MAT Card . . . 17
4.6 OPTION Block . . . 19
4.6.1 BOUND_COND Card . . . 20
4.6.2 ITER_LIM Card . . . 20
4.6.3 CONV_CRIT Card . . . 20
4.6.4 SOLVER Card . . . 21
4.6.5 RAY Card . . . 21
4.6.6 PARALLEL Card . . . 22
4.6.7 VIS_EDITS Card . . . 23
4.6.8 EXP_TABLE Card . . . 23
4.6.9 VALIDATION Card . . . 24

4.6.10 SCATT_METH Card 24
4.6.11 CMFD Card 24
4.6.12 CMFD_SHIFT_METHOD Card 25
4.6.13 CMFD_NODAL Card 26
4.6.14 BC_AVERAGE Card 26
4.6.15 UNDERRELAX Card 27
4.6.16 NODAL Card 27
4.6.17 POWER_EDIT Card 28
4.6.18 PARAM Card 29
4.6.19 VERBOSITY Card 29
4.6.20 AMPX_OUT Card 30
4.6.21 CRITBORON Card 30
4.6.22 XENON Card 30
4.6.23 RESTART Card 31
4.6.24 AXIAL_TL Card 31
4.6.25 SIMPLIFIEDTH Card 32
4.6.26 CPM_INIT Card 32
4.6.27 SUBPLANE Card 33

4.7 STATE Block 33
4.7.1 RATED_POWER Card 33
4.7.2 CORE_POWER Card 34
4.7.3 RATED_FLOW Card 34
4.7.4 TINLET Card 34
4.7.5 BORON Card 34
4.7.6 PRESSURE Card 35

4.8 XSEC Block 35
4.8.1 XSLIB Card 35
4.8.2 ADDPATH Card 36
4.8.3 XSSHIELDER Card 37

Chapter 1

Introduction

The MPACT (**M**ichigan **P**Arallel **C**haractistics based **T**ransport) code is designed to perform high-fidelity light water reactor (LWR) analysis using whole-core pin-resolved neutron transport calculations on modern parallel-computing hardware. The code consists of several libraries which provide the functionality necessary to solve steady-state eigenvalue problems. Several transport capabilities are available within MPACT including both 2-D and 3-D Method of Characteristics (MOC). A three-dimensional whole core solution based on the 2D-1D solution method provides the capability for full core depletion calculations.

Specific features available in the current release of MPACT are:

- Support for Microsoft Windows Operating Systems (32-bit and 64-bit)
- Support for Linux-based Operating Systems (32-bit and 64-bit)
- OpenMP parallelism for MOC sweeps
- Support for MPACT and AMPX working cross section library formats
- Steady-state eigenvalue calculations using power iteration
- 2-D and 3-D MOC transport solvers
- 2D-1D full core solution
- Depletion capability
- Generalized pressurized water reactor (PWR) geometry
- Export of computational and results mesh to VTK files
- Visualization via VisIt

The purpose of this document is to provide users with sufficient background to be able to utilize MPACT for PWR design and analysis applications. For a more detailed description of the methods or sofware design, the reader is directed to the theory and programmer's documentation.

This user document is divided into several chapters. After this introduction, an overview is provided regarding code execution capabilities and limitations in both serial and parallel environments. Finally, a detailed description of the user input is provided.

For specific questions about the use of MPACT, the licensing of the code, or to report bugs users are encouraged to send an email to `support@casl.gov`. When reporting bugs, users are requested to attach the problematic input to the email and to provide information in the body of the email about the code version, machine, runtime environment and any other relevant details to permit debugging.

Explanation of Notation

In several of the code examples that follow in this document a specific syntax is used which can be described as:

- Words appearing in typewriter font within the normal text, such as `this` indicate the word is a reference to a something that is used in an example. Paragraphs or lines of text in this formatting are examples of usage.
- Words bracketed with '`<`' and '`>`' in examples such as `<token>` indicate a single token for which a value is expected. This value is typically some intrinsic data type such as an integer, string, real, or logical.
- Tokens that are bracketed by '`[`' and '`]`' in examples such as `[<token>]` indicate an optional value. Optional values may become nested such as `[<token1> [<token2>]]`
- A vertical bar '`|`' in an example indicates only one out of the set should be used. The set will be defined by one of the above sets of brackets. For example, `[<token1> | <token2>]` means that only one of `<token1>` or `<token2>` should be entered and that this entry is optional.
- Tokens appended by a '`(:)`' indicate an array of values, where the number of '`:`' indicates the number of dimensions of the array. For example, `<matrix(:,:)>` is a token that is a 2-D array.

Additionally, the input cards presented in Chapter 4 are described in the following tabular format:

`The name of the input card`	Argument data type(s) (e.g. Integer)	Required / Optional
Units: The default units associated with this card, if any. (default), Other units that can be associated with this card, if any.		
Applicable Value(s): The default values if this card is optional. (default), A list of allowable values for this card, and any explanation of those values.		
Limitation(s): This section explains the limitations of this card, specifically in regards to its interactions with other inputs.		
Description: This section offers a more verbose description of the card. It also describes interactions this card may have with other cards.		
Notes: The notes section is a catch-all for information that is not elsewhere in the table.		

Chapter 2

Executing MPACT

Depending on how MPACT was configured, compiled, and installed, it may be executed in serial and/or parallel. The following sections outline the procedures for running the code in either serial or parallel. When run as a standalone analysis tool, MPACT is executed from the command line.

2.1 Standalone Serial Execution

The syntax for MPACT is shown below:

```
$> <path_to_MPACT>/mpact.exe [<input_file> [<output_file> [<log_file>]] | -
    help]
```

All command line arguments are optional. The meaning of each is described as:

- `-help` - Displays the help message. This message describes the command line arguments and their usage.
- <`input_file`> - The name of the input file to process. If no input file is listed, then MPACT tries to process the file `mpact.inp` in the present working directory. The <`input_file`> may include an absolute or relative path to the file. This file must exist and be readable prior to execution.
- <`output_file`> - The file to use for writing the default output. If no file is listed then a file with <casename>.out will be created in the present working directory. In general, if the output file does not exist it will be created, and if it already exists it will be replaced **without** warning. <`output_file`> may include an absolute or relative path. The directory pointed to by the path must exist prior to execution.
- <`log_file`> - The file to use for writing the execution log information. If no file is listed then a file with <casename>.log will be created in the present working directory. In general, if the log file does not exist, it will be created, and if it already exists it will be replaced **without** warning. <`log_file`> may include an absolute or relative path. The directory pointed to by the path must exist prior to execution.

2.2 Standalone Parallel Execution

MPACT may only be executed in parallel if a parallel build has been installed. MPACT uses two kinds of parallel models. The first is the shared memory model which is based on the OpenMP standard (`http://www.openmp.org`) and the other is a distributed memory model which is based on the MPI Standard. If the MPACT executable is built with MPI then it is executed differently than in serial, but otherwise the serial description in the previous section is correct. When executing MPACT with MPI the command has the following syntax:

```
$> <mpirun_cmd> [<mpi_options>] <mpact_exe> [<mpact_options>]
```

- <`mpirun_cmd`> - This is the command used to launch MPI executables. This command can vary because different machines may have different implementations of the MPI library installed. Therefore, it is suggested the user consult the documentation for their cluster or workstation for this command and its arguments. For most implementations of MPI (such as OpenMPI, `http://www.open-mpi.org/`) the <`mpirun_cmd`> command is `mpirun`.
- <`mpirun_options`> - These are command line arguments for <`mpirun_cmd`>. Again, the user should consult their machine's documentation for usage.
- <`mpact_exe`> - The name of the MPACT executable that is installed. Typically, one should include the full path to the executable since the parallel execution environment may not have the same PATH setting as the run time environment in which the <`mpirun_cmd`> was invoked.
- <`mpact_options`> - The command line arguments for MPACT. See the Serial Execution section of this chapter for a complete description.

Chapter 3

Input File Structure

3.1 Native MPACT Input

One can use MPACT's own input processor for reading ASCII formatted input files. The purpose of this chapter is to describe general rules and conventions of how the input file is processed. Some general formatting rules are:

- Input is generally free formatted, unless otherwise indicated by the specific syntax of a CARD. This means continuous blocks of white space are treated as a single block of white space, whether it is spaces, blank lines or tabs.
- All input lines must not exceed 256 characters.
- `'!'` is the comment symbol. In general it may appear at the beginning of a line or at the end of a line of input, unless otherwise indicated by a specific CARD. All text on a line after the comment symbol is ignored.
- `'.'` is a special symbol that indicates the end of input when it appears in the first column. No lines after this symbol are processed.
- `'*'` is a special symbol to indicate a token is to be repeated. The syntax for this symbol is: "`r*n`" where the token `n` is repeated `r` times. `r` must be an integer and `n` may be any single intrinsic type such as an integer, real, logical, or string. As an example, "`3*4` would repeat integer entry 4 three times.
- If a string input entry contains a space then it must be enclosed by double quotation marks.
- Logicals must be indicated by the single characters `T` for true or `F` for false. These entries are not case sensitive.
- The input is not case sensitive (unless otherwise noted). The main exception to this is that any directories and/or filename are case sensitive. Capitalization is used throughout this document as a way to help the user distinguish words that have special meaning within MPACT.
- When inputing floating point numbers, any of the following formats are acceptable:

```
1
1.
1.0
1.0e-0
1.0e0
1.0e+00
1.0E+0
1.0E+000
1.0d0
```

The native input file is based on the concepts of BLOCKS and CARDS. The input is made up of several BLOCKS; each BLOCK is made up of several CARDS. Some general rules for BLOCKS and CARDS are:

- All BLOCK and CARD names are **NOT** case sensitive.
- All BLOCK names must start *in* the first column.
- All CARD names must start *after* the first column.
- No text must appear on the same line as a BLOCK name with the exception of the `CASEID` block
- BLOCKS may appear in any order.
- CARDS within a block may generally appear in any order within the BLOCK with the exception of the `GEOM` block.

Chapter 4

Description of MPACT Native Input Blocks and Cards

This chapter describes each of the blocks and cards that can be used in the input. Each block will be briefly described and each of the associated cards will be described in a tabular format including the following information:

- Card name used by MPACT to uniquely identify the associated input.
- Card data input format (e.g., free-form character string, fixed character string, integer, floating-point real number).
- Whether the card is required or optional for a given block.
- Available unit to associate with the card data input. If multiple units are available, the default units (if the units are left unspecified) will be indicated.
- Applicable value and/or range of values (for a given set of units). Unless otherwise specified, other units will have a corresponding, appropriately scaled based on the unit conversion, range of applicablity. The default value (if left unspecified) will be indicated.
- Limitations on the input. These limitations might arise from conflicts that arise from two incompatible inputs being used.
- A technical description of the ramifications of using a particular card.
- General notes regarding use. These notes might include cautionary notes, best practices, general guidance, etc.

In general, comments can be entered throughout the input to assist the analyst in documenting his or her input. Specific details regarding these comments are as follows (and will not be repeated later):

- Only the '!' indicates that the following text is a comment.
- A '!' cannot appear in the first column of the input file.

4.1 CASEID Block

The CASEID block is used to define a unique identifying string for the calculation. It is the first block of the input that is processed. It is also the only block that consists of a single line with input on the same line as the block name. It has no cards and must only appear once within a given input file. The case name is used to generate the filenames of any file created during execution of the case. Typically this includes the output file, log file, and visualization files. The syntax for the block is given below.

```
CASEID ‘‘<case_name>’’ [‘‘<description>’’] [!<comment>]
```

A detailed description of the required and optional parameters is as follows:

caseid	Free-Form Character Strings, Max. Length: 400	Required
Units: N/A		
Applicable Value(s): , Combination of upper or lower case characters, spaces, dashes, underscores. These values are applicable for both the `case_name` and `description`.		
Limitation(s): Strings provided over 120 characters will be truncated when creating visualization files. Strings with spaces in them must be encapsulated with blocking double quotation marks.		
Description: The `case_name` is used as an overall identifier for the case being run as well as all files created as a result of running this case. The `description` is used to provide a more verbose description of the contents of the input file than what is available (or appropriate) to use in the `case_name`.		
Notes: If the `case_name` contains spaces these will be replaced by underscore characters for file naming purposes.		

4.2 DEPL Block

The DEPL block is for providing radioisotope depletion input parameters. The use of this block is **optional**. If used, the cards have the following format:

```
kernel [<BATEMAN|ORIGEN>] [!<comment>]
substep <user_depsubstep> [!<comment>]
t_unit [<DAYS | EFPD | MWDKG | GWDMT>] [!<comment>]
time_step_method [<P-C | None| SemiP-C | SemiP-C_PostCorr |others >] [
     !<comment>]
resoxs_dt <user_resoxs_dt> [!<comment>]
burnup_dt <t1 t2 t3:t4:t_interval> [!<comment>]
```

Each of these cards will be described in more detail in the following subsections.

4.2.1 KERNEL Card

```
kernel  [<CRAM|BATEMAN|ORIGEN>] [!<comment>]
```

kernel	Fixed Character String	Optional
Units: N/A		
Applicable Value(s): BATEMAN (default), ORIGEN		
Limitation(s): None		
Description: This card is used to select the point depletion solver for depletion calculation. The BATEMAN solver is an internal point depletion solver, while ORIGEN solver is an external point depletion solver coupled with ORIGEN.		
Notes: The internal BATEMAN solver utilizes fewer isotopes than ORIGEN and runs much faster.		

4.2.2 SUBSTEP Card

```
substep <user_depsubstep> [!<comment>]
```

`substep`	Integer	Optional
Units: N/A		
Applicable Value(s): 1 (default)		
Limitation(s): Positive		
Description: This card is used to read the number of substep for the Depletion step. The substep method is applied to perform multiple depletion calculations between transport calculations. Since the depletion calculation typically takes less time than the transport calculation this will often save computational time.		
Notes: 2 or 3 substeps per depletion step is recommended.		

4.2.3 T_UNIT Card

```
t_unit [<DAYS | EFPD | MWDKG | GWDMT>] [!<comment>]
```

`t_unit`	Fixed Character String	Optional
Units: N/A		
Applicable Value(s): `GWDMT` (default), `DAYS`, `EFPD`, `MWDKG`		
Limitation(s): None		
Description: This card is used to specify the units for the burnup time. MPACT will convert the units automatically.		
Notes: `MWDKG` and `GWDMT` are the same units		

4.2.4 TIME_STEP_METHOD Card

```
time_step_method [<P-C | None | SemiP-C | PostCorrector | others >] [!<comment
     >]
```

`time_step_method`	Fixed Character String	Optional
Units: N/A		
Applicable Value(s): `P-C` (default), `None`,`SemiP-C`,`PostCorrector`		
Limitation(s): None		
Description: This card is used to specify the time step method for the Depletion module. `P-C` denotes the Predictor-Corrector method, `SemiP-C` denotes the semi-Predictor-Corrector method, `PostCorrector` denotes the semi-Predictor-Corrector method with Post Corrector, and None means the standard time step method.		
Notes: Predictor-Corrector method is a well demonstrated depletion time step method and it can perform depletion using large time step.		

4.2.5 RESOXS_DT Card

```
resoxs_dt <user_resoxs_dt> [!<comment>]
```

`resoxs_dt`	Floating-point Real Number	Optional
Units: Same as `T_UNIT` (default)		
Applicable Value(s): 10 (default)		
Limitation(s): Positive		

continued on next page...

`resoxs_dt`, continued...

Description: This card is used to read the time interval for calling subgroup calculation in the Depletion module. When depletion is performed, the subgroup calculation is required if background XSEC changs noticeably due to the composition changes
Notes: The default 10 GWDMT time interval is efficient for most applications.

4.2.6 BURNUP_DT Card

```
BURNUP_DT <t1 t2 t3:t4:t_interval> [!<comment>]
```

burnup_dt	Array of Floating-point Real Numbers	Required
Units: Same as T_UNIT (default)		
Applicable Value(s):		
Limitation(s): Positive		
Description: This card is used to define the time steps for the Depletion module. The format is t1 t2 t3:t4:t_interval, where t1 and t2 explicitly define the time steps and t3:t4:t_interval defines the beginning time t3, ending time t4 and the time interval. The follwing two inputs are identical: 1 2 3 4 5 and 1:5:1		
Notes: None		

4.3 EDIT Block

The `EDIT` block is for specifying edit input parameters. The use of this block is **optional**. If used, the cards have the following format:

```
PNUM <T|F> [!<comment>]
ISUM <T|F> [!<comment>]
CHECKPOINT <format> [<filename>] [!<comment>]
EDT_BURN <t1 t2 t3:t4:t_interval> [!<comment>]
```

Each of these cards will be described in more detail in the following subsections.

4.3.1 PNUM Card

```
PNUM <T|F> [!<comment>]
```

pnum	Fixed Character String	Optional
Units: N/A		
Applicable Value(s): F (default), T		
Limitation(s): None		
Description: This card is used to control printing isotope number density. `T` enables printing, while `F` disables printing.		
Notes: None		

4.3.2 ISUM Card

```
ISUM <T|F> [!<comment>]
```

`isum`	Fixed Character String	Optional
Units: N/A		
Applicable Value(s): F (default), T		
Limitation(s): None		
Description: This card is used to control printing isotope summary. `T` enables printing, while `F` disables printing.		
Notes: None		

4.3.3 CHECKPOINT Card

```
CHECKPOINT [<DA32|HDF5>] [<filename>] [!<comment>]
```

`checkpoint`	Fixed & Free-form Character String	Optional
Units: N/A		
Applicable Value(s): , Format: 'DA32' or 'HDF5' character strings. Filename: Combination of upper or lower case characters, spaces, dashes, underscores.		
Limitation(s): None		
Description: This card is used to control the format and filename of the checkpoint file.		
Notes: None		

4.3.4 EDT_BURN Card

```
EDT_BURN <t1 t2 t3:t4:t_interval> [!<comment>]
```

`edt_burn`	Array of Integers, Arbitrary Length	Optional
Units: Same as `T_UNIT` (default)		
Applicable Value(s): N/A (default), Positive integers		
Limitation(s): None		
Description: This card is used to control burnup steps to print results. The burnup steps in this card should be within the time steps defined in the depletion block, otherwise the closest time step result will be printed. There are two ways to define the time steps: 1 2 3 4 5 or 1:5:1. The combination of both methods is also recognized.		
Notes: None		

4.4 GEOM Block

The `GEOM` block is used for specifying the model geometry and spatial mesh. It is the most complicated part of the input. It is the fourth block to be processed when reading the input file. It contains the cards: `MOD_DIM`, `PINMESH`, `PIN`, `MODULE`, `LATTICE`, `ASSEMBLY`, `CORE`, and `FILE`. At least one instance of each of these cards must appear in the `GEOM` block, with the exception of the `FILE` card. Any of the cards may appear an arbitrary number of times within the block. The order of appearance of the cards is also important because the values entered for a card refer to objects that must be defined in cards that are processed prior to the current card. The cards must appear in the following order within the block.

1. `MOD_DIM`
2. `PINMESH`
3. `PIN`
4. `MODULE`

5. LATTICE
6. ASSEMBLY
7. CORE

The optional FILE card can also appear anywhere in this block, and functions like a direct insertion of the text of the file into the current line. Therefore, care should be taken when using this card that the contents of the file adhere to the order of the overall input.

Briefly, the cards have the following input format:

```
file ‘‘<path/file_name>’’ [!<comment>]
mod_dim <x> <y> <z1:zN> [!<comment>]
pinmesh <id> rec <xvals(:)> / <yvals(:)>                 / <zvals(:)> / <ndivx(:)> / <ndivy(:)> / <ndivz(:)> [!<comment>]
pinmesh <id> cyl  <r(:)>    / <pitch>                    / <zvals(:)> / <ndivr(:)> / <ndiva(:)> / <ndivz(:)> [!<comment>]
pinmesh <id> qcyl <r(:)>    / <pitch> <iquad>            / <zvals(:)> / <ndivr(:)> / <ndiva(:)> / <ndivz(:)> [!<comment>]
pinmesh <id> gcyl <r(:)>    / <xMin> <xMax> <yMin> <yMax> / <zvals(:)> / <ndivr(:)> / <ndiva(:)> / <ndivz(:)> [!<comment>]
pin <pin_id> <pin_mesh_id> / <mat_id_list> [!<comment>]
module <id> <nx> <ny> <nz> [!<comment>]
 <pid(1,1)> ... <pid(nx,1)>
    .             .
    .             .
    .             .
<pid(1,ny)> ... <pid(nx,ny)>
lattice <id> <nx> <ny> [name ][!<comment>]
 <mid(1,1)>  ... <mid(nx,1)>
   .               .
   .               .
   .               .
 <mid(1,ny)> ... <mid(nx,ny)>
 assembly <id> [<name>] [!<comment>]
   <lattice_ids(:)>
 core [<symmetry_opt>] [!<comment>]
   <assembly_ID_map(:,:)>
```

Each of these cards will be described in more detail in the following subsections.

4.4.1 FILE Card

```
file ‘‘<path/file_name>’’ [!<comment>]
```

file	Free-form Character String	Optional
Units: N/A		
Applicable Value(s):		
Limitation(s): None		
Description: This card is included as a convenience to insert the contents of another file into the current line. This has the convenience that for models with very long and complicated geometric descriptions the top level input file may be shortened for readability. the <path_file_name> argument is a string containing the full or relative (to the main input file directory) path to the file and the file name with the extension of the file to be included.		
Notes: None		

4.4.2 MOD_DIM Card

```
mod_dim <x> <y> <z1:zN> [!<comment>]
```

mod_dim	Array of Positive Double-Precision Real Numbers, Minimum Length 3	Required
Units: cm (default)		
Applicable Value(s): , All entries require positive real numbers. For a 3-D problem, the *z* dimension of every axial plane's thickness must be specified. The order of the *z* entries does not matter.		
Limitation(s): None		
Description: This card is for specifying the dimensions of the ray tracing modules. All three dimensions must be specified regardless of whether using a 2-D or 3-D transport solver. The three dimensions correspond to the *x*-, *y*- and *z*- dimension of all ray tracing modules. Note that the positive *z*-direction is opposite to the direction of gravitational force.		
Notes: None		

4.4.3 PINMESH Card

The PINMESH card describes the flat source region mesh to use for a pin cell in the reactor. It is the most complicated card of the input. Several types of pin mesh are definable, and they are described below. All pin shapes are cuboids. The type of pin is selected using a mnemonic for each pin type.

Rectangular Mesh Pin: The syntax for defining a pin mesh made up of a 3-D rectilinear grid is:

```
pinmesh <id> rec <xvals(:)> / <yvals(:)> / <zvals(:)> / <ndivx(:)> / <ndivy(:)
     > / <ndivz(:)> [!<comment>]
```

- <id> - an integer id for this pin mesh. It must be unique amongst all PINMESH defined in the model. IDs may be nonsequential, and include any positive 32-bit integer.
- <xvals(:)> - The x-coordinates marking material interfaces within the pin. The array must have a size of at least 1, and the last value listed is taken to be the pin pitch in the x-direction. All values must be positive reals and they must be listed in ascending order.
- <yvals(:)> - The y-coordinates marking material interfaces within the pin. The array must have a size of at least 1, and the last value listed is taken to be the pin pitch in the y-direction. All values must be positive reals and they must be listed in ascending order.
- <zvals(:)> - The z-coordinates marking material interfaces within the pin. The array must have a size of at least 1, and the last value listed is taken to be the pin pitch in the z-direction. All values must be positive reals and they must be listed in ascending order. If a 2-D transport method is going to be used, then there should only be one <zvals(:)> and <zvals(:)>=Pz.
- <ndivx(:)> - The number of equally spaced flat source regions to use when dividing the grid defined by <xvals(:)>. The number of entries here must equal the number of entries in <xvals(:)>. All values are positive integers.
- <ndivy(:)> - The number of equally spaced flat source regions to use when dividing the grid defined by <yvals(:)>. The number of entries here must equal the number of entries in <yvals(:)>. All values are positive integers.
- <ndivz(:)> - The number of equally spaced flat source regions to use when dividing the grid defined by <zvals(:)>. The number of entries here must equal the number of entries in <zvals(:)>. All values are positive integers. If a 2-D transport method is going to be used, then <ndivz(:)>=1.

Centered Cylindrical Mesh Pin: The syntax for defining a pin mesh made up of concentric cylinders is:

```
pinmesh <id> cyl <r(:)> / <pitch> / <zvals(:)> / <ndivr(:)> / <ndiva(:)> / <
     ndivz(:)> [!<comment>]
```

- <id> - an integer id for this pin mesh. It must be unique amongst all PINMESH defined in the model. IDs may be nonsequential, and include any positive 32-bit integer.
- <r(:)> - is an array of radii for indicating the different material interfaces. The array must have a size of at least 1. All values must be positive reals and they must be listed in ascending order.
- <pitch> - is the pitch of the pin. This assumes that the pin is square.
- <zvals(:)> - The z-coordinates marking material interfaces within the pin. The array must have a size of at least 1, and the last value listed is taken to be the pin pitch in the z-direction. All values must be positive reals and they must be listed in ascending order. If a 2-D transport method is going to be used, then there should only be one <zvals(:)> and <zvals(:)>=Pz.
- <ndivr(:)> - The number of equal-volume rings to use when dividing the concentric cylinders defined by <r(:)> into flat source regions. The number of entries here must equal the number of entries in <r(:)>.
- <ndiva(:)> - The number of equal-angle "pie-slices" to use when dividing each concentric ring defined by <r(:)>+<ndivr(:) into flat source regions azimuthally. The number of entries here must equal the sum of the values in <ndivr(:)> plus 1.
- <ndivz(:)> - The number of equally spaced flat source regions to use when dividing the grid defined by <zvals(:)>. The number of entries here must equal the number of entries in <zvals(:)>. All values are positive integers. If a 2-D transport method is going to be used, then <ndivz(:)>=1.
- <xMin> <xMax> <yMin> <yMax> are optional, and if omitted are assumed to be a half-pitch in each direction (in other words, the pin is centered).

Quarter-Cylindrical Mesh Pin: The syntax for defining a pin mesh made up of concentric cylinders centered at a pin corner is:

```
pinmesh <id> qcyl <r(:)> / <pitch> <iquad> / <zvals(:)> / <ndivr(:)> / <ndiva(
     :)> / <ndivz(:)> [!<comment>]
```

- <id> - an integer id for this pin mesh. It must be unique amongst all PINMESH defined in the model. IDs may be nonsequential, and include any positive 32-bit integer.
- <r(:)> - is an array of radii for indicating the different material interfaces. The array must have a size of at least one. All values must be positive reals and they must be listed in ascending order.
- <pitch> - is the pitch of the pin. This assumes that the pin is square.
- <iquad> - an integer on [1,4] to indicate which corner to use for the center of rotation. The values correspond to the following corners
  - 1 - south-west corner
  - 2 - south-east corner
  - 3 - north-east corner
  - 4 - north-west corner
- <zvals(:)> - The z-coordinates marking material interfaces within the pin. The array must have a size of at least 1, and the last value listed is taken to be the pin pitch in the z-direction. All values must be positive reals and must be listed in ascending order. If a 2-D transport method is going to be used, then there should only be one <zvals(:)> and <zvals(:)>=Pz.
- <ndivr(:)> - The number of equal-volume rings to use when dividing the concentric cylinders defined by <r(:)> into flat source regions. The number of entries here must equal the number of entries in <r(:)> minus 1.

- <`ndiva(:)`> - The number of equal-angle "pie-slices" to use when dividing each concentric ring defined by <`r(:)`>+<`ndivr(:)` into flat source regions azimuthally. The number of entries here must equal the sum of the values in <`ndivr(:)`> plus 1.
- <`ndivz(:)`> - The number of equally spaced flat source regions to use when dividing the grid defined by <`zvals(:)`>. The number of entries here must equal the number of entries in <`zvals(:)`>. All values are positive integers. If a 2-D transport method is going to be used, then <`ndivz(:)`>`=1`.

General Cylindrical Mesh Pin: The syntax for defining a pin mesh made up of a concentric cylinders centered at (0,0) with arbitrary pin boundaries is:

```
pinmesh <id> gcyl <r(:)> / <xMin> <xMax> <yMin> <yMax> / <zvals(:)> / <ndivr(:
     )> / <ndiva(:)> / <ndivz(:)> [!<comment>]
```

- <`id`> - an integer id for this pin mesh. It must be unique amongst all `PINMESH` defined in the model. IDs may be nonsequential, and include any positive 32-bit integer.
- <`r(:)`> - is an array of radii for indicating the different material interfaces. The array must have a size of at least one. All values must be positive reals and must be listed in ascending order.
- <`xMin`><`xMax`><`yMin`><`yMax`> - Specify the boudaries of the pin.
- <`zvals(:)`> - The z-coordinates marking material interfaces within the pin. The array must have a size of at least 1, and the last value listed is taken to be the pin pitch in the z-direction. All values must be positive reals and must be listed in ascending order. If a 2-D transport method is going to be used, then there should only be one <`zvals(:)`> and <`zvals(:)`>`=Pz`.
- <`ndivr(:)`> - The number of equal-volume rings to use when dividing the concentric cylinders defined by <`r(:)`> into flat source regions. The number of entries here must equal the number of entries in <`r(:)`> minus 1.
- <`ndiva(:)`> - The number of equal-angle "pie-slices" to use when dividing each concentric ring defined by <`r(:)`>+<`ndivr(:)` into flat source regions azimuthally. The number of entries here must equal the sum of the values in <`ndivr(:)`> plus 1.
- <`ndivz(:)`> - The number of equally spaced flat source regions to use when dividing the grid defined by <`zvals(:)`>. The number of entries here must equal the number of entries in <`zvals(:)`>. All values are positive integers. If a 2-D transport method is going to be used, then <`ndivz(:)`>`=1`.

4.4.4 PIN Card

The `PIN` card is for applying a set of materials to a given `PINMESH`. The syntax for this card is:

```
pin <pin_id> <pin_mesh_id> / <mat_id_list> [!<comment>]
```

- <`pin_id`> - an integer id for this pin. It must be unique amongst all `PIN` defined in the model. IDs may be nonsequential, and include any positive 32-bit integer.
- <`pin_mesh_id`> - an integer id for a valid pin mesh id previously defined in a `PINMESH` card.
- <`mat_id_list`> - a list of integer IDs for valid materials defined in the `MAT` card of the `MATERIAL` block. The number of entries must be equal to the number of uniform material regions defined in the pin mesh. The order of the entries depends on the mesh type.
  - For Rectangular Mesh Pins the numbers are given for lexicographical ordering (e.g. left-to-right then top-to-bottom).

- For Cylindrical Mesh Pins the numbers are given in a "in-out" manner for each ring in the mesh. Only each specified radius needs a material definition, not sub-regions. If the pin mesh has multiple axial levels, material ids are specified for each level, again, not for sub-levels.
- For Quarter-Cylindrical Mesh Pins the ordering is the same as the Cylindrical Mesh Pins.

4.4.5 MODULE Card

This card is for defining a ray tracing module which is a 3-D array of pins. A 2-D array of pin IDs is input in the card, and then extruded. The pin boundaries must make a structured rectilinear grid within the module and the min/max dimensions of the pin grid must conform with the modular ray tracing dimensions defined in the `MOD_DIM` card. The syntax for the card is:

```
module <id> <nx> <ny> <nz> [<modsym>] [!<comment>]
 <pid(1,1)> ... <pid(nx,1)>
    .             .
    .             .
    .             .
<pid(1,ny)> ... <pid(nx,ny)>
```

- `<id>` - an integer id for this ray tracing module geometry. It must be unique amongst all `MODULE` defined in the model. IDs may be nonsequential, and include any positive 32-bit integer.
- `<nx>` - the number of pins along the x-direction. Must be a positive integer.
- `<ny>` - the number of pins along the y-direction. Must be a positive integer.
- `<nz>` - the number of pins along the z-direction. Must be a positive integer.
- `[<modsym>]` - the module symmetry option. This input describes if the module is a full or quarter module. If it is a quarter, it specifies which quarter the module represents. Valid inputs are FULL, NE, NW, SE, or SW. Must be a string.
- `<pid(:,:)>` - a 2-D array of valid integer pin IDs defined previously with the `PIN` card. This array gets extruded `<nz>` times in the z-direction. The number of entries on a line must equal `<nx>` and the array must span `<ny>` lines.

The ray tracing module is not a natural engineering structure, so as a general guideline when building models to minimize computational resource usage the ray tracing module should represent the *smallest* unit of repeatable geometry in the model. In some special cases this may be a single pin-cell. For most practical cases this may be a quarter-assembly, and for the most general cases this should not exceed a full lattice description of a single assembly.

4.4.6 LATTICE Card

This card is used to define a lattice geometry. Lattice meshes are defined in terms of ray tracing modules and represent the geometry for the full x-y geometrical description of an assembly. All lattices within the same assembly must have the same values for `<nx>` and `<ny>`.

```
lattice <id> <nx> <ny> [name ][!<comment>]
 <mid(1,1)>  ... <mid(nx,1)>
   .                .
   .                .
   .                .
 <mid(1,ny)> ... <mid(nx,ny)>
```

- `<id>` - an integer id for this lattice geometry. It must be unique amongst all lattices defined by the `LATTICE` card in the model. IDs may be nonsequential, and include any positive 32-bit integer.

- <nx> - the number of pins along the x-direction. Must be a positive integer.
- <ny> - the number of pins along the y-direction. Must be a positive integer.
- <name> - an optional descriptive name may also be provided for referencing this lattice type. Presently the name is not used by any other part of the code.
- <mid(:,:)> - a 2-D array of valid integer ray tracing module IDs that were defined previously with the MODULE card. The number of entries on a line must equal <nx> and the array must span <ny> lines.

4.4.7 ASSEMBLY Card

This card is used for defining an assembly geometry. An assembly is defined in terms of lattices given as a 1-D array.

```
assembly <id> [<name>] [!<comment>]
  <lattice_ids(:)>
```

- <id> - an integer id for this lattice geometry. It must be unique amongst all lattices defined by the LATTICE card in the model. IDs may be nonsequential, and include any positive 32-bit integer.
- <name> - an optional descriptive name may also be provided for referencing this assembly type. Presently the name is not used by any other part of the code.
- <lattice_ids(:)> - a 1-D array of valid integer lattice IDs that were defined previously with the LATTICE card. The number of entries on a line must be the same for all ASSEMBLY cards. The assembly is built using the leftmost lattice ID as the bottom and the rightmost as the top.

4.4.8 CORE Card

This card is used for specifying the CORE radial configuration. The syntax for this card is:

```
core [<symmetry_opt>] [!<comment>]
  <assembly_ID_map(:,:)>
```

- <symmetry_opt> - is an optional input indicating the core symmetry. Presently only 360 symmetry (or no symmetry) is supported.
- <assembly_ID_map(:,:)> - is the 2-D map of the core assembly locations. The map must be internally continuous (i.e. no empty locations inside the boundary), but is allowed to have a "stair-case" like boundary. The values must be valid integer assembly IDs of assemblies defined previously with the ASSEMBLY card.

4.5 MATERIAL Block

The MATERIAL block is for specifying the materials to be used in the calculation. This block is the third block to be processed when the input file is read, and it is required. Currently materials are defined using the MAT card which is described below. This card may appear an arbitrary number of times within the block and must appear at least once.

4.5.1 MAT Card

The MAT card is for specifying the materials that are used in the calculation. In general it requires an identifying number, a type description, and a list of cross section record identifiers. Different materials may use different cross section library data, if more than one library is specified in the XSEC block.

```
mat <id> <type> [<name>] [<temp> <tunit>] [<density> <dunit>] [-> <libname>] [
      (<unit>)] <delim> <xsname> [<num>] [!<comment>]
      [<xsname> <num> [...]] [!<comment>]
       .
       .
       .
      [!<comment>]

      [<xsname> <num> [...]] [!<comment>]
```

Essentially there are two parts to the input of the card, the part before the delimiter symbol (<`delim`>) and the part after. The part before must include the card name (`mat`) and the first 2 parameters <`id`> and <`type`> which have the following meaning:

- <`id`> - an identifying integer for this material to be used in the `PIN` card. This field is required.
- <`type`> - an integer enumeration indicating the type of this material. This field is required. The enumerations are listed in the table below:

Type Value	Is Fluid	Is Depletable	Has Resonance Data	Is Fuel
0	F	F	F	F
1	T	F	F	F
2	F	T	T	T
3	F	T	T	F
4	F	F	T	F
5	F	T	F	F

The rest of the parameters up to the delimiter are optional and they include the following:

- <`name`> - an optional string name for the material. Names must start with a letter and must not be any of the recognized unit strings (e.g. a material name cannot be at%). The default value is 'MAT <`id`>'. If specifying a name with space characters in it, it must be enclosed in double-quotes.
- <`temp`> <`tunit`> - the temperature of the material with the units. The card will accept degrees Celsius (denoted by 'C'), degrees Fahrenheit (denoted by 'F') and degrees Rankine (denoted by 'R'). The default unit is Kelvin (denoted by 'K'). The default value is 293.15 K. Absolute temperature must be greater than zero.
- <`density`> <`dunit`> - the material density with the units. Only default units are implemented. The default unit is g/cc. Default value is 1.0 g/cc. Values must be greater than zero.
- <`libname`> - the XS library file name (without the path and with the file extension) specified in a `XSLIB` card in the `XSEC` block, which should contain the XS records specified for this material. The default library is the first library listed in the `XSEC` block. Note that this name must be preceded by a `'->'` symbol.
- `(<unit>)` - the optional input for signifying whether the given `[<num>]` are number densities, atom, or weight percents. The default units are number densities, and are represented by not having the optional `(<unit>)` input. The other available input units are:
  - `(%at)` or `(at%)` for atom percent
  - `(%wt)` or `(wt%)` for weight percent
  - If either of these values are present in the input, the values will be used to convert the percents into number densities in the code.

Additionally:

- a specified density may appear before the specified temperature, or vice-versa

- each of these parameters may only be specified once per material.
- units must always appear with temperature or density.

An error is thrown if any of the rules for this card are violated.

Two delimiters are allowable. If the XS record is one macroscopic XS type, then the delimiter symbol is `'::'`. If more than one XS record constitutes this material, then the delimiter symbol is `'\'`. After the delimiter is a multi-line list of key-value pairs. The key and value must be given as a pair and each key in the list must be unique. There are several allowable formats for the key-value pair based on the units of the provided values, which is given as the next input after the `'\'` delimiter.

- `<xsname> [<num>]` - is a key-value pair consisting of a XS record identifier and number density pair. `<xsname>` is the XS record identifier which must exist in the XS library associated with this material, it is treated as a string. The `<num>` part of the pair is optional, but if it is present for one `<xsname>` entry, it must be present for all. It is treated as a double-precision value. It must not be used if the `'::'` delimiter is used. If the other delimiter (`'\'`) is used then the `<num>` field must be present. This field may occur with a variety of units. The list formats for the respective units that are allowable are:
- Number densities (default) have no unit tag.

```
... \ <xsname1> <num1>
      <xsname2> <num2> ...
```

- Atom percent have the `(%at)` unit tag before the `'\'` delimiter.

```
... (%at) \ <xsname1> <num1>
            <xsname2> <num2> ...
```

- Weight percent have the `(%wt)` unit tag before the `'\'` delimiter.

```
... (%wt) \ <xsname1> <num1>
            <xsname2> <num2> ...
```

In general, comments can be included at the end of a line, as long as they are the last item on the line. For the multi-line input, all blank lines or lines with just comments are allowed between continuing key-value pairs. The end of the card is not signaled until the next card or block is reached.

4.6 OPTION Block

The `OPTION` block is for specifying optional input parameters for which default values are not desired. This includes options for things like the maximum number of iterations, convergence control, solver types and meshing parameters. It is the last block to be processed when reading the input file. There are several cards which must only be used *once* within the block. They are described below. If used, the cards have the following format:

```
bound_cond           <west_bc> <north_bc> <east_bc> <south_bc> <top_bc> <bottom_bc>
iter_lim             <noutermax> <ngsweep> <ninner> [!<comment>]
conv_crit            <ecritk> <ecritphi> [!<comment>]
solver               <solvertype> <solverdim> [!<comment>]
ray                 [<spacing>] [<quad_name> [<order> [<order_theta>]]]] [!<comment>]
parallel             <nspace> <nangle> <nenergy> <nthreads> [<SpacePartitionType> [<SpacePartitionOpts>]] [!<comment>]
vis_edits            <T|F> [<T|F>] [!<comment>]
exp_table           [<LINEAR|POLAR|EXACT>]  [!<comment>]
validation           <T|F> <Delimeter Character> [!<comment>]
scatt_meth           <P0|TCP0|LTCP0|P1|P2|P3|P4|P5> [!<comment>]
cmfd                 <T|F|I|Y|A|S|M> [<1GSweep|MGNode|MGGroup> [<T|F> [<k_shift>]]] [!<comment>]
cmfd_shift_method   [<none|constant|adaptive|sdws-ips>] [!<comment>]
cmfd_nodal           <T|F> [<nCMFDperNodal>] [!<comment>]
```

```
bc_average            <average_xy> <average_xz> <average_yz> [!<comment>]
underrelax            <f_ur>
nodal                 <T|F> [<NEM|SANM|MOC|NEM-MG|SP1|SP3|SP5|Sn-0...Sn-3>] [<sn_type>] [<splitRTL>] [<nMom>]  [!<comment>]
power_edit           [<KAPPA-FISSION|FISSION>]
param                 <plist path> <datatype> <val> [!<comment>]
verbosity            [<INFO> <T|F>] [<WARN> <T|F>] [<DEBUG> <T|F>] [!<comment>]
ampx_out              <T|F> <file_name> <num_cart_1> <num_cart_2> ...
critboron             <T|F> [<target keff>] [<relaxation factor>] [!<comment>]
xenon                 <EQ|TR|NONE> [<relaxation factor>] [<T|F>] [!<comment>]
restart               <I|T|F|W|R|RW> [<filename>] [!<comment>]
axial_tl              <T|F> [<ISO|LIN|QUAD|DIFF|P1|DP0|EXP>] [<FLAT|LFLAT|FLUXXSTR>] [!<comment>]
simplifiedth          <T|F> [<ZR|SS>] [!<comment>]
cpm_init              <T|F> [!<comment>]
```

4.6.1 BOUND_COND Card

```
bound_cond <west_bc> <north_bc> <east_bc> <south_bc> <top_bc> <bottom_bc>
```

bound_cond	Fixed Array of Integers, Length 6	Optional
Units: N/A		
Applicable Value(s): (6×)0 — vacuum (default), 1 — reflective, 2 — periodic		
Limitation(s): None		
Description: This card is used for specifying the boundary conditions on each face of the simulation domain. The boundaries are listed in the following order: `<west_bc>` `<north_bc>` `<east_bc>` `<south_bc>` `<top_bc>` `<bottom_bc>`. Note that the boundary condition types can be freely mixed, but the default is that all boundaries are vacuum.		
Notes: None		

4.6.2 ITER_LIM Card

```
iter_lim <noutermax> <ngsweep> <ninner> [!<comment>]
```

iter_lim	Fixed Array of Positive Integers, Length 3	Optional
Units: N/A		
Applicable Value(s): {500 2 3} (default), {≥1 ≥1 ≥1}		
Limitation(s): None		
Description: This card is used to control the maximum number of iterations for the different levels of iteration. There are three values that must be specified: `<noutermax>` `<ngsweep>` `<ninner>` which govern the maximum number of outer iterations to perform (i.e., the amount of fission source iterations), the number of upscatter-sweeps between fission source updates, and the number of inner 1-group fixed source transport sweeps to perform between scattering source updates, respectively. The calculation will terminate once `<noutermax>` iterations have been performed.		
Notes: For 2-D/1-D problems, it is usually optimal for ngsweep and ninner to be set to 1. However, numerical instability is frequently an issue with these settings. For some problems, an extra upscatter sweep (ngsweep=2) will be sufficient for stabilization. If not, ninner=2 or 3 (with ngsweep=1) should be enough to stabilize the iteration.		

4.6.3 CONV_CRIT Card

```
conv_crit <ecritk> <ecritphi> [!<comment>]
```

`conv_crit`	Fixed Array of Positive Double-Precision Real Numbers, Length 2	Optional
Units: N/A		
Applicable Value(s): {1.0e-5 1.0e-4} (default), {≥1.0e-9 ≥1.0e-9}		
Limitation(s): None		
Description: This card is used to define the convergence criteria. The calculation will terminate once all the criteria have been met (or the maximum iterations has been reached). The two inputs in the following order are `<ecritk>` `<ecritphi>` where `<ecritk>` is the convergence criteria for k-eff measured as the absolute difference in k-eff between successive outer iterations and `<ecritphi>` is the convergence criteria for the scalar flux measured as the L2-norm of the difference of the multi-group scalar flux between successive outer iterations.		
Notes: None		

4.6.4 SOLVER Card

```
solver <solvertype> <solverdim> [!<comment>]
```

`solver`	Fixed Array of Integers, Length 2	Optional
Units: N/A		
Applicable Value(s): {0 2} (default), {0–3 2–3}		
Limitation(s): None		
Description: This card is for specifying the transport solver type and takes two arguments: `<solvertype>` `<solverdim>`. These inputs are identified as follows: `<solvertype>`: • 0 — Do all pre-processing as if MOC but do not solve the case. • 1 — MOC • 2 — CDP (What does this stand for?) • 3 — Discrete ordinates (S_N) `<solverdim>`: • 2 — 2-D • 3 — 3-D		
Notes: None		

4.6.5 RAY Card

```
ray [<spacing>] [<quad_name> [<order> [<order_theta>]]]] [!<comment>]
```

`ray`	Array of Mixed Types, Length 0–4	Optional
Units: {cm, Fixed String, Unitless, Unitless} (default)		
Applicable Value(s): {0.05 Chebyshev-Chebyshev 4 4} (default), See table in description for other options.		
Limitation(s): None		

continued on next page...

ray, continued...

Description: This card is for specifying the desired spacing between the characteristic rays and the angular quadrature and its order. This card takes up to four arguments. If the user wants to specify latter arguments, earlier arguments must be provided (even if they are the default values). These inputs are:

1. spacing is the spacing between characteristic rays in centimeters. Only one value is allowed (i.e., all rays have the same spacing. Valid entries are positive double-precision real numbers.
2. quad_name is the name of the angular quadrature to use. The below shows the acceptable combinations of quadratures and orders.
3. order is the order of the angular quadrature (or the azimuthal angle if the quadrature is a product quadrature).
4. order_theta is the order of the polar angle quadrature if a product quadrature is used, otherwise it is not needed.

Quadrature Name	Type	Order	Order Θ
CHEBYSHEV-CHEBYSHEV	Product	integers > 0	integers > 0
CHEBYSHEV-GAUSS	Product	integers > 0	integers > 0
CHEBYSHEV-BICKLEY	Product	integers > 0	1, 2, 3, or 4
CHEBYSHEV-YAMAMOTO	Product	integers > 0	1, 2, or 3
LEVEL-SYMMETRIC	General	even integers in [2,16]	N/A
QUADRUPLE-RANGE	Product	integers in [1,37]	integers in [1,18]

Notes: None

4.6.6 PARALLEL Card

```
parallel <nspace> <nangle> <nenergy> <nthreads> [<SpacePartitionType> [<
     SpacePartitionOpts>]] [!<comment>]
```

This card is used to specify the parallel environment at run time. The applicability of the specifications in this card depends on the problem input, machine, and current parallel domain decomposition algorithm.

- <nspace> - Integer number of spatial domains to divide the problem into. Default is 1.
- <nangle> - Integer number of angular domains to divide the problem into. Default is 1.
- <nenergy> - Currently not enabled. Integer number of energy domains to divide the problem into. Default is 1.
- <nthreads> - Integer number of threads to use per MPI process. How the threads are utilized depends on the solvers. Default is 1. If 0 is entered for the number of threads, then the value returned by the OpenMP routine OMP_GET_NUM_THREADS() will be used. Typically this corresponds to the OMP_NUM_THREADS environment variable when it is set.
- <SpacePartitionType> - String representing the type of partitioning to do in space. Can be one of: "FULLCORE", "ASSEMBLY", "BLOCK", or "FILE". Default is "FULLCORE".
- <SpacePartitionOpts> - Options based on previous entry. See note below.

The FULLCORE partitioning method performs binary spatial partitioning (BSP) based on the modular geometry of the full core. This is the default. It does not necessarily give the best load balance for a given number of processors for every grid, especially those with large prime factors. So if a better load balance is known for a given problem, one of the other methods may provide better parallel performance. It has no additional options for <SpacePartitionOpts>.

The ASSEMBLY partitioning method performs BSP first on the 2-D assembly grid of the core, and then BSP within each assembly. This partitioning guarantees that the problem may be run with <nspace> == n∗nasy. It has no additional options for <SpacePartitionOpts>.

The BLOCK partitioning method takes the modular geometry grid of the full core and factors out a sub-block of size xdim∗ydim∗zdim, where each of xdim, ydim, zdim are integer factors of the full core modular geometry grid along the respective axes. This partitioning guarantees that the problem may be run with <nspace> == nx∗ny∗nz/(xdim∗ydim∗zdim). BSP is also performed within each sub-block. The additional options for <SpacePartition-Opts> when "BLOCK" is chosen are:

- xdim - size of the sub-block on the x-axis in grid units. Must be greater than 0. Default is 1.
- ydim - size of the sub-block on the y-axis in grid units. Must be greater than 0. Default is 1.
- zdim - size of the sub-block on the z-axis in grid units. Must be greater than 0. Default is 1.

The FILE partitioning method takes the modular geometry grid of the full core and partitions it based on a text file in which the partitioning information is provided. The text file specifies loosely how to construct a tree by specifying subsequent block sizes to "grow" on each leaf node of the tree. To illustrate this say the file has the following:

```
2 2 2
2 2 2
```

This would create a block that is 2x2x2 then grow a tree based on BSP resulting in 8 leaf nodes. From these 8 leaf nodes a new 2x2x2 block would be created, this 2x2x2 block would grow a tree based on BSP at each of the 8 leaf nodes of the existing tree. So this example would produce an oct-tree with 2 levels defining a 4x4x4 grid with 64 nodes.

So each line of the file adds to the line above it and the product of the 3 numbers on the line means there are that many leaf nodes at that level (multiplied with the number of nodes from all previous levels).

The additional option for <SpacePartitionOpts> when "FILE" is chosen is the full path name of the file. The default is "./partition.txt" The path may be absolute or relative to the present working directory.

4.6.7 VIS_EDITS Card

```
vis_edits <T|F> [<T|F>] [!<comment>]
```

vis_edits	Fixed Array of Character Strings, Length 1 or 2	Optional
Units: N/A		
Applicable Value(s): {F F} (default), T where the first (required) entry determines whether or not to create the visualization files of the geometry / mesh descriptions. The second entry is optional to provide flat source region data on the full domain which includes material boundaries, mesh identification indices, and group-wise scalar flux.		
Limitation(s): None		
Description: This card is used to specify the type of visualization a outputs (edits). The visualization outputs are created in the form of the VTK legacy file format which is suitable for use with VisIt (https://wci.llnl.gov/simulation/computer-codes/visit/), or other suitable programs capable of reading the format.		
Notes: The FSR edits will be very large and may consume considerable time to generate the visualization files.		

4.6.8 EXP_TABLE Card

```
exp_table [<LINEAR|POLAR|EXACT>] [!<comment>]
```

`exp_table`	Fixed Character String	Optional
Units: N/A		
Applicable Value(s): LINEAR (default), POLAR, EXACT		
Limitation(s): None		
Description: This card controls the type of exponential table or function to be used when evaluating the exponentials in the MOC equations. The `LINEAR` option uses linear interpolation to evaluate the exponential function. The `POLAR` option uses linear interpolation as well but also stores all polar angles in the table. The `EXACT` option uses the Fortran intrisic `EXP(x)` function and no lookup table is used.		
Notes: None		

4.6.9 VALIDATION Card

```
validation <T|F> <Delimeter Character> [!<comment>]
```

`validation`	Fixed Array of Character Strings, Length 2	Optional
Units: N/A		
Applicable Value(s): {F S} (default), {T C}		
Limitation(s): None		
Description: This card is for developers and is not intended for use by general users and is furthermore marked for depracation. This card controls whether or not a secondary output will be generated. This output will be formatted for use in validation tests and is only ever needed for these tests. The input is a boolean with valid values of `T` or `F`, followed by an optional delimiter character. Acceptable values for this character are `S` (space) and `C` (comma).		
Notes: None		

4.6.10 SCATT_METH Card

```
scatt_meth <P0|TCP0|LTCP0|P1|P2|P3|P4|P5> [!<comment>]
```

`scatt_meth`	Fixed Character String	Optional
Units: N/A		
Applicable Value(s): TCP0 (default), P0, P1, P2, P3, P4, P5, LTCP0		
Limitation(s): None		
Description: This card is used to specify the scattering order to use in the MOC calculation. The Pn notation implies the number of Legendre scattering moments to use. The higher the value of n the more moments get used and the more accurate the solution. Using more moments can dramatically increase run times. `TCP0` (transport corrected P0 approximation) is the recommended option as it provides improved accuracy with the fastest run times. Not all cross section libraries include higher order scattering data, so the simulation will use the lowest of the specified scattering order and order of the data. Both the `TCP0` and `LTCP0` (limited transport corrected P0 approximation) options currently provide transport-corrected P0 solutions; however, `LTCP0` will truncate any cross-section values encountered above 1 MeV.		
Notes: None		

4.6.11 CMFD Card

```
cmfd  <T|F|I|Y|S|A|M> [<1GSweep|MGNode|MGGroup|1grbsor|mgrbsor> <T|F> [k_shift
```

```
    ]] [!<comment>]
```

`cmfd`	Array of Mixed Types, Length 1–4	Optional
Units: N/A		
Applicable Value(s): {T 1GSWEEP T 0.0} (default), This card controls options related to CMFD. The first argument specifies the type CMFD acceleration to perform: • `T` — Use conventional CMFD acceleration • `F` — Disable CMFD acceleration • `I` — Initializes solution with CMFD then performs no further CMFD • `A` — Specifies to use adCMFD • `S` — Same as T option • `M` — Uses multi-level CMFD • `Y` — Specifies an alternative CMFD formulation (used for research purposes) When CMFD is used, one may specify the CMFD solution algorithm. These options are • `1GSWEEP` — sets up and solves 1-group linear system and performs Gauss-Seidel iteration in energy • `MGNODE` — sets up the CMFD linear system for all space and groups with node major ordering, so matrix structure has group $\times$ group blocks. • `MGGROUP` — sets up the CMFD linear system for all space and groups with group major ordering, so matrix structure has nnode $\times$ nnode blocks. • `1grbsor` — sweeps through all of the energy groups one by one using Red-Black Successive Over-Relaxtion iteration. • `mgrbsor` — sets up a full multigroup CMFD matrix in node-major ordering (e.g. each node is a group-by-group block) and uses Red-Black Successive Over-Relaxtion iteration. If the `MGNODE` solution algorithm is used, then eigenvalue deflation (Wielandt shift) may also be used. To specify the use of eigenvalue deflation set the option to `T` and specify the shift parameter `k_shift` to use in the deflation.		
Limitation(s): None		
Description:		
Notes: None		

4.6.12 CMFD_SHIFT_METHOD Card

```
cmfd_shift_method  [<none|constant|adaptive|sdws-ips>] [!<comment>]
```

`cmfd_shift_method`	Fixed Character String	Optional
Units: N/A		

continued on next page...

`cmfd_shift_method`, continued...

Applicable Value(s): constant (default), none, adaptive, sdws-ips described as: • `none` — does not apply a shift to the CMFD system.   • `constant` — applies a constant, iteration-independent shift to the CMFD system. The constant is given by the reciprocal of the input to the `cmfd` card.   • `adaptive` — uses a traditional Wielandt shift method. The shift parameter is an iteration-dependent, spatially-constant quantity defined by: $$\lambda^{(n)}_{adaptive} = \max\left\{\lambda^{(n)} - c_1\left\lvert\lambda^{(n)} - \lambda^{(n-1)}\right\rvert - c_0\,, \lambda_{min}\right\}.$$ c_1, c_0, and λ_{min} have been hard-coded to 10, 0.02, and 0.3, respectively. Future implementations of the method may allow the user to specify these parameters.   • `sdws-ips` — uses a space- and iteration-dependent Wielandt shift based on the local infinite-medium eigenvalues, $\lambda_{adaptive}$, and the current guess of the eigenvalue: $$\lambda^{(n)}_{IPS}(\mathbf{x}) = \max\left\{\lambda^{(n)}_{adaptive}\,, \min\left\{\lambda_{\infty}(\mathbf{x}), \lambda^{(n)} - 0.01\right\}\right\}$$
Limitation(s): These methods can only be used with the mgnode CMFD solver or the MGRBSOR solver.
Description: This card is used to specify which Wielandt shift method will be used to accelerate the power iterations on the CMFD problem.
Notes: None

4.6.13 CMFD_NODAL Card

```
cmfd_nodal  <T|F> [<nCMFDperNodal>] [!<comment>]
```

`cmfd_nodal`	Fixed Character String, Positive Integer	Optional
Units: N/A		
Applicable Value(s): , This card controls options related to the CMFD Nodal extended type. The first parameter specifies the type CMFD acceleration to perform, @c T will use CMFD Nodal acceleration while @c F will disable CMFD Nodal acceleration but the CMFD card will still be used. The second parameter `nCMFDperNodal` specifies how many CMFD eigenvalue updates are performed between each Nodal update.		
Limitation(s): None		
Description:		
Notes: None		

4.6.14 BC_AVERAGE Card

```
bc_average  <average_xy> <average_xz> <average_yz> [!<comment>]
```

`bc_average`	Array of Positive/Negative? Precision? Numbers, Length 3	Optional
Units: N/A		
Applicable Value(s): {0 0 0} (default)		
Limitation(s): None		

continued on next page...

bc_average, continued...

Description: This card is for research applications and is not intended for use by the general user. It only applies to the CDP transport kernel. This card toggles whether or not to perform averaging of the angular flux at cell boundaries. Currently, the default is {0 0 0}, which means no average is done on all boundaries. Three integers indicate the number of rays(2-D) / planes(3-D) to be averaged on X-Y, X-Z and Y-Z faces.
Notes: None

4.6.15 UNDERRELAX Card

```
underrelax <f_ur> [!<comment>]
```

underrelax	Floating-point Real Number	Optional
Units: N/A		
Applicable Value(s): 1.0 (default), 0.0 $<$ `f_ur` $\leq$ 1.0		
Limitation(s): None		
Description: This card is used to specify a fixed under-relaxation coefficient for 2-D/1-D for all energy groups. The parameter `f_ur` is the underrelaxation coefficient.		
Notes: None		

4.6.16 NODAL Card

```
nodal <T|F> [<NEM|SANM|MOC|NEM-MG|SP1|SP3|SP5|HYSP3|Sn-0...Sn-3>] [<sn_type>] [<splitRTL>] [<nMom>] [!<comment>]
```

nodal	Array of Mixed Types, Length 5	Optional
Units: N/A		

continued on next page...

`nodal`, continued...

Applicable Value(s): {T NEM} (default), This card is used to specify the nodal method used for the 1-D solution in the 2-D/1-D method. Many of the options have only a research application (and are marked as such) and are not recommended for use by the general user. The options are:

- `NEM` — Use 4th order nodal expansion method formulation of diffusion equation.
- `SANM` — Use semi-analytic nodal expansion method formulation of diffusion equation.
- `SP1` — Use NEM formulation of Simplified P1 equations.
- `SP3` — Use NEM formulation of Simplified P3 equations.
- `SP5` — Use NEM formulation of Simplified P5 equations.
- `HYSP3` — Use Hybrid-Simplified Pn 3rd Order with NEM equations.
- `NEM-MG` — (RESEARCH) Use NEM nodal formulation with full-group response matrix.
- `Sn-0` — (RESEARCH) Use nodal formulation of Sn transport equations with a flat source.
- `Sn-1` — (RESEARCH) Use nodal formulation of Sn transport equations with a linear source.
- `Sn-2` — (RESEARCH) Use nodal formulation of Sn transport equations with a quadratic source.
- `Sn-3` — (RESEARCH) Use nodal formulation of Sn transport equations with a cubicsource.
- `MOC` — (RESEARCH) Use nodal formulation of MOC transport.

The remaining options apply to only the Sn kernels and are not recommended for use by the general user. `sn_type` specifies the type of Sn formulation to use with respect to source and transverse leakage treatment in the angular domain. The options are:

- AZIINT-ISOTROPIC — (RESEARCH)
- AZIINT-AZIINT — (RESEARCH)
- EXPLICIT-ISOTROPIC — (RESEARCH)
- EXPLICIT-AZIINT — (RESEARCH)
- EXPLICIT-EXPLICIT — (RESEARCH)
- EXPLICIT-MOMENT — (RESEARCH)
- MOMENT-MOMENT — (RESEARCH)

The `splitRTL` option is a logical for whether or not to split the radial transverse leakage source term to maintain positivity of the total source. This only applies to the Sn and SPn kernels.

The `nMom` option is used to specify the number of azimuthal moments to use in the `EXPLICIT-MOMENT` or `MOMENT-MOMENT` Sn solvers.

Limitation(s): None

Description:

Notes: None

4.6.17 POWER_EDIT Card

```
power_edit [<KAPPA-FISSION|FISSION>] [!<comment>]
```

`power_edit`	Fixed Character String	Optional
Units: N/A		
Applicable Value(s): KAPPA-FISSION (default), FISSION		
Limitation(s): None		
Description: This card is used to specify a cross section used for the "power" calculations `KAPPA-FISSION` is the standard power calculation whereas `FISSION` actually produces the normalized fission reaction rate distribution.		
Notes: None		

4.6.18 PARAM Card

```
param <plist path> <datatype> <val> [!<comment>]
```

`param`	Array of Mixed Types, Length 3 $\times$ number of parameters	Optional
Units: N/A		
Applicable Value(s): , This card is used to allow general access to the `mpactOptionBlockParams` parameter list and is a tool intended for developers. The three arguments are as follows: 1. `<plist path>` — the hierarchical variable path in the parameter list to specify. For example, `foo->value`. 2. `<datatype>` — specifies to the input processor what what type of value is being specified. Except for `STRING` types, each specifier can be followed with an `A` and the values afterward will be read into an array rather than a scalar variable. The types allowed are: • `SIK` — integer valued • `SLK` — 64-bit integer valued • `SNK` — 32-bit integer valued • `SRK` — real valued • `SSK` — single valued • `SDK` — double valued • `STRING` — string valued 3. `<val>` — the value to store in the specified path in the parameter list		
Limitation(s): None		
Description:		
Notes: None		

4.6.19 VERBOSITY Card

```
verbosity [<INFO> <T|F>] [<WARN> <T|F>] [<DEBUG> <T|F>] [!<comment>]
```

`verbosity`	Fixed Array of Character Strings, Length 6	Optional
Units: N/A		
Applicable Value(s): {INFO F WARN F DEBUG F} (default), Other combinations of T / F		
Limitation(s): None		
Description: This card toggles whether or not to report specified exceptions to the log file. Currently, the default is for only errors and fatal errors to be printed to the log. Information, warnings, and debug warnings are suppressed.		
Notes: None		

4.6.20 AMPX_OUT Card

This card is **CURRENTLY DISABLED**.

```
ampx_out <T|F> <file_name> <num_cart_1> <num_cart_2> ...
```

`ampx_out`	Array of Mixed Types	Optional
Units: N/A		
Applicable Value(s):		
Limitation(s): None		
Description: This card is used to specify an output file in which to store an AMPX working library with flux-weighted cross sections. The cross sections are generated upon convergence of the flux solution. This is an experimental feature and is currently deactivated.		
Notes: None		

4.6.21 CRITBORON Card

```
critboron <T|F> [<target keff>] [<relaxation factor>] [!<comment>]
```

`critboron`	Array of Mixed Types, Variable Length	Optional
Units: N/A		
Applicable Value(s): {F 1.0 1.0} (default), This card is used to set the parameters for the critical boron search. The various arguments are as follows: • `<T|F>` — Boolean flag to enable/disable critical boron search. • `target keff` — an optional input to set the target keff for the critical boron search and must be a positive floating-point value. • `relaxation factor` — an optional input to set the relaxation factor (a floating-point value) with a valid range of 0.0 to 1.0		
Limitation(s): None		
Description:		
Notes: None		

4.6.22 XENON Card

```
xenon <EQ|TR|NONE> [<relaxation factor>] [<T|F>] [!<comment>]
```

`xenon`	Array of Mixed Types, Variable Length	Optional
Units: N/A		

continued on next page...

xenon, continued...

Applicable Value(s): {NONE 1.0 F} (default), This card is used to specify the equilibrium xenon calculation option. The options are: • EQ — enables equilibrium xenon calculation • TR — enables transient xenon calculation (same as none) • NONE — disables equilibrium xenon calculation If equilibrium xenon is turned on an optional relaxation factor can be defined for this part of the iteration. In some problems the solution may oscillate if relaxation is not used. Typically these problems must be 3-D and involve depletion and use reflective boundary conditions. Valid ranges for the relaxation factor are on the interval [0.0,1.0]. The 3rd optional argument is to enable the equilibrium Sm calculation along with Xe.
Limitation(s): None
Description:
Notes: None

4.6.23 RESTART Card

```
restart <I|T|F|W|R|RW> [<filename>] [!<comment>]
```

restart	Array of Mixed Types, Length 1–2	Optional
Units: N/A		
Applicable Value(s): {I <CASEID>.mcp} (default), T, F, R, W, RW which correspond to: • I — specifies that a checkpoint file may be written through a user interrupt. • T — specifies that the case will be started from a checkpoint file. • F — disables initialization of the checkpoint file. • R — same as T. • W — specifies that a checkpoint file is to be written. • RW — same as T and R but after the checkpoint file is read it can be overwritten during the calculation. By default, the checkpoint file is named <CASEID>.mcp, but this can be overridden by providing an additional, optional, input on this card, as shown.		
Limitation(s): None		
Description: This card is used to control whether the calculation is restarted from a checkpoint file. The user can send the interrupt signal to MPACT after execution has begun by creating a file named “MPACT_CHECKPOINT_FILE” in the simulation's working directory. The existence of this file causes a checkpoint file to be written after every outer iteration. Likewise, the removal of “MPACT_CHECKPOINT_FILE” disables the writing of a checkpoint file.		
Notes: None		

4.6.24 AXIAL_TL Card

```
axial_tl <T|F> [<ISO|LIN|QUAD|DIFF|P1|DPO|EXP>] [<FLAT|LFLAT|FLUXXSTR>] [!<
     comment>]
```

`axial_tl`	Array of Mixed Types, Variable Length	Optional
Units: N/A		
Applicable Value(s): {ISO FLAT} (default), `<T|F>` is used to set axial transverse leakage (TL) splitting on/off. The following table describes the various ways to approximate the TL leakage source in angle: • `ISO` — Isotropic • `LIN` — Linear • `QUAD` — Quadratic • `DIFF` — Diffusion • `P1` — Simplified P1 • `DP0` — (RESEARCH) • `EXP` — (RESEARCH) The following table describes the various ways to approximate the TL leakage source in space: • `FLAT` — A flat or uniform spacial distribution. • `LFLAT` — Limited flat or uniform spatial distribution. A check is performed on the total / transport cross section. If the value is below the threshold, leakage will not be put into that region. This option is used to avoid leakage in the fuel-clad gap. It will then redistribute the leakage to the other regions in that pin. • `FLUXXSTR` — (RESEARCH)		
Limitation(s): None		
Description:		
Notes: None		

4.6.25 SIMPLIFIEDTH Card

```
simplifiedth <T|F> [<ZR|SS>] [!<comment>]
```

`simplifiedth`	Array of Mixed Types, Length 1–2	Optional
Units: N/A		
Applicable Value(s): F ZR (default), This card is used to control the execution of the simplified internal T/H capability. 1. `<T|F>` — used to set simplified TH on/off. 2. `<ZR|SS>` — used to set the thermal conductivity correlation for the clad. `ZR` is for zircaloy and `SS` is for stainless steel.		
Limitation(s): None		
Description:		
Notes: None		

4.6.26 CPM_INIT Card

```
cpm_init <T|F> [!<comment>]
```

`cpm_init`	Fixed Character String	Optional
Units: N/A		
Applicable Value(s): F (default), T		
Limitation(s): None		
Description: This card is for research applications and is not intended for use by the general user. It is used to control the flux initialization procedure (enabling/disabling CPM).		
Notes: None		

4.6.27 SUBPLANE Card

```
subplane  <subplane_max> [<subplane_target>] [!<comment>]
```

`subplane`	Fixed Array of Floating-point Real Numbers, Length 1 or 2	Optional
Units: N/A		
Applicable Value(s): , The input is positive real numbers. When CMFD is used, subplane may be used to refine the axial mesh for the CMFD system. If an axial solver is used, the axial mesh used by the solver will also be refined. The default is to disable subplane so that the CMFD axial mesh is the same as the MOC axial mesh. The first value is the maximum thickness of a CMFD plane. Any MOC plane which is thicker than this value will be divided into sub-planes for the CMFD system. If the `subplane_target` argument is present, it will be used as the "target" value to divide the plane. Otherwise, the `subplane_max` argument will be used as the "target" thickness.		
Limitation(s): Requires CMFD to be enabled to be used. This option has no effect on 2-D problems.		
Description:		
Notes: Under Development, do not use!		

4.7 STATE Block

The `STATE` block is for specifying input parameters that define the core's state conditions. This includes options for parameters like the core power and rated power. However, the block may be used more than once. If used, the cards have the following format:

```
RATED_POWER <rated_core_power> [!<comment>]
CORE_POWER <percent_rated_power(:)> [!<comment>]
RATED_FLOW <rated_core_flow> [!<comment>]
TINLET <tinlet> [!<comment>]
BORON <boron> [!<comment>]
PRESSURE <pressure> [!<comment>]
```

Each of these cards will be described in more detail in the following subsections.

4.7.1 RATED_POWER Card

```
RATED_POWER <rated_core_power> [!<comment>]
```

`rated_power`	Positive Floating-point Real Number	Optional
Units: MW (default)		

continued on next page...

rated_power, continued...

Applicable Value(s): 1.0E-6 (default)
Limitation(s): None
Description: This card is used to specify the whole core rated thermal power. It is assumed that if the rated power is not specified for the problem, that Zero Power conditions will be used.
Notes: None

4.7.2 CORE_POWER Card

```
CORE_POWER <percent_rated_power(:)> [!<comment>]
```

core_power	Arbitrary Length Array of Floating-point Real Numbers	Optional
Units: % (default)		
Applicable Value(s): 100.0 (default), 0.0–100.0		
Limitation(s): None		
Description: This card is used to specify the percentage of full core power. For depletion, multiple powers can be input. The last power will be used for the remaining steps. For non-depletion cases, only a single value is needed.		
Notes: None		

4.7.3 RATED_FLOW Card

```
RATED_FLOW <rated_core_flow> [!<comment>]
```

rated_flow	Floating-point Real Number	Optional
Units: kg/s (default)		
Applicable Value(s): 1.0 (default), $\geq$0.0		
Limitation(s): None		
Description: This card is used to specify the whole core rated coolant flow rate. This value is fixed during all states. There is no input card currently that allows the flow rate to be varied from state to state.		
Notes: None		

4.7.4 TINLET Card

```
TINLET <tinlet> [!<comment>]
```

tinlet	Floating-point Real Number	Optional
Units: °C (default)		
Applicable Value(s): 300.0 (default)		
Limitation(s): None		
Description: This card is used to specify the core inlet temperature.		
Notes: None		

4.7.5 BORON Card

This card is used to specify the core boron concentration. The units for this card are parts per million boron (ppmB). The default value is 0.0 ppmB.

```
BORON <boron> [!<comment>]
```

`boron`	Floating-point Real Number	Optional
Units: parts-per-million boron (ppm-B) (default)		
Applicable Value(s): 0.0 (default)		
Limitation(s): None		
Description: This card is used to specify the core soluble boron concentration. The boron is composed of both B-10 and B-11 in their natural abundancies at 19.9% B-10, and 80.1% B-11.		
Notes: None		

4.7.6 PRESSURE Card

```
PRESSURE <pressure> [!<comment>]
```

`pressure`	Floating-point Real Number	Optional
Units: Absolute Pounds per Square Inch (psia) (default)		
Applicable Value(s): 0.0 (default)		
Limitation(s): None		
Description: This card is used to specify the core pressure.		
Notes: None		

4.8 XSEC Block

The `XSEC` block is for specifying which files contain the cross section data to be used during the calculation. It is the second block that is read when processing the input file. This block must only appear once for a given input. It contains three cards `ADDPATH`, `XSLIB` and `XSSHIELDER`. `ADDPATH` and `XSLIB` can be used an arbitrary number of times within the block. If used, the cards have the following format:

```
xslib <xslibformat> <xslibname> [!<comments>]
addpath <path> [!<comment>]
xsshielder [<T|F> [<shieldingmethod> [<isubgrpset>] [<T|F>] ]] [!<comment>]
```

Each of these cards will be described in more detail in the following subsections.

4.8.1 XSLIB Card

```
xslib <xslibformat> <xslibname> [!<comments>]
```

`xslib`	Array of Fixed and Variable Character Strings, Length 2	Optional
Units: N/A		

continued on next page...

`xslib`, continued...

Applicable Value(s): USER (default), This card is used to specify the name of the cross section library file and its format. The `<xslibformat>` and `<xslibname>` arguments are string inputs. If the cross section library name contains spaces then double quotation marks must be used to enclose the string. The name must not include the path, but must include the file extension. For information about the library formats see the programmer's documentation. Allowable values for the `xslibformat` are: • `USER` — Default • `AMPX` — Requires SCALE license • `AMPX_CE` — Requires SCALE license (ESSM-X must use this card) • `AMPX_MASTER` — Not yet fully functional • `ORNL` — Requires SCALE license • `HELIOS` — Requires Helios license • `DEPLETION` — Not yet fully functional • `ORIGEN` — Requires SCALE license
Limitation(s): None
Description:
Notes: `AMPX_CE` is currently only used in ESSM-X for self-shielding correction. `<xslibname>` for this library format should be the indexing file of continous-energy AMPX library ,e.g., ce_v7_endf. As a convention, the path for individual isotopic cross section files should be provided at the firstline of the indexing file.

4.8.2 ADDPATH Card

```
addpath <path> [!<comment>]
```

`addpath`	Free-form Character String	Optional
Units: N/A		
Applicable Value(s):		
Limitation(s): None		

continued on next page...

`addpath`, continued...

Description: This card is used to append the search path for the cross section libraries specified with the `XSLIB` card of the `XSEC` block. This card is primarily for facilitating portability of inputs across different user environments and machines. There are four default directories that are searched prior to the paths specified in the `ADDPATH` cards. The additional paths are searched in the order in which they are defined and subdirectories are not searched. The default paths that are always searched are: 1. the present working directory (i.e., the directory from which the command was launched) 2. the directory containing the main input file 3. the directory containing the default output file 4. the directory containing the executable The `<path>` is a string input. An error will be produced when the file cannot be located. If the name contains spaces then double quotation marks must be used to enclose the string. The `<path>` can optionally end with a `SLASH` character, but if it is not present, one will be added. The `SLASH` characters in the `<path>` string will also be replaced by the correct `SLASH` character of the file system. For example, if the input file was written in a Windows environment and '/' were used in this card, and the file was later run on a linux environment which uses ', all of the '/' will be replaced automatically. Relative paths may be used and they are relative to the `PWD` environment variable.
Notes: None

The syntax for this card is shown below.

4.8.3 XSSHIELDER Card

```
xsshielder [<T|F> [<shieldingmethod> [<isubgrpset>] [<T|F>] ]] [!<comment>]
```

`xsshielder`	Array with Mixed Types, Variable Length	Optional
Units: N/A		
Applicable Value(s): {T SUBGROUP 4} (default), This card is used to control how MPACT performs the resonance self-shielding. Inputs are as follows: 1. `<T|F>` — the logical input that toggles the resonance self-shielding on (`T`) or off (`F`). 2. `<shieldingmethod>` — The name of the shielding method to use. The available options are `ESSM` or `SUBGROUP`. 3. `<isubgrpset>` — The set number (integer-valued) used if the shielding method is subgroup. 4. `<T|F>` — the logical input that toggles the ESSM-X on (`T`) or off (`F`) if the shielding method is ESSM.		
Limitation(s): None		
Description:		
Notes: If `ESSM` is used, the `<isubgrpset>` must be omitted and the second `<T|F>` may be provided by user to toggle ESSM-X on or off. The default option for ESSM-X is off.		