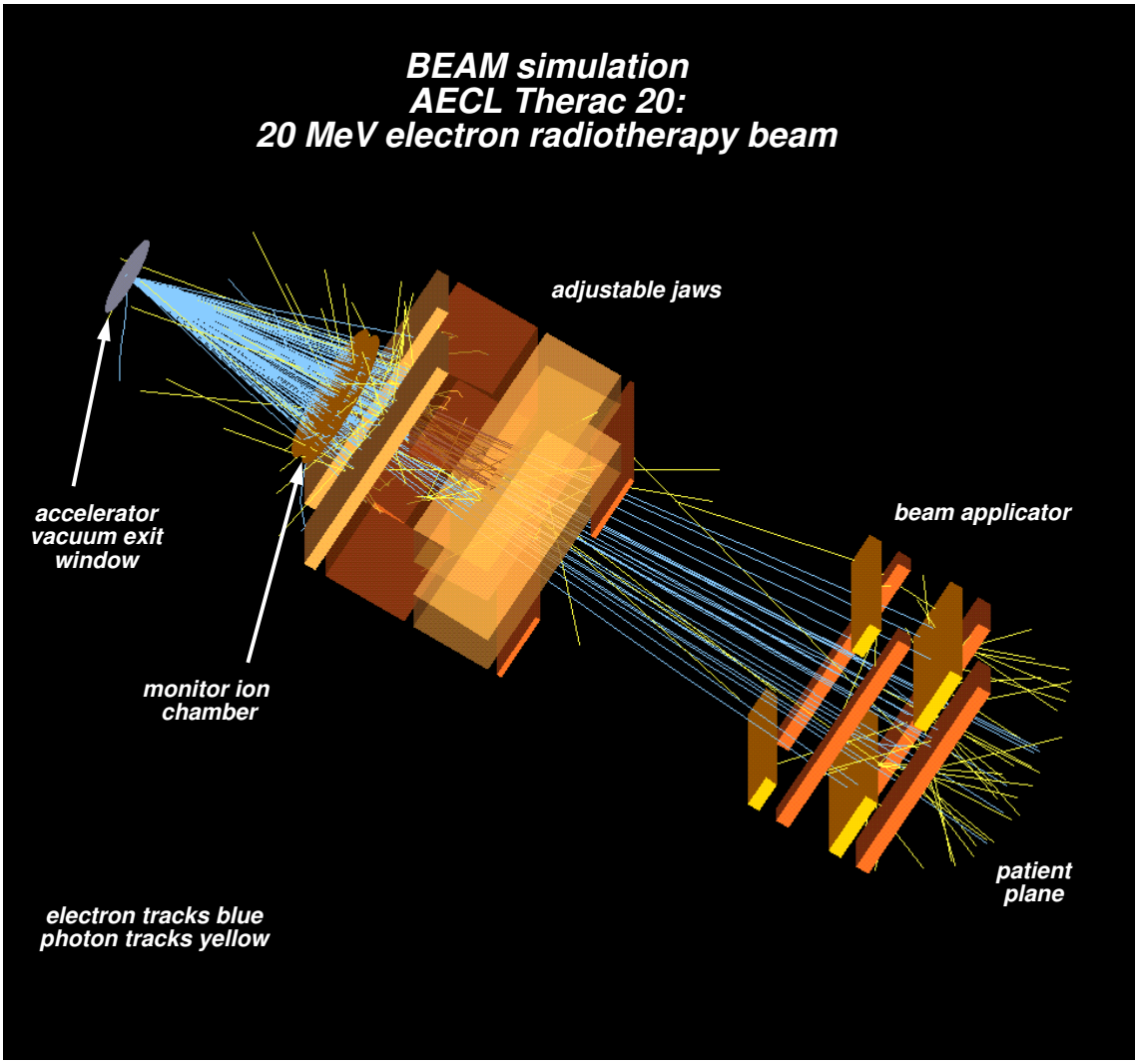
BEAMnrc Users Manual

D.W.O. Rogers, B. Walters, I. Kawrakow
Ionizing Radiation Standards
National Research Council of Canada
Ottawa, K1A OR6
Printed: April 15, 2021
NRCC Report PIRS-0509(A)revL



Abstract

BEAMnrc is a Monte Carlo simulation system (Med. Phys. **22**,1995,503 – 524) for modelling radiotherapy sources which was developed as part of the OMEGA project to develop 3-D treatment planning for radiotherapy (with the University of Wisconsin). BEAMnrc is built on the EGSnrc Code System[1] and, as of 2004[2], can run on Unix/Linux, Windows and macOS systems. The purpose of this manual is to be a reference/guide to someone using the BEAMnrc system.

This user's manual covers general BEAMnrc inputs and component module (CM) geometries and inputs. It discusses how to use the variance reduction techniques which are part of the system; specifically, range rejection, bremsstrahlung splitting, photon forcing and Russian Roulette. It also covers the directory structure within which the BEAMnrc system resides, the utility codes available (readphsp, addphsp, checkCM8 *etc.*), the installation procedure, the phase space file definition, and it has cross references to all related BEAMnrc documentation. Appendix A gives a specification for writing new component modules.

Contents

List of Figures

1 Overview of BEAMnrc

An extensive paper describing the BEAM system and its application has been published (ref [3]) and it is assumed that the user is familiar with that paper which is also available online at http://www.physics.carleton.ca/ drogers/pubs/papers/Ro95.pdf. We draw attention to the index in the present document since a great deal of effort has gone into making it and we find it a useful way to find information in this manual. This manual was originally written before the BEAM graphical user interface (GUI) was developed[4] and, thus, those sections describing input formats are somewhat redundant since the GUI makes the formatting of the input much easier. The EGSnrc system[5, 1, 6] was released in February 2000 and the BEAM code system was ported to using EGSnrc in 2001, used at the course in October 2001 and generally released in Feb 2002. In spring 2004 a version of the code was released to course participants with Directed Brem Splitting available. In fall 2004 this version of the code was made available generally with a port to the multi-platform version of EGSnrc (originally dubbed EGSnrcMP) which allows the code system to be run on Windows as well as Unix/Linux and MacOS. There is a 'new version' of this manual with each release of the code (usually with the annual course) and there have been five major revisions[7, 8, 9, 10, 11] with changes in authorship to represent those primarily responsible for the current version.

1.1 The Physics in BEAMnrc

BEAMnrc uses the EGSnrc Monte Carlo system of radiation transport. The physics in EGSnrc is described in detail in the EGSnrc manual [1]. The transport physics in EGSnrc is greatly improved over that in EGS4 (the basis for all BEAM versions up to and including BEAM00). Among the improvements are the ability to include relativistic spin effects in elastic scattering cross-sections for electrons, the ability to simulate atomic relaxations after Compton and photoelectric events and improved electron transport and multiple scattering algorithms. All these improvements increase the accuracy of EGSnrc over EGS4, especially at lower energies. The user has control over the extent to which the new physics is used in a simulation and also over some parameters required for the new physics. This means there are additional inputs for BEAMnrc that did not exist in previous versions of BEAM. On the other hand, the introduction of EGSnrc has eliminated the need for PRESTA and its associated inputs. Also, the EGSnrc inputs supersede some of the main BEAM inputs (such as `IDORAY` for controlling Rayleigh scattering, and `IFLUOR` for turning on K-shell X-ray fluorescence). See section 11 for a full description of the EGSnrc inputs and a more detailed description of the improved physics. The default EGSnrc parameters used by BEAMnrc were selected to balance efficiency and accuracy for typical accelerator simulations so, *eg.*, the EGSnrc default `EXACT` boundary crossing algorithm (BCA) is used instead of the faster `PRESTA-I` BCA because the latter has been shown to result in significant overestimates of dose when the CHAMBER component module is used as a phantom[12], however, atomic relaxations in BEAMnrc default to "off" (default in EGSnrc is "on") because their effect is insignificant at accelerator energies.

BEAMnrc retains compatibility with older BEAM input files that do not include EGSnrc inputs. In this case, the EGSnrc parameters simply revert to the default values used in BEAMnrc which differ from the EGSnrc defaults (see section 11).

1.2 Other documents available

There are several other documents which describe associated subjects.

BEAMnrc, DOSXYZnrc and BEAMDP GUI User's Manual: Describes how to install and use the graphical user interfaces for BEAMnrc and related codes[4].

BEAMDP as a General-Purpose Utility: User's manual for using BEAMDP to do simple analysis of phase space files - presenting spectra, fluence distributions, average energies, listings, angular distributions *etc.*, including the possibility of using the `LATCH` values in the phase space file[13].

QA for the BEAMnrc System: Component Modules, Variance Reduction Options and Source Routines. This 100 page document is for internal use at NRC but describes the extensive QA program carried out on component modules, variance reduction options and source routines. It also describes the automated system for ongoing QA[14].

DOSXYZnrc User's Manual: DOSXYZnrc is an associated code for doing dose distribution studies in a CT voxel phantom irradiated by a beam calculated using BEAMnrc[15].

Specifications for Component Modules for BEAMnrc: This document is for code developers who need to know what is going on in the code so they can write a new CM. It is attached as Appendix A(section A.1).

EGS_Windows_4.0 User's Manual: User's manual for an X-windows based display tool for EGS histories and BEAMnrc geometries[16].

The EGSnrc Code System Manual- PIRS-701: Complete manual describing the EGSnrc simulation system including the physics and inputs for Monte Carlo simulation parameters[1].

EGSnrcMP: the multi-platform environment for EGSnrc- PIRS-877: Manual describing the EGSnrcMP system[2].

History by history statistical estimators in the BEAM code system: Med Phys **29** (2002) 2745–2752: In-depth description of the method used to estimate uncertainty in BEAMnrc and DOSXYZnrc[17].

Large efficiency improvements in BEAMnrc using directional brem splitting: Med Phys **31** (2004) 2883 – 2898. Describes major improvement in efficiency for photon beam simulations[18].

1.3 Overview of the directory structure

The OMEGA/BEAM system has a well defined structure of directories. It can be thought of as having two or possibly three general parts. The first sub-system is generally referred to as `$OMEGA_HOME` and contains all the source code needed to run BEAMnrc and associated codes such as readphsp, BEAMDP *etc.*. In practice `$OMEGA_HOME` is the directory `$HEN_HOUSE/omega` (more about `$HEN_HOUSE` below). Figure 1 outlines the `$OMEGA_HOME`

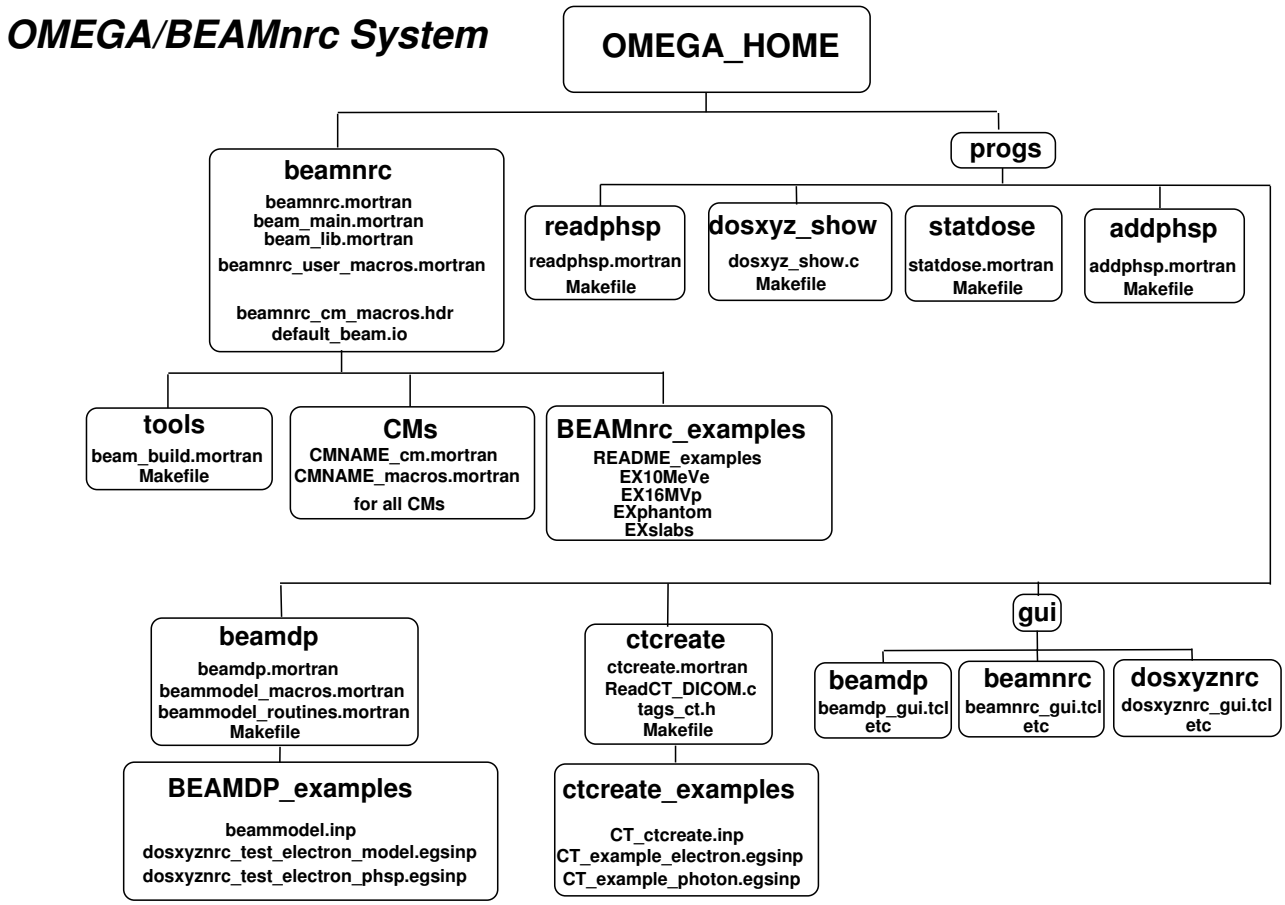


Figure 1: Main components of the `$OMEGA_HOME` directory. `$OMEGA_HOME` is a subdirectory, called `omega`, of `$HEN_HOUSE` which is given in fig 2

subsystem. Note that this part of the system contains no execute modules related to beam simulation. These all reside on the user's `$EGS_HOME` area described below.

`$OMEGA_HOME` resides (as directory `omega`) within the `$HEN_HOUSE`, which contains the EGSnrc system. This is shown in more detail in fig 2. By default, `$HEN_HOUSE` is sudirectory `EGSnrc/HEN_HOUSE` off the directory into which you downloaded/cloned the EGSnrc distribution. Originally, EGSnrc was a Unix-based script system developed primarily by Alex

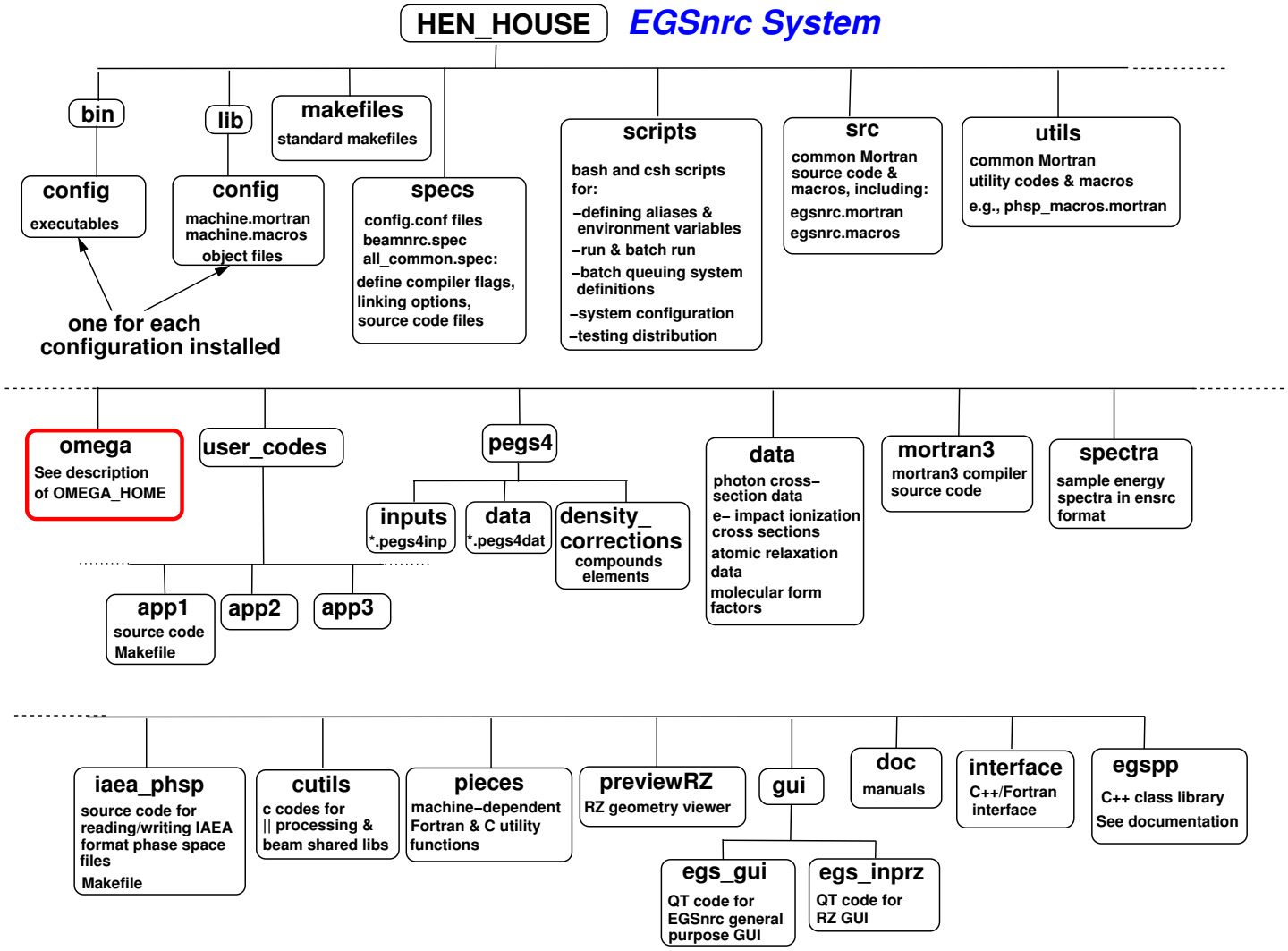


Figure 2: The structure of the EGSnrc system. Note that the `$OMEGA_HOME` subsystem shown in fig 1 is included in this structure.

Bielajew and Dave Rogers at NRC. With the advent of the multi-platform EGSnrc system (EGSnrcMP[2]) and the requirement for compiling on Windows-based systems in addition to Unix-based systems, we have largely eliminated the scripts and, in the case of compilation, replaced them with the GNU `make` utility.

The final component of the EGSnrc directory structure is the user's area which is shown in fig 3. This is known as `$EGS_HOME` and is subdirectory `EGSnrc/egs_home` off the directory in which you downloaded/cloned the EGSnrc installation. One of the main complications is

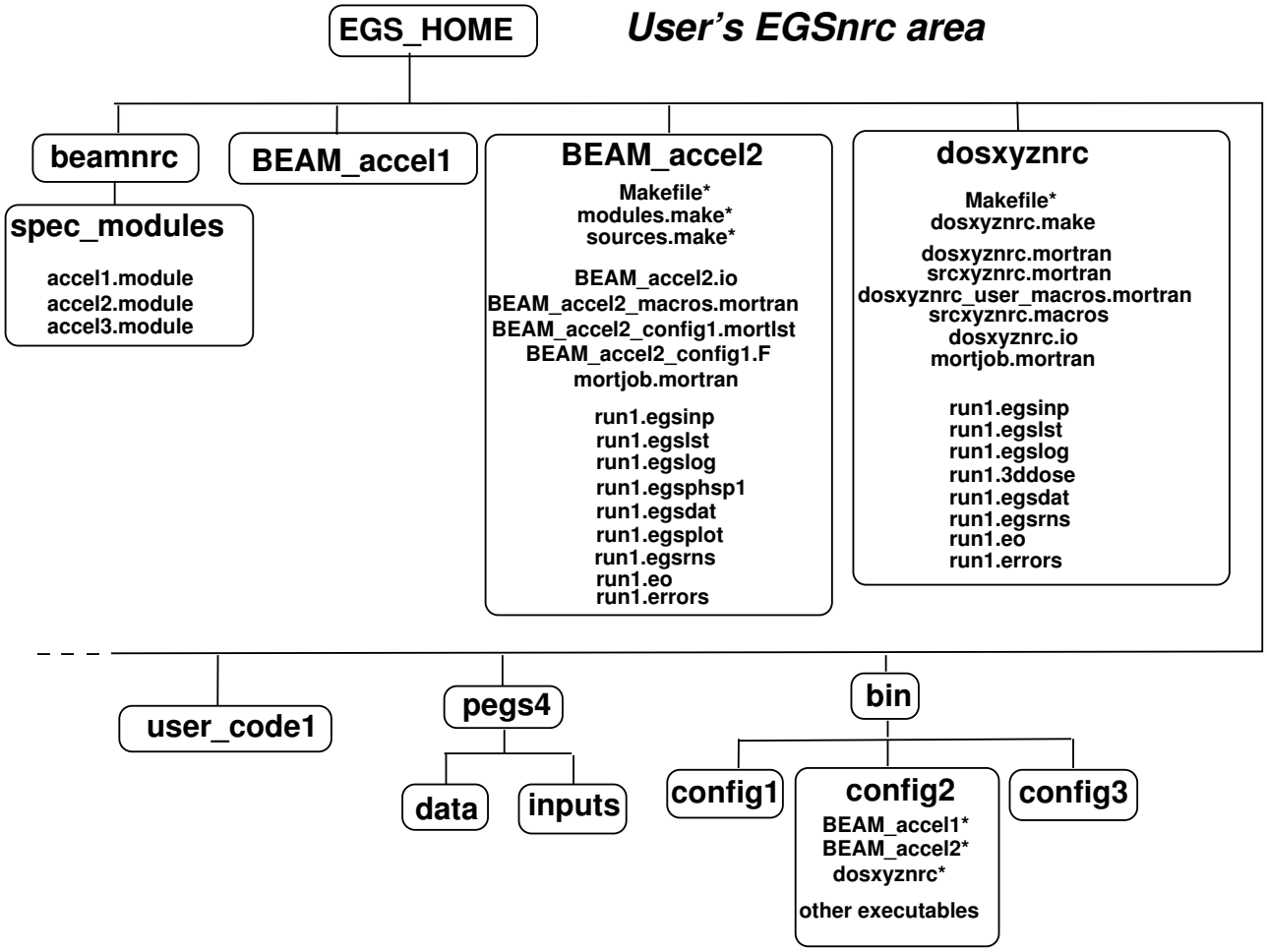


Figure 3: The user's `$EGS_HOME` area. A $*$ indicates an executable file. In the example shown, the user has 3 accelerator models and the disk system is being used with three different configurations, *eg.*, gcc, pgf77 and Windows. The complete directory contents are only shown for accel2 and config2. Other EGSnrc user codes (eg DOSXYZnrc) also reside in the `$EGS_HOME` area.

that the EGSnrc system is set up to use one disk system to support multiple configurations and associated compilers. Thus, all execute modules and various compiler options *etc.* must be handled separately for each configuration. Within the `bin` area, the modules within each subdirectory apply only to the configuration shown.

1.4 Overview of running BEAMnrc

Figure 4 presents a schematic of the overall steps required to do an accelerator simulation. At the specify accelerator and build accelerator steps, the user is instructing the system how to pull together the source code and make an execute module. At this stage, only a broad

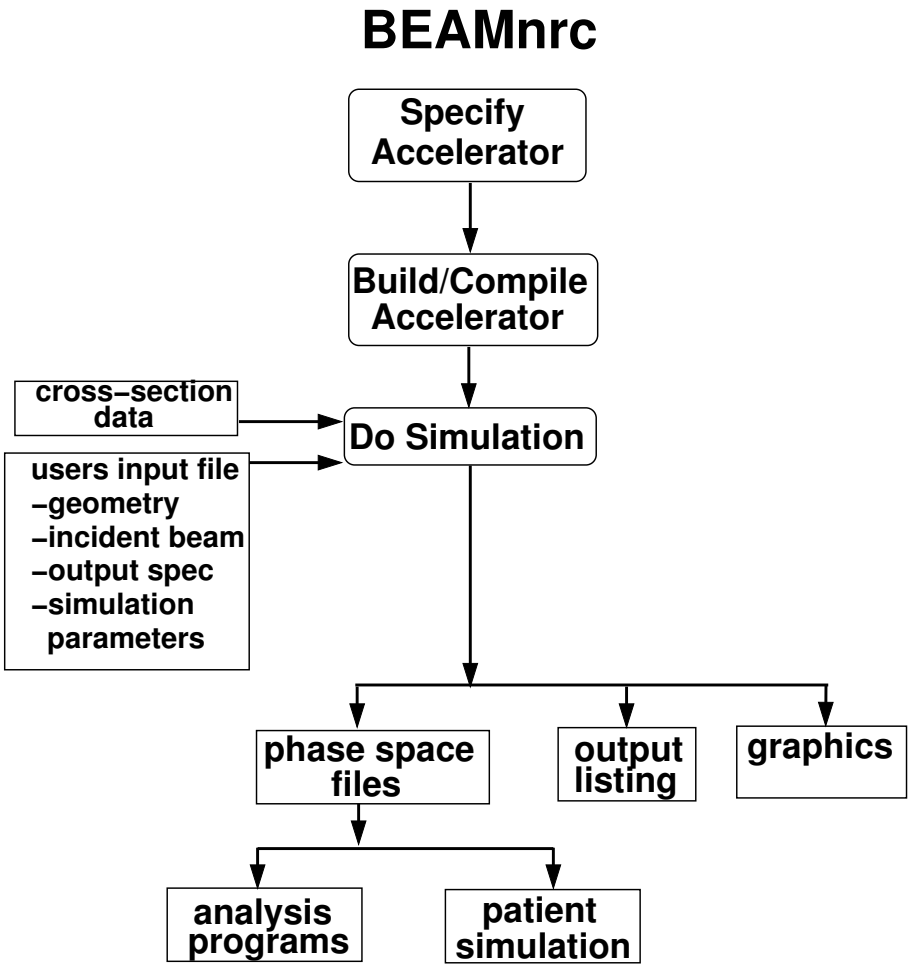


Figure 4: The steps involved in using the BEAMnrc system, from ref [3].

class of accelerator is defined (*eg.* whether the flattening filter is before or after the primary collimator). During the execution stage, the program reads in a large quantity of data related to photon and electron cross sections for the specific materials in this accelerator model. These data are generated by a code, called PEGS4 (which is included with the EGSnrc system see section 16) or read from data files such as `msnew.data`, `spinms.data`, etc, which are contained in the `$HEN_HOUSE/data` subdirectory. Also the user creates an input file which specifies all the details about the particular accelerator (*eg.*, there are 4

scattering foils located at specified distances from the vacuum exit window, made of certain materials and of particular thickness). In addition, the user must specify all the parameters controlling the radiation transport modelling and must also select and control the various variance reduction techniques being used.

The final stage of the simulation is the analysis of the outputs which consist of raw phase space files (which may be on the order of GBytes), an output listing and optionally a 3D graphics file to be displayed by `EGS_Windows`[16].

2 Building/compiling/running BEAMnrc

See section 17 at the end of this manual for instructions for installing BEAMnrc and the rest of the OMEGA/BEAM codes.

If you have installed BEAMnrc on a Linux/Unix system then the first step towards running the code is to go into your `$HOME/.cshrc` or `$HOME/.bashrc` (depending on whether you're running in a C-shell or Bourne-shell, respectively) file and, at the end of the file, add the following statements:

```
setenv EGS_HOME /full directory path to $EGS_HOME/
(or export EGS_HOME=/full directory path to $EGS_HOME/ .bashrc)
setenv EGS_CONFIG /full directory path to $HEN_HOUSE/specs/config.conf/
(or export EGS_CONFIG=/full directory path to $HEN_HOUSE/specs/config.conf/
                                                          for .bashrc)
source /full directory path to $HEN_HOUSE/scripts/egsnrc_cshrc_additions
(or source /full directory path to $HEN_HOUSE/scripts/egsnrc_bashrc_additions
                                                          for .bashrc)
```

where `config` is the name of the particular configuration (eg `gcc`, `pgf77`) that you have installed BEAMnrc on. Once you have added these statements you must source the `.cshrc` (or `.bashrc`) file or start a new terminal window to bring them into effect.

Instructions for adding these statements are given explicitly at the end of the EGSnrc/BEAMnrc installation (see section 17).

No such additions are required when installing BEAMnrc on a Windows system.

The following subsections define the processes of specifying, building and compiling an accelerator. In general, these are carried out via the BEAM GUI. However, command-line options are also given.

2.1 "Specifying" Accelerators

Before compiling and running a BEAM accelerator simulation, you must "specify" which component modules are to be used and in what order. The available `CMs` are discussed in detail in section 15. Each CM can be used in a wide variety of applications and user should not restrict themselves by the names. For example, the JAWS CM is well suited to simulating a wedge. It is useful to select identifiers (8 characters or less) which are physically

meaningful (eg xit_win, scatfoil, jaws *etc.*) since these will appear throughout the code and listing files and thereby help the user understand what is going on in various applications. The user may use the same CM as often as needed, as long as the identifiers used are unique. One restriction is that the identifier name should not start with the word "exit" since the `MORTRAN` compiler gets confused with this.

Accelerators are stored in a very simple format in `myaccel.module` files that reside in the user's `$EGS_HOME/beamnrc/spec_modules` subdirectory (see Figure 3 above). When built and compiled, the executable code will have the name `BEAM_myaccel`, so it is useful to have `myaccel` indicate the machine you are modeling.

The `myaccel.module` file can created from scratch using the BEAM GUI (just select "Specify a new accelerator" from the "File" menu), or you can copy an existing `.module` file (eg one of the examples included with the distribution) into `myaccel.module` and modify the latter for your own purposes using an editor.

Any number of specified accelerators can coexist at the same time.

2.2 Building an Accelerator: The `beam_build` Code

Once the accelerator has been specified, it must be "built", which corresponds to concatenating all of the relevant source code for the `CMs` and editing it to avoid duplicate variable names (this is discussed below for those who need details). The resulting files are on the user's area and are `$EGS_HOME/BEAM_myaccel/BEAM_myaccel_cm.mortran` and `$EGS_HOME/BEAM_myaccel/BEAM_myaccel_macros.mortran`. Note that the name of the subdirectory and the file names themselves are just the name of the specification module with the prefix `BEAM_`.

An accelerator must be re-built any time one of its component source files (*e.g.*, the code for one of its CMs, or `beamnrc.mortran`) is modified.

An accelerator is built using the MORTRAN code `beam_build.mortran`. The command for running this code to build the accelerator specified by `myaccel.module` is:

```
beam_build.exe myaccel
```

`beam_build.exe` creates the `BEAM_myaccel_cm.mortran` and `BEAM_myaccel_macros.mortran` files and puts them in the `BEAM_myaccel` subdirectory (creating the subdirectory on the way if it does not already exist).

In addition, `beam_build` creates the files `Makefile`, `modules.make` and `sources.make` and puts them in the `BEAM_myaccel` subdirectory. These files are used for compiling beam. More details are given in section 2.12.3, on page 30, below.

If the `beam_build` utility should be compiled when configuring your system. However, if it has not, then before typing the above command line, go into `$HEN_HOUSE/omega/beamnrc/tools/beam_buil` and type `make`. This will put a copy of the `beam_build.exe` executable in the appropriate subdirectory of `$HEN_HOUSE/bin`.

2.3 Compiling an Accelerator using `make`

Once specified and built, the accelerator must be compiled. This requires using the `MORTRAN` compiler (which is supplied as part of the EGSnrc system) followed by the `FORTRAN` compiler.

Compilation of BEAMnrc (and all other EGSnrc codes) uses the GNU `make` utility.

To compile an accelerator, go into the `BEAM_myaccel` subdirectory and type:

```
make [options]
```

The options for `make` are:

```
make              Compile the accelerator executable with default optimization
make opt          turned on.  Default optimization is level 2 (-O2).

make noopt        Compile the accelerator executable with optimization
                  turned off.

make debug        Compile an accelerator executable for debugging.

make fortran      Do mortran compilation only, leaving behind the fortran
                  file BEAM\_myaccel\_config.F.

make clean        Remove the fortran file, mortjob file, mortlst file
                  and the various executables.
```

Upon typing `make` you will see output to the screen indicating files that are being concatenated together to create the final MORTRAN code that is then compiled.

After compiling an accelerator, the following files will be left behind:

In the accelerator directory:

`mortjob.mortran` The concatenated MORTRAN file which is MORTRAN compiled.

`BEAM_myaccel_config.mortlst` Listing from the MORTRAN compiler. Contains the concatenated MORTRAN code, formatted with levels of loops and `IF` blocks indicated. Output at the end of the file indicates if there were any MORTRAN errors and, if so, how many and of what type.

`BEAM_myaccel_config.F` Fortran output from the MORTRAN compiler. This is the f77 code that is ultimately Fortran compiled. The use of capital "F" indicates that some C routines are to be linked with the Fortran code at compile time. In the case of BEAM, these C routines pertain to parallel job submissions.

In the `$EGS_HOME/bin/config` directory:

`BEAM_myaccel*` The executable code.

If none of the MORTRAN source coding or macros have changed since the last compilation and the file `BEAM_myaccel_config.F` exisits in the accelerator directory and there is an executable in your `$EGS_HOME/bin/config` directory, then typing `make` will not recompile the accelerator.

If you want to include the beam characterization source in the accelerator, see the instructions in section 4.16 page 86.

2.3.1 Compiling an Accelerator as a Shared Library

DOSXYZnrc[15] and other EGSnrc user codes[19] allow the user to use a full BEAM simulation as a particle source (instead of a stored phase space file). In order to use this source, the BEAM accelerator code must first be compiled as a shared library.

Once an accelerator has been built, it can be compiled as a shared library by going into `$EGS_HOME/BEAM_myaccel` and typing:

```
make library
```

The codes concatenated to create the `mortjob.mortran` file are the same as those used for a standard BEAM compilation with the exception that `$OMEGA_HOME/beamnrc/beam_lib.mortran` is used in place of `$OMEGA_HOME/beamnrc/beam_main.mortran` (see section 2.12.4, page 32). The library compilation also leaves behind `libBEAM_myaccel_config.mortlst` (output from MORTRAN compiler) and `libBEAM_myaccel_config.F` (Fortran source code), where `config` is the name of your configuration.

The BEAM shared library is archived as the file `libBEAM_myaccel.so` (for unix/Linux machines) or `BEAM_myaccel.dll` (for Windows machines) in directory `$EGS_HOME/bin/config`.

When using the BEAM shared library as a source, you must also supply a working input file and the pegs data for the BEAM simulation. The input file must exist in your `$EGS_HOME/BEAM_myaccel` directory and be set up to write a phase space file at a scoring plane. See the DOSXYZnrc Manual[15] and the manual for EGSnrc user codes[19] for more details about this source.

2.4 Compiling an Accelerator using `mf` (Unix/Linux specific)

To preserve compatibility with old usage, an accelerator can also be compiled using `mf`, which is aliased to the Unix script `$HEN_HOUSE/scripts/compile_user_code`. To compile using `mf`, go into the `BEAM_myaccel` directory and type:

```
m[f] BEAM_myaccel [a] [opt|noopt|debug]
```

The options for `mf` are: `mf` => Mortran and Fortran compile and then link
`m` => Mortran compile and create the Fortran file
`opt` => use optimization
`noopt` => use no optimization
`debug` => create executable ready for a debug run

The parameter "a" is not used and is only present for compatibility with the previous version of `mf`.

Once the `mf` command is issued, the `compile_user_code` script then calls `make` with the appropriate options.

2.5 Building/Compiling with the BEAM GUI

Building and compiling are done in a single step when using the BEAM GUI. Once an accelerator `.module` file has been specified or read into the GUI, then it can be built/compiled by selecting "Compile" from the "Execute" menu. This opens up a window in which you can choose any of the `make` options outlined in section 2.3 above. Then press the "BUILD & COMPILE" to run `beam_build` followed by `make`. If there is no `beam_build.exe*` executable in the appropriate `$EGS_HOME/bin/config` directory, then the GUI will automatically compile `beam_build` for your configuration.

2.6 Internal Documentation and Input Description

Since a given accelerator model can have an arbitrary structure, defining the input file for a given accelerator must be done after the accelerator is specified. Extensive descriptions of the main BEAM inputs and individual CM inputs are given at the top of the `beamnrc.mortran` and `CM_cm.mortran` codes, however the BEAMnrc graphical user interface (GUI) provides extensive descriptions of each input variable in addition to being able to display graphical representations of component module and accelerator geometries as input by the user. Thus, the GUI is the preferred method for generating input files. For more information on the BEAMnrc gui, see the BEAMnrc, DOSXYZnrc and BEAMDP GUI User's Manual[20].

2.7 Running an Accelerator Simulation

Once the accelerator model is compiled, the user may run a simulation. To run a simulation from the command line, type:

```
BEAM_myaccel -i inputfile -p pegsdata
```

where `inputfile` is the name of the BEAM inputfile (with no `.egsinp` extension) and `pegsdata` is the name of the PEGS4 material data file (with no `.pegs4dat` extension). These two files are discussed in more detail below.

To preserve compatibility with old usage, accelerators can also be run with the "`ex`" command:

```
ex BEAM_myaccel inputfile pegsdata
```

"`ex`" is aliased to the Unix script `$HEN_HOUSE/scripts/run_user_code`. This script uses the non-script command line shown above to actually run the accelerator, but first it checks for the existence of the relevant directories (`$EGS_HOME` and the `BEAM_myaccel` subdirectory). It

also checks for the existence of the `BEAM_myaccel*` executable and compatibility between the `config.conf` file you are using and the actual configuration that you are running on.

The users input file, *i.e.* `inputfile.egsinp`, defining the geometry of the accelerator and the BEAM simulation parameters, must exist in `$EGS_HOME/BEAM_myaccel`. The best way for a new user to begin creating a new input file is to use the BEAM GUI[20]. From the GUI, you can either begin a new input file from scratch or read in an existing input file (for the same accelerator) and modify the parameters accordingly. More experienced users may find it faster to use an editor to modify an existing input file.

Note that the input file extension is `.egsinp`. Theoretically, BEAMnrc can run a BEAM00 `.egs4inp` input file, however, depending on the parameters, there may be new inputs that will cause an error on reading the older file. It is safer to read the old `.egs4inp` file into the GUI and resave it. The GUI will then write to the file all the necessary inputs for the current version of BEAM. Note that `.egs4inp` files do not contain any EGSnrc transport parameter inputs, so standard BEAMnrc defaults for these parameters will be used when older files are read into the GUI and/or used in a BEAM simulation.

Much of the rest of this document is devoted to discussing the meaning of all the options available through the input file.

In order to run an accelerator, the code must have access to electron stopping power and interaction cross section data. This can be supplied either vi a separate `.peg4dat` file or, if running in "pegsless" mode, it can be calculated on the fly based on media inputs in the `.egsinp` file. Note that all media used in a simulation must be defined, either in the `.pegs4dat` file or in the `media definition` section of the `.egsinp` file.

There is more information on PEGS4 files in section 16 but basically the files `700icru.pegs4dat` and `521icru.pegs4dat` contain data for a large number of commonly used materials. The numbers in the names correspond to AE=521 and AE=700, *i.e.* thresholds for secondary electron production of 10 and 189 keV kinetic energy respectively. The data sets go up to 55 MeV in both cases. These data sets are on the distribution on area `$HEN_HOUSE/pegs4/data`. If the user wishes to make their own data files using the program PEGS4, these files can be put either on that area, or on the user's pegs4 area, *viz.* `$EGS_HOME/pegs4/data`.

For more information on pegsless runs, the user is referred to section 16.3 in this manual.

2.7.1 Batch Runs

Thus far, we have described interactive BEAM runs, in which the BEAM input and job execution information is echoed to the screen and where the window in which the job is running cannot be used for anything else while the job is executing. However, if you are running on a Unix/Linux system, then it is often desirable to run simulations in batch mode. For example, batch job submission can make use of a network queuing system (eg PBS or NQS) and is required for parallel jobs (see section 13 in this manual for more information on parallel jobs).

Submitting a batch job uses the `exb` command, which is aliased to the Unix script `$HEN_HOUSE/scripts/run_user_code_batch`. The basic syntax of the `exb` command is:

```
exb BEAM_myaccel inputfile pegsdata [short|medium|long] [batch=batch_system] [p=N]
```

The input `[short|medium|long]` determines the name of the queue to be used (the default is `long`). With our naming system at the NRC, the `short` queue has the lowest maximum CPU time but the highest priority, while `long` has an unlimited CPU time with lowest priority.

The `batch=batch_system` input determines the name of the queuing system to use. The `run_user_code_batch` script sources the file `$HEN_HOUSE/scripts/batch_options.batch_system`, which defines the batch submission commands for the particular queuing system chosen and may redefine the names of the queues available on that system (this means that the queue names `short|medium|long` are not necessarily general). Currently, `batch_system` can be set to:

- `at` (the standard Unix batch command)
- `pbs` (for the PBS queuing system)
- `nqs` (for the NQS queuing system)
- `pbsdsh` (PBS in distributed shell mode)
- `keg` (SUN SGE scheduler)

This means that the files `batch_options.at`, `batch_options.pbs`, `batch_options.nqs`, `batch_options.pbsdsh`, and `batch_options.keg` are included with the EGSnrc distribution. However, the batch submission commands in these files are for our system at the NRC and you may have to make some changes for your system. The default value for `batch_system` is `at`, unless you have the environment variable `$EGS_BATCH_SYSTEM` set to something else.

Finally, the `p=N` input is used if you want to split the simulation into `N` parallel jobs. For more information on parallel runs, see section 13, page 133.

In addition to the files which are output from an interactive accelerator simulation, a batch run will also output a `inputfile.egslog` and file and a `inputfile.eo` file. The `.egslog` file contains all the output that would be echoed to the screen during an interactive run. **IT IS IMPORTANT TO LOOK AT THE `.egslog` FILE TO SEE IF ANY INPUT OR RUN TIME ERRORS OCCURRED**. The `.eo` file contains messages from the queuing system. If, for some reason, a job did not get submitted properly, then this file may contain clues to the problem. More information on BEAM output files is given in section 2.10, page 24 below.

2.8 Running BEAMnrc using the GUI

It is possible to use the BEAM GUI to submit interactive, and if your system is set to handle batch jobs, batch and parallel runs. To do this, select "Run" from the "Execute" menu. This opens a running window which will give you the option to switch from an interactive to a batch (and parallel) run, and will allow you to select the queue you wish to run on. If you have not set the environment variable `$EGS_BATCH_SYSTEM`, then the GUI automatically

submits to a PBS network queuing system (ie sets `batch_system=pbs`). When you press the "Execute" button, the GUI then assembles and executes the appropriate command line as described in the sections above. Output which would normally appear on the screen during an interactive run is now displayed in the GUI running window.

2.9 A Note on Temporary Working Directories

Before a BEAM run (batch or interactive), a temporary working directory, `$EGS_HOME/BEAM_myaccel/egsrun_pid_inputfile_hostname` is created, where `pid` is the process ID number, `inputfile` is the name of the input file (no `.egsinp` extension), and `hostname` is the name of the computer the job is running on. All output files from a run (more about output files in the next section) are written to this directory with the exception of phase space (`.egsphsp#`) files, which are written directly to the `$EGS_HOME/BEAM_myaccel` directory. Once the run is finished, all output files are moved from `$EGS_HOME/BEAM_myaccel/egsrun_pid_inputfile_hostname` into `$EGS_HOME/BEAM_myaccel` directory and the temporary working directory is removed. The reason phase space files are written directly to the accelerator directory is that they often end up being large, which makes moving them from one directory to another time consuming. If, for some reason, the simulation terminates prematurely, the temporary working directory, containing its output files, will be left behind.

2.10 BEAMnrc Output Files

Each of the BEAM output files that will be found in your `$EGS_HOME/BEAM_myaccel` directory at the end of a run is described briefly below.

.egslst This may be considered the "main output file" because it contains all of the dose and fluence results of the simulation. The file is broken up into different sections:

Error and Warning Messages Most error and warning messages that appear on screen (or in the log file) during input are repeated at the top of the listing file.

Summary of the Main BEAMnrc Inputs This section summaries those inputs that are not CM-specific, such as the number of histories to run, photon forcing inputs, *etc.*. The section also includes some characteristics of a phase space source (if used), such as the total number of particles in the source, the maximum kinetic energy of the source, *etc.*. If range rejection was chosen, this section also contains a table of maximum electron range versus electron energy for the various materials to be used in the simulation.

Material Data Shows relevant data (density, `AE`, `AP`, *etc.*) for all materials used in the simulation as read from the `.pegs4dat` file.

Source Parameters This is a summary of parameters such as the type of source, the Z location of the face on which the source is incident, *etc.* For a phase space source, this section repeats the source characteristics found in the first section of the file.

Region and Range Rejection Summary This is a very useful summary of all of the regions in the simulation. Regions are identified by both their absolute and local numbers. The absolute region number identifies the region within the entire simulation geometry, the local region number identifies the region within its own component module. The region summary indicates what component module a region is in, the dose zone number and bit number the user has assigned to a region (called the "bit region") and the material that the user has specified the region to be made of. If range rejection is turned on, this summary also contains the value of `ECUTRR` (the minimum energy an electron must possess in a region in order for it to reach the bottom of the simulation geometry) the residual range and the value of `ESAVE` (maximum electron energy at which range rejection is considered) for each region.

Component Module Summary This section of the listing files summaries the geometry and region information for each component module in the order that the component modules appear in the simulation geometry. This section is useful for making sure that the geometry, as interpreted by BEAMnrc, is the same as the geometry the user intended to input. This section gives specific dimensions of a component module and the specific location of each region within a component module. For each region, `ECUT`, `PCUT`, `ECUTRR`, `ESAVE`, the material, the dose zone, and the bit number are output. There is enough information in the component module summary to reconstruct the input file.

Execution Information and Warning Messages This section contains a post-run summary of some relevant run-specific information, such as the elapsed and CPU time required for the run, the total number of charged particle steps, the fraction of steps for which multiple scattering was switched off, *etc.* When a phase space source is used, this section contains a summary of the source characteristics (number of electrons, number of photons, maximum energy, *etc.*) determined from those particles actually used in the simulation. Some warning messages, such as those printed when all particles in a phase space source have been used and the source must be restarted with the first particle, are printed in this section.

Number, Fluence, Average Energy and Average Angle Results This section summaries the results for all scoring planes in the geometry. For the purpose of `.egslst` output, each scoring plane is divided into a user specified number of square or annular scoring zones with the zone half-widths (square) or radii (annular) also specified by the user. BEAMnrc outputs a summary of each scoring plane, including the plane's Z-position, the number of particles that crossed it, and the radii or half-widths of the scoring zones, followed by the actual fluence results for the scoring zones. The fluence is taken as the weighted sum of $1/\cos\theta$ where θ is the angle of the particle with respect to the z-axis. The number averaged energy and number averaged angle are also output. Scoring zones are numbered in order of increasing radius or half-width. Note that the fluence results may be output for one more scoring zone than the number of scoring zones input by the user. This "extra" scoring zone represents the area of the scoring plane not covered by the scoring zones. Fluence results for particles crossing the scoring plane only once and those for particles crossing the plane two or more times ("multiple passers") are output separately. Note that all relevant results (number,

fluence, dose, energy deposited) are normalized per initial source particle in the entire accelerator model, *i.e.* per initial particle which is not from a phase space file. This is done by using the variable `NINCPHSP` discussed in section 7.1. one advantage of this method is that the same results are obtained automatically, whether the calculation is split up into several components or not.

Dose Results The first table shows the total dose and total energy deposited in each of the dose scoring zones that the user set up in the accelerator. Note that each dose scoring zone may include several geometric regions. The second table (if contaminant dose calculations are asked for) shows the dose in each region due to the contaminant particles which are defined as the charge state selected by the user as the particles cross a user-defined planar boundary. This option was originally designed for calculation of dose in a phantom but is applied in general. Finally, the doses in each region with bit filters applied are given. These are called "dose components" and are numbered to correspond to the order of the bit filters in the `.egsinp` file. For more information about bit filters and dose components see section 9 below.

.egslog The log file (created only for a batch run) contains all of the dialogue that would be appear on the screen during an interactive run. Thus, input parameters are echoed and any input errors will appear here. Below the input is a summary of the materials in the simulation (and whether or not Rayleigh scattering data is available for each material), a summary of `PRESTA` parameters, followed by the run-time execution messages (number of histories completed, CPU time for a batch, warning and error messages, *etc.*). After completion of a batch job, looking in the log file is often the easiest way to make sure that all histories were completed and that no run-time errors, such as endless loops, or negative `USTEP` errors, occurred. When using a phase space file for the source, the log file contains a warning line every time all particles in the phase space source have been used, necessitating a restart of the source from the beginning.

.egsdat This file contains all of the information required to restart a run and also the information required to obtain dose and fluence when parallel jobs are recombined. Data stored in the `.egsdat` file include:

1. $\sum_{i=1}^{nhist}(\text{energy deposited})_i$, $\sum_{i=1}^{nhist}(\text{energy deposited})_i^2$, $\sum_{i=1}^{nhist}(\text{fluence})_i$, and $\sum_{i=1}^{nhist}(\text{fluence})_i^2$ in each dose and fluence scoring zone, where `nhist` is the number of primary (non-phase space) histories completed.
2. elapsed CPU time
3. no. of histories completed, no. of primary histories completed
4. state of the random number generator at the end of the last batch

When a run is restarted, BEAMnrc reads the energy deposited and fluence data (item 1 above) for the previous run from the `.egsdat` file and then adds it to the current results before calculating doses, fluence values and uncertainties [17]. When recombining parallel jobs, BEAMnrc reads the energy deposited and fluence data for each parallel job from its `.egsdat` file, sums the data across all jobs and (without running any further simulations) calculates the doses, fluence values and uncertainties for the complete

simulation. See section 10.3 about the input variable IRESTART and Section 13 about parallel runs.

.egsrns This file contains the complete state of the random number generator at the start the current batch (`ISTORE=0`) or current history (`ISTORE=1`) in a simulation. This is only used for debugging when problems occur. These data are used to restart the run with this RNG when `ISTORE` is set to -1. Note that running with `ISTORE=1` slows down a simulation significantly.

.egsplot This file contains dose *vs* depth for all dose components when a CHAMBER CM is used as a depth dose phantom. The format of the file is suitable for plotting with `xmgrace`. Note that the output of the file only makes sense for doses scored in a CHAMBER depth dose phantom.

.egsphsp1(2 or 3) These files contain phase space information (see section 7) for all particles crossing the scoring planes. If `IO_OPT` is set to 1 (see section 10.4), no phase space is scored, and these files are empty.

.egsgeom If a graphical output is requested (`IWATCH`=4), this file specifies the accelerator geometry for display by `EGS_Windows` [16].

.egsgph If a graphical output is requested, this file contains the track histories for display by `EGS_Windows`. Note that this file becomes large for even a few 10's of histories.

.errors Lists any errors in the EGSnrc transport parameter inputs. The contents of this file are generated by the code `$HEN_HOUSE/src/get_inputs.mortran`, which takes care of reading the EGSnrc transport parameter inputs in BEAM (and other user codes).

.eo Contains output from the network queuing system (batch job submissions only).

.mederr Output from pegsless routines including errors, warnings and information regarding where the medium composition and density are being read from (*i.e.*, a density correction file or the `.egsinp` file itself). This file is only output for pegsless runs (see section 16.3).

Since the `.egsphsp` files are often very large, and may exceed the available disk space, BEAMnrc provides options for changing the output directory for `.egsphsp` files from the default output directory, `$EGS_HOME/BEAM_myaccel` (see Section 7.3 for more information). Changing the output directory for the other output files, however, is more complicated, and involves going into `$HEN_HOUSE/src/egs_utilities.mortran` and changing the directory path specified in subroutine `egs_open_file`.

2.11 Changing the defaults

At the top of the `.egslog` file, there is a list of 10 user selectable internal parameters which the user may wish to vary. These are all found in `$OMEGA_HOME/beamnrc/beamnrc_user_macros.mortran` along with 10 or so other parameters you may change.

```
 The following internal parameters are set:
 Max number of CMs: 20                Max number of media  12
 Max number of regions: 250           Max stack: 10000
 Max bremsstrahlung split: 2000       Max number dose zones:  50
 Max number of scoring planes:  3     Max number of scoring zones:   5
 Max number dose components: 12       Minimum air gap:      0.0100 cm
   All of above can be adjusted in beamnrc_user_macros.mortran
```

Some of these parameters are obviously correlated, so *eg.* the `Max stack` should have a value at least 4 times greater than `Max bremsstrahlung split`. Nonetheless, the user should feel free to vary these parameters if they need to. The default settings should be OK using directional bremsstrahlung splitting in an MV photon beam.

Once any of the parameters in `beamnrc_user_macros.mortran` have been changed, the accelerator must be recompiled to make the changes effective.

On a stand alone system you may be able to change `$OMEGA_HOME/beamnrc/beamnrc_user_macros.mortran` but this change will affect all accelerator models recompiled after that. If you wish to make the change for just one accelerator, then copy `$OMEGA_HOME/beamnrc/beamnrc_user_macros.mortran` to the directory of the local accelerator, make the changes in that copy and then edit the `sources.make` file and delete `$(BEAM_HOME)` before `beamnrc_user_macros.mortran`. When the accelerator is recompiled, it will pick up the new version of `beamnrc_user_macros.mortran`.

2.12 Some Details

This section can be skipped without problems for new users.

2.12.1 The `BEAM_myaccel.io` File

This file (which is a renamed copy of `$OMEGA_HOME/beamnrc/default_beam.io`) is put into your `BEAM_myaccel` directory by `beam_build` when you build your accelerator. This file can be used to associate file names with Fortran unit numbers. In the current version of BEAMnrc, all output files (eg `.egsphsp#`, `.egsdat`, `.egsplot`, etc) are opened explicitly from within the code and so the default `BEAM_myaccel.io` is empty (except for comments at the top). However, the file is useful if you wish to customize BEAMnrc to write your own output quantities. For example, if you wish to output `QUANTITY` to the file `inputfile.myoutput`, the easiest way is to go into `beamnrc.mortran` and add statements, `WRITE(UNIT,FORMAT)QUANTITY;` (where `UNIT` is an arbitrary Fortran unit number), where required and then add the line:

```
UNIT    .myoutput
```

to the `BEAM_myaccel.io` file (Note that the only the file extension needs to be specified).

Files specified in `BEAM_myaccel.io` are opened at the beginning of the simulation and are not closed until the simulation has completed. Moreover, the files specified here are ALWAYS opened (ie there are no conditions on whether the file is created or not), so if no

quantity is written an empty file will be created.

2.12.2 What's in BEAM_myaccel_macros.mortran and BEAM_myaccel_cm.mortran

The file BEAM_myaccel_macros.mortran, created by beam_build when you build an accelerator, comprises the following code (in this order):

- beamnrc_cm_macros.hdr. Copied from $OMEGA_HOME/beamnrc, this is a short header file with nothing but comments.
- $CM_LIST macro. This macro has the form:

```
REPLACE{$CM_LIST} WITH {CMLIST(
       CMID1,
       CMID2,
       CMID3,
       CMID4
   )}
```

  where CMID1 is the identifier for the first CM, CMID2 is the identifier of the second CM, etc., as specified in the myaccel.module file. This is ultimately expanded during MORTRAN compilation of the accelerator to generate a list of CM identifiers (ie CMLIST(1)='CMID1', etc) which is used in many places in the BEAM code.

- $CM_TYPE macro, which has the form:

```
REPLACE{$CM_TYPE} WITH {CMTYPE(
       CM1,
       CM2,
       CM3,
       CM4
   )}
```

  where CM1 is the name of the first CM (eg SLABS), CM2 is the name of the second CM, etc, as specified in myaccel.module. During MORTRAN compilation, this macro gets expanded to a list of CM names (ie CMTYPE(1)='CM1', etc), which is used in many places in the BEAM code.

- CM_macros.mortran for each CM in the accelerator model (in the order in which they appear in the myaccel.module file) with CM replaced by the CM identifier as specified in myaccel.module. The CM_macros.mortran files are copied from the directory $OMEGA_HOME/beamnrc/CMs. Note that even if a given CM is used more than once in an accelerator, a CM_macros.mortran file for each occurence of the CM will be copied into BEAM_myaccel_macros.mortran, but, of course, the identifier used to replace CM will be different in each case.

The BEAM_myaccel_cm.mortran file, also created by beam_build when you build an accelerator, consists of the CM_cm.mortran files for every CM in the accelerator (in the order specified in myaccel.module) with CM replaced by the CM identifier specified in myaccel.module. Similar to BEAM_myaccel_macros.mortran, even if a CM occurs more than once in the accelerator, CM_cm.mortran is copied into BEAM_myaccel_cm.mortran for every occurence of the CM, but the identifier in each case will be different. The CM_cm.mortran files are copied from $OMEGA_HOME/beamnrc/CMs.

2.12.3 Files used during Compilation with make

The make command uses several different files to direct the concatenation and compilation of the final MORTRAN code. This section describes those files, along with their functions, which are likely to be relevant to BEAM users.

Makefile Located in the BEAM_myaccel subdirectory and created by beam_build when the accelerator is built. This provides the "overall" instructions for compilation. It tells the compilation to include the $HEN_HOUSE/specs/config.conf for your particular "config" (ie gcc, pgf77, Windows, etc). It also instructs the compilation to include the files $HEN_HOUSE/specs/beamnrc.spec and $HEN_HOUSE/makefiles/beam_makefile. Finally it defines the name of the accelerator (ie myaccel) for use in other files.

config.conf Located in the $HEN_HOUSE/specs subdirectory, where config is the name of the configuration you are running (eg gcc, pgf77, win2k, etc). This file contains many definitions essential for compiling all user codes. Definitions include:

- DSEP The directory separator for your machine ("/" for Unix/Linux, "\" for Windows).
- my_machine The name of your machine.
- make_prog The make command used on installation.
- Definitions of $HEN_HOUSE, $EGS_HOME and $SPEC_DIR (directory where the config.conf file is found)
- Definitions of the f77 and C compilers to be used, along with the default compiler options (eg optimization levels).

Other variables defined in config.conf include:

- CUTIL_OBJECTS Object files compiled from the C utility codes (such as file locking functions, etc) linked with BEAM at compile time. On Unix/Linux, machines this will point to egs_c_utils.o if you have a working C compiler and is empty otherwise. On Windows machines, CUTIL_OBJECTS will point to egs_c_utils.obj, a precompiled object file that eliminates the need to have a working C compiler when using these functions on a Windows machine.
- BEAMLIB_OBJECTS The object file required at compilation if a user code (such as DOSXYZnrc) is to use a BEAM shared library as a source. On Unix/Linux systems, BEAMLIB_OBJECTS will point to load_beamlib.o if you have a working C

compiler, and will be empty otherwise. On Windows machines BEAMLIB_OBJECTS will point to load_beamlib.obj, which is a precompiled object file and elminates the need to have a C compiler when using a BEAM library source on a Windows machine.

- BEAMLIB_EXTRA_LIBS The library required at compilation if a user code is to be able to use a BEAM shared library source (-ldl on Unix/Linux machines with a working C compiler, nothing otherwise), and also $(IAEA_LIB), which, if set, points to the library required for a user code to use IAEA-format phase space files as sources (see next item).
- IAEA_LIB and IAEA_PHSP_MACROS Variables defining the library of IAEA phase space handling routines (IAEA_LIB) and the Mortran macros which use these routines (IAEA_PHSP_MACROS). If your system does not have a working C++ compiler, then the IAEA phase space handling routines cannot be compiled and these two variables will be left blank. Variables are used during the compilation of BEAMnrc and other EGSnrc user codes and must be defined in order to be able to read/write phase space data in IAEA format.

The config.conf file also instructs compilation to include the files $HEN_HOUSE/specs/[unix.spec or windows.spec] and $HEN_HOUSE/specs/all_common.spec described below.

unix.spec **or** windows.spec Contains definitions (such as the extension to add to an executable file) unique to Unix or Windows.

all_common.spec Contains definitions common to all systems. This including the mortran sources and macros required to compile a generic EGSnrc user code. This file also defines the variable RANDOM, for the random number generator, but the definition of RANDOM in beamnrc.spec (see below) overrides this one.

beamnrc.spec Located in the $HEN_HOUSE/specs subdirectory, this file contains definitions required for compiling codes in the BEAM system. Among these are the definitions of directories, $OMEGA_HOME, $BEAM_HOME, $DOSXYZ_HOME and $CM_HOME. beamnrc.spec also defines the extension, FEXT to add to the Fortran code output by the MORTRAN compiler. Currently, FEXT = F, which instructs the compiler to use the C-preprocessor before compiling the code. This is necessary for the implementation of parallel processing in BEAM. If, for some reason, you do not have a C compiler on your system, then the installation will set FEXT = f. In addition, the variable SOURCES in beamnrc.spec defines all the MORTRAN macros and codes to concatenate and the order in which to concatenate them to create the final mortjob.mortran file that is then Fortran compiled (see section 2.12.4 below for more details). beamnrc.spec also defines the variable LIB_SOURCES, which lists the macros and codes (in order) that are concatenated to create the mortjob.mortran when BEAMnrc is compiled as a shared library for use as a source in DOSXYZnrc and other user codes (see section 2.3.1 for more information). In practice, the definitions of SOURCES and LIB_SOURCES are copied into a file called sources.make in the accelerator directory when the accelerator is built, and these latter definitions supersede those in beamnrc.spec (see below for more about sources.make). Finally, beamnrc.spec defines the variable RANDOM,

which determines the random number generator to be used in the simulation. Two random number generators, `RANLUX` and `RANMAR` are included in the EGSnrc system. Currently, `RANMAR` is the default random number generator. See section 10.8 for more information about the random number generators and how to switch from the default RANMAR to RANLUX.

`beam_makefile` Located in `$HEN_HOUSE/makefiles`, this defines the final compilation of the MORTRAN and Fortran codes. All of the `make` options outlined above are taken care of here. In addition, this file sources the files `$EGS_HOME/BEAM_myaccel/modules.make` and `$EGS_HOME/BEAM_myaccel/sources.make` created when the accelerator was built (more about these files below). `beam_makefile` ensures that the accelerator will be recompiled if there is a change in any of the MORTRAN code or macro files used by the accelerator (these are compiler "dependencies"). `beam_makefile` also automatically rebuilds (by calling `beam_build.exe`) the accelerator if there is a change in any of the individual CM macros or MORTRAN codes or if `myaccel.module` has been changed to use different CMs and/or CM identifiers (these are build "dependencies").

`modules.make` Located in `$EGS_HOME/BEAM_myaccel` and created by `beam_build` when the accelerator is built. This file is a list of the MORTRAN codes (`$HEN_HOUSE/omega/beamnrc/CMs/CM_cm.mortran`) and macros `$HEN_HOUSE/omega/beamnrc/CMs/CM_macros.mortran`) for all CMs used in the accelerator. It is through the `modules.make` file that `beam_makefile` has access to the individual CM macros and MORTRAN codes and can, thus, rebuild the accelerator if any of the CM coding changes.

`sources.make` Located in `$EGS_HOME/BEAM_myaccel` and created by `beam_build` when the accelerator is built. This file contains definitions of the variables `SOURCES` and `LIB_SOURCES` copied from the `beamnrc.spec` file (see above). These definitions override those in `beamnrc.spec` and allow the user to customize their accelerator, adding their own source code and/or changing the directory from which source codes are picked up. Note when adding your own coding that macros must appear above the MORTRAN coding they are used in and, in the case where there are several occurrences/versions of the same macro, the last occurrence is the one that is used. Note that changes made to this file will be lost if you execute `beam_build` or build the accelerator using the GUI. This file must be adjusted to include source 31 for beam characterization (see section 4.16.

2.12.4 Files Concatenated to Create `mortjob.mortran`

`mortjob.mortran`, the file which is actually MORTRAN compiled on the way to creating an executable for your accelerator, consists of many concatenated macros and MORTRAN files. The files which are concatenated and the order in which they are concatenated are determined by the `SOURCES` (for stand-alone accelerators) or `LIB_SOURCES` (for shared libraries) definition in the file `$EGS_HOME/BEAM_myaccel/sources.make` discussed in section 2.12.3 above. Figure 5 below shows all the files that are concatenated to create a `mortjob.mortran` for a stand-alone accelerator simulation.

Note that macros must appear before the MORTRAN code in which they are used. Hence, the first portion of `mortjob.mortran` consists entirely of macro files.

Figure 5: Files concatenated to form the complete BEAM source code, `mortjob.mortran`, where `BEAM_myaccel_cm.mortran` and `BEAM_myaccel_macros.mortran` contain the source code related to all the selected CMs. The files and the order of concatenation are defined by the `SOURCES` variable in `$EGS_HOME/BEAM_myaccel/sources.make`.

A brief description of each of these files follows:

`egsnrc.macros` Macros for the EGSnrc system. This includes common blocks, such as `STACK`, that are used by all user codes. See the EGSnrc Manual[1] for more details.

`timing.macros` Definitions of timing macros used in BEAMnrc (and other user codes). Replaces the macros with calls to EGS subroutines used to determine elapsed CPU time, elapsed total time, etc.

`phsp_macros.mortran` Macros used by BEAMnrc to read/write data from/to phase space files. See section 7 for more details.

`iaea_phsp_macros.mortran` Macros used by BEAMnrc to read/write phase space data in IAEA format. This file is only present if there is a working C++ compiler (required to compile the library of IAEA phase space handling routines). Otherwise, these macros are defined as blank (";") in `phsp_macros.mortran` and IAEA-format phase space data cannot be handled.

`machine.macros` Machine/compiler-dependent macros. Defines `$LONG_INT` as `integer*8` if it is available on this machine, the record length factor (`$RECL-FACTOR`) for a 4-byte record in a phase space file, etc.

`ranmar.macros` **(or** `ranlux.macros`**)** Macros used by BEAMnrc (and other user codes) to obtain random numbers, store random number states to a file, retrieve random number states from a file and initialize the random number generator.

`transportp.macros` Macros used by `get_inputs` to define text patterns searched for in the EGSnrc input section of a BEAMnrc input file (see section 11).

pegs4_macros.mortran

pegsless mode

`pegs4_macros.mortran` Macros required for running in pegsless mode. This file contains common block definitions of variables used to calculate and store electron stopping powers and interaction cross sections as well as definitions of the macros, `$INIT-PEGS4-VARIABLES` and `$GET-PEGSLESS-XSECTIONS`, used in the `HATCH` subroutine to initialize data arrays and compute stopping powers and cross sections, respectively. For more information on pegsless mode, see section 16.3.

`beamnrc_user_macros.mortran` Macros defining many default BEAMnrc parameters such as the maximum number of dose/fluence scoring zones, the maximum stack depth, etc. These parameters often need to be changed by the user. See section 2.11 for more details.

`BEAM_myaccel_macros.mortran` Macros for all CMs that comprise `myaccel` concatenated together by `beam_build` with the names of the CMs replaced by their identifiers as specified in the `myaccel.module` file. See section 2.2 for more details.

`egs_utilities.mortran` Various EGSnrc utility codes. Includes the `egs_init` functioning for initializing variable arrays and opening any Fortran output units with the names given in `BEAM_myaccel.io` (see section 2.12.1). Also includes `egs_combine_runs` routine for recombining results after parallel runs (see section 13.3).

`beam_main.mortran` BEAMnrc main code. Consists of calls to the main BEAMnrc subroutines, `beam_init`, `beam_shower_loop` and `beam_finish` (in that order). These subroutines are in `beamnrc.mortran` (see below).

`beam_lib.mortran` Used instead of `beam_main.mortran` when compiling BEAMnrc as a shared library for use as a source (in DOSXYZnrc or other EGSnrc user code). Contains macro (re)definitions essential for use as a source along with subroutines for initializing and sampling the BEAMnrc source. Among these is a redefinition of the

main phase space writing macro which causes phase space data that would normally be written to a phase space file to be stored in an array instead. Incident particles for the simulation using the BEAMnrc simulation as a source are then sampled from this array.

`beamnrc.mortran` Contains definitions of macros used by BEAMnrc, common blocks used by BEAMnrc and all BEAMnrc subroutines.

`xvgrplot.mortran` Subroutine for creating a data file suitable for plotting with xmgr/xmgrace. BEAMnrc uses this subroutine to output an `.egsplot` file (see section 2.10).

`BEAM_myaccel_cm.mortran` Subroutines for all CMs comprising `myaccel` concatenated together by `beam_build` with CM names replaced by their identifiers as specified in `myaccel.module`. See section 2.2 for more details.

`get_inputs.mortran` Coding for reading Monte Carlo transport parameters from the BEAMnrc input file. See section 11 for more about these inputs.

`get_media_inputs.mortran` Coding for reading media definitions from the `.egsinp` file when running in pegsless mode.

`ranmar.mortran` **(or** `ranlux.mortran`**)** Subroutines used by the RANMAR (or RANLUX) random number generator. Many of the macros defined in `ranmar.macros` (or `ranlux.macros`) include calls to these subroutines.

`nrcaux.mortran` Auxiliary MORTRAN routines used by BEAMnrc (and other EGSnrc user codes). Includes `SUBROUTINE WATCH`, which outputs details of each particle step/interaction or particle tracks to the `.egsgeom` file for display using EGS_Windows, depending on the value of the input variable `IWATCH` (see section 10.1 for more on `IWATCH`).

`machine.mortran` Machine/configuration-dependent routines such as date/time routines and system calls. This file is created during EGSnrc installation.

`egs_parallel.mortran` Subroutines for creating, opening, reading from and writing to the job control (`.lock`) file during a parallel run (see section 13).

`pegs4_routines.mortran` Subroutines for calculating electron stopping powers and cross sections when running in pegsless mode. With the exception of some variable definitions, these routines have been copied verbatim from their counterparts in `pegs4.mortran`. See section 16.3 for more information on running in pegsless mode.

`egsnrc.mortran` EGSnrc subroutines. See the EGSnrc Manual[1] for more details.

Note that in previous versions of BEAMnrc, all of the BEAMnrc-specific coding was contained in the file `$OMEGA_HOME/beamnrc/beamnrc.mortran`. Now, however, this coding has been broken up into two files: the BEAMnrc main code is in `$OMEGA_HOME/beamnrc/beam_main.mortran`, and the rest of the macro/subroutine definitions in `$OMEGA_HOME/beamnrc/beamnrc.mortran`. Separating out the BEAMnrc main code was necessary to allow us to compile BEAMnrc as a shared library for use as a source in

DOSXYZnrc and other EGSnrc user codes, since shared libraries do not use the BEAMnrc main code. See section 2.3.1 (page 20) for more information about BEAMnrc as a shared library.

3 Description of main BEAMnrc input file

The first section of an input file (`.egsinp`) for BEAMnrc concerns all issues not related to individual `CMs`. The following is taken from the source code. More detailed discussion of most of these options is given in the following sections (use the Index to find where in most cases), but this is included here as a short summary/reference. Note that the graphical user interface (`BEAMnrc_GUI`) greatly simplifies the task of creating input files[20] and the help files in the GUI mostly duplicate what is below.

```
                    FORMAT OF PARAMETER-DEFINITION(input) FILE
                    ******************************************
                      GENERAL INPUT/OUTPUT PARAMETERS
                      *******************************

  First record    TITLE   80A1
  ************

  Next Record  MEDIUM for nominal air (as in pegs4dat file)
  ***********
               In many CMs, the region about the central-axis or at the
               front or back of the CM, is assumed to be this medium.
               It is thought of and referred to as air, but can be anything.
               Default is VACUUM. MEDIUM must exactly match name in pegs4dat
-------------------------------------------------------------------------------

  Next Record
  ***********
     IWATCH,ISTORE,IRESTART,IO_OPT,IDAT,LATCH_OPTION,IZLAST

             IWATCH = 0 for normal output (the default)
                    = 1 output for every discrete interaction
                    = 2 output for every electron/photon step as well
                    = 4 outputs file for graphics
                    = -N set to 2 on history N, set to 0 on all other
                         histories (for debugging purposes)
             ISTORE = 0 store rn seeds for the 1st history of a batch
                    = 1 store initial rn status (unit 2) for each history
                        being simulated
                    =-1 start first history with rn status from file (unit 2)
                        This is a debugging tool. If run quits, rerun with
                        ISTORE=1, then again ISTORE=-1 and IWATCH = 1/2
                        and/or the debugger on.
          IRESTART = 0 first run for this data set (the default)
```

```
                  = 1 restart of a previous run
                  = 2 just create the input file and exit
                  = 3 read in the raw data from a previous run and do
                     the statistical analysis on dose etc.
                  = 4 read in the .egsdat files from parallel jobs
                      having the same base name as the input file but
                      with the extension _w#, where #
                      can be any positive integer.  These .egsdat files
                      will be summed and then the result analyzed similar
                      to IRESTART=3.
           IO_OPT = 0 phase-space output at each scoring plane(the default)
                  = 1 no phase-space output when particles cross scoring
                      plane
                  = 2 no phase-space output but do data analysis for
                      simplified source models
                  = 3 phase-space output up to 100 k particle histories
                      then do analysis only for simplified source models
                  = 4 output phase space in IAEA format
            IDAT  = 0 store data arrays for re-use (takes time but safer)
                  = 1 don't store them
     LATCH_OPTION = 0 defaults to 2
                  = 1 LATCH for secondaries not inherited from primaries
                      Bits 1-23 set for all regions particle is in
                  = 2 LATCH bits set for all regions particle is in
                      and inherited by secondaries
                      also record bit regions where secondaries created
                      and whether they were created by brem photons
                  = 3 = option 2 but the region numbers are recorded
                      for photons where they interact rather than where
                      they pass through
           IZLAST = 0 do not score ZLAST etc. (the default)
                  = 1 score the z-position of the last site of interaction
                      for photons and creation of electrons by a photon.
                  = 2 score the xyz-position of the last site of
                      interaction in the file $.egs4gph to be used by
                      EGS_WINDOWS. IWATCH=4 must not be used at same time.
                      Note that for phase space inputs, ZLAST is passed
                      through, but XLAST and YLAST are not.
-------------------------------------------------------------------------------

  Next Record            MONTE CARLO CONTROL INPUT
  ***********            ************************

    NCASE,IXXIN,JXXIN,TIMMAX,IBRSPL,NBRSPL,IRRLTT,ICM_SPLIT
           NCASE  = # of histories to run for this simulation
                    (min:$NCASEMIN = 100 for IWATCH=0)
           IXXIN  = 1st random number initial seed (blank or 0 OK)
                    Note that, if using the ranlux random no. generator,
                    this input is the luxury level and should have a
```

```
                value >=0 and <=4.  Otherwise, a default luxury
                level of 1 will be used.
          JXXIN  = 2nd random number initial seed (blank or 0 OK)
          TIMMAX = max cpu time allowed for this run in hours
                (default=0.99)
          IBRSPL = 0 no brem splitting
                 = 1 with uniform brem splitting
                 = 2 with directional bremsstrahlung splitting (DBS)
          NBRSPL = brem splitting number AND
                annihilation splitting number (if IRRLTT=2)
          IRRLTT = 0 no Russian Roulette (the default).  Also, no
                   annihilation or higher-order splitting.
                 = 1 no longer used.  This defaults to IRRLTT=2
                 = 2 perform Russian Roulette--eliminates all but one
                   of secondary charged particles created by split
                   photons.
                   If the surviving particle undergoes another (higher-
                   order) bremsstrahlung event or an annihilation,
                   resulting photons are split again by NBRSPL for
                   IBRSPL=1. (ie uniform splitting)

          Note: The input IRRLTT is automatically set to 0 if IBRSPL=2
                This is because Directional Bremsstrahlung Splitting
                does not use the built-in Russian Roulette of EGSnrc

       ICM_SPLIT = 0 no splitting of photons and electrons as they cross
                   a plane at the start of a user-specified CM
                 > 0 Split photons and electrons a user-specified number
                   of times as soon as they cross the arbitrary
                   splitting plane at the top of this CM #.

Next record (if IBRSPL=2)
***********

   FS,SSD,ICM_DBS,ZPLANE_DBS,IRAD_DBS,ZRR_DBS  (6F12.0)
              FS = radius of field (cm) into which bremsstrahlung photons
                   must be directed if they are to be split.
             SSD = distance from bremsstrahlung target where FS
                   is defined.
     ICM_DBS and   These are only required to define the splitting
                   plane if IBRSPL=2.  As soon as
      ZPLANE_DBS   a fat electron reaches ZPLANE_DBS within CM number
                   ICM_DBS, it gets split NBRSPL times.  This is designed
                   to improve electron statistics in the current
                   implementation of directional bremsstrahlung
                   splitting (DBS).  If ICM_DBS=0, then no electron
                   splitting is done (recommended if only good photon
                   statistics are required).  Note that ZPLANE_DBS is
                   the index of the plane within ICM_DBS, not the Z
```

```
                    position of the plane.  Usually, ICM_DBS will
                    be the CM number of the flattening filter in the
                    accelerator.  If this is modelled using FLATFILT
                    or CONESTAK, then ZPLANE_DBS will denote the layer
                    no. (starting from the top).  If the flattening
                    filter is modelled using CONS3R, then only two
                    planes are available: ZPLANE_DBS=1 is the plane
                    at the top of the structure and ZPLANE_DBS=2 is the
                    plane at the bottom of the structure.  Currently,
                    only FLATFILT, CONESTAK and CONS3R support these
                    inputs.  Usually ZPLANE_DBS is the plane defining the
                    bottom of the flattening filter.
          IRAD_DBS  Set to 1 if you want the NBRSPL split electrons
                    to be distributed in a radially-symmetric manner
                    about the beam axis.  Note that the beam must be
                    radially symmetric above the splitting plane for this
                    to make sense.  Set to 0 (the default) otherwise.
          ZRR_DBS   Z position of the russian roulette plane (cm).  Only
                    required if IBRSPL=2.  This defines the Z
                    position of a plane within the geometry below which
                    non-fat photons about to undergo a compton, pair
                    or photoelectric event will NOT be subject to
                    russian roulette and compton, pair or photoelectric
                    events from fat photons will be split NBRSPL times.
                    This is designed to increase the number of electrons
                    (albeit with a lower weight) below this plane and
                    is only used if electron splitting is on (ie
                    ICM_DBS above is > 0).  Note that radiative events
                    (bremsstrahlung, annihilation) of non-fat electrons
                    below this plane are not split. Usually, the Russian
                    Roulette plane is above the electron splitting plane,
                    and so it is within the flattening filter
                    but somewhere above the bottom.  Note that ZRR_DBS is
                    in cm whereas the electron splitting plane must be
                    on a horizontal boundary in a CM.

  Next record (if ICM_SPLIT>0)
  ***********

     NSPLIT_PHOT,NSPLIT_ELEC (2I3)
            NSPLIT_PHOT = The photon splitting number.
            NSPLIT_ELEC = The electron splitting number.
     This input is unrelatted to bremm splittin and is designed to improve
     efficiency in phantom depth-dose calculations.
-------------------------------------------------------------------------

                SOURCE GEOMETRICAL CONFIGURATION INPUT
                **************************************
```

```
  Next Record  specifies charge and type of source of incident particles
               The meaning of parameters depends on source type

  ISOURC = 0   PARALLEL BEAM INCIDENT FROM THE FRONT (+VE Z-AXIS)
  **********

     IQIN,ISOURC,RBEAM,UINC,VINC,WINC
             IQIN    charge of the incident beam (defaults to 0)
             ISOURC  =  0
             RBEAM   radius of parallel beam in cm (defaults to max radius
                     if RBEAM < 0 or > max radius
                     max radius =RMAX_CM(1) for circular CM boundary
                                =RMAX_CM(1)*SQRT(2) for square CM )
             UINC    incident x-axis direction cosine
             VINC    incident y-axis direction cosine
             WINC    incident z-axis direction cosine
                     Note: (UINC,VINC,WINC) get automatically normalized
                           defaults to (0.0,0.0,1.0)
-------------------------------------------------------------------------

  ISOURC = 1   POINT SOURCE ON Z-AXIS INCIDENT FROM THE FRONT
  **********                circular or square

     IQIN,ISOURC,DISTZ,RBEAM,GAMMA,XINL,XINU,YINL,YINU
             IQIN    charge of the incident beam (defaults to 0)
             ISOURC  =  1
             DISTZ   distance of the point source above front of first
                     CM at Z=Z_min_CM(1). Defaults to  100 cm
             RBEAM   radius of the beam on front of first CM
                     defaults to max radius of first CM if GAMMA is also 0.
                     or:
                     If negative, denotes that that field on front of
                     first CM is rectangular
             GAMMA   1/2 angle about z-axis(degrees) of source,
                     ONLY if RBEAM=0.0
             XINL,XINU,YINL,YINU  Lower and upper X boundaries and
                     Y boundaries of rectangular field on first CM in
                     cm.  ONLY if RBEAM<0.
-------------------------------------------------------------------------

  ISOURC = 3  UNIFORM ISOTROPICALLY RADIATING SOURCE WITHIN CMs
  **********        Circular: Vertical ring centred on Z-axis
                       or     horizontal cylinder centred parallel to X-axis

     IQIN,ISOURC,RMINBM,RBEAM,ZSMIN,ZSMAX,i_dsb,splitcm_dsb,dsb_delta
             IQIN    charge of particles from source (defaults to 0)
             ISOURC  =  3
             RMINBM  inner radius of vertical ring (RBEAM >= 0) (cm)
                     or:
```

```
                Z position of centre of horizontal cylinder
                (RBEAM < 0) (cm)
          RBEAM outer radius of vertical ring (RBEAM >= 0) (cm)
                or:
                -radius of horizontal cylinder (RBEAM < 0) (cm)
          ZSMIN Z of top of vertical ring (RBEAM >= 0) (cm)
                or:
                min. X of horizontal cylinder (RBEAM < 0) (cm)
          ZSMAX Z of bottom of vertical ring (RBEAM >= 0) (cm)
                or:
                max. X of horizontal cylinder (RBEAM < 0) (cm)
          i_dsb Set to 1 to use directional source biasing.  Note that
                directional bremsstrahlung splitting (IBRSPL=2) must
                also be used.  Splitting number (NBRSPL), splitting
                field radius (FS), source to surface distance
                (SSD) and electron splitting parameters (if e-
                contamination is desired) are read from the DBS
                inputs.
     splitcm_dsb The CM no. at which primary photons are split and
                radially redistributed about the Z-axis.  Photons
                are split/redistributed immediately upon entering
                CM no. splitcm_dsb.  Note that this should be the
                the no. of the first CM in the treatment head without
                radial symmetry.  The number of times a photon is
                split depends upon the radial bin into which it
                is directed.  Bin radii are determined by the input
                dsb_delta below.  Set splitcm_dsb=0 for no splitting/
                redistribution.
       dsb_delta The min. linear distance, in cm, between split/
                redistributed photons, projected to the SSD of the
                splitting field. dsb_delta is used to divide the
                splitting field into radial bins, where photons
                directed into bin i are split i times.

       NOTE: The sign of RBEAM determines if the source will be a
                     vertical ring a horizontal cylinder.
             The Z-span of the source must be in the range
                     Z_min_CM(1)-Z_min_CM(MAX_CMs+1).
             Currently, this source is limited to being placed within
                     CONESTAK, FLATFILT or SIDETUBE
-----------------------------------------------------------------------
  ISOURC = 3a  A cylindrical, isotropically radiating Co60 source within CMs
  **********  using directional source biasing (DSB).

    IQIN,ISOURC,RMINBM,RBEAM,ZSMIN,ZSMAX,i_dsb,DSB_DELTA
          (same is source 3)
-----------------------------------------------------------------------


  ISOURC = 5   NRC SWEPT BEAM SOURCE
```

```
  **********

     IQIN,ISOURC,GAMMA,RBEAM
             IQIN    charge of the incident beam (defaults to 0)
             ISOURC  =  5
             GAMMA   1/2 angle of cone in degrees
             RBEAM   radius of beam spot at Z = 0.0 (cm)

     Note apex of cone is at x=y=0,z=Z_min_CM(1)
-----------------------------------------------------------------------

  ISOURC = 6   RECTANGULAR BEAM INCIDENT FROM THE FRONT
  **********

     IQIN,ISOURC,XBEAM0,YBEAM0,XBEAM,YBEAM
             IQIN    charge of the incident beam (defaults to 0)
             ISOURC  = 6
             XBEAM0  X position of centre of beam (cm)
             YBEAM0  Y position of centre of beam (cm)
             XBEAM   half-width in X direction (cm)
             YBEAM   half-width in Y direction (cm)
-----------------------------------------------------------------------

  ISOURC = 7   SCANNING BEAM SOURCE (sawtooth like Therac20)
  **********

     IQIN,ISOURC,FD_AT100, IRATIO_YXF, RBEAM
             IQIN    charge of the incident beam (defaults to 0)
             ISOURC  =  7
             FD_AT100   length & width of scanning field at  SSD=100cm
             IRATIO_YXF =  the number of Y scans per X scan
                          (rounds 2*IRATIO_YXF up to nearest odd
                          number--default IRATIO_YXF = 6.5)
             RBEAM   radius of the beam at Z=0, defaults to 0.01cm
-----------------------------------------------------------------------

  ISOURC = 8   SCANNING BEAM FOR MM50 (uniform circular beam from
  *********                           a point on axis at Z=0)

     IQIN,ISOURC,DISTZ,RBEAM
             IQIN    charge of the incident beam (defaults to 0)
             ISOURC  =  8
             DISTZ   SSD (default to 100 cm)
             RBEAM   radius of scanned beam at SSD.  If set <=0 or
                     too large, then RBEAM gets reset to
                     RMAX_CM(1)*DISTZ/Z_min_CM(1) (circular CM 1) or
                     SQRT(2)*RMAX_CM(1)*DISTZ/Z_min_CM(1) (square CM 1)
                     so that particles strike the front of
                     CM 1 within a circle of radius RMAX_CM(1)
```

```
                    or SQRT(2)*RMAX_CM(1).
             RBEAM0  Radius of beam spot at Z=0cm.  Defaults to 0
                     if RBEAM0<0 and gets reset to
                     RMAX_CM(1)*DISTZ/Z_min_CM(1)-RBEAM (circular CM 1)
                     or SQRT(2)*RMAX_CM(1)*DISTZ/Z_min_CM(1)-RBEAM
                     (square CM 1) if RBEAM+RBEAM0 >
                     RMAX_CM(1)*DISTZ/Z_min_CM(1) or >
                     SQRT(2)*RMAX_CM(1)*DISTZ/Z_min_CM(1).
    For this source the particles start at Z_min_CM(1) and hence
    Z_min_CM(1) must be >= 0.0
-----------------------------------------------------------------------

  ISOURC = 9   SCANNING BEAM FOR MM50 (discrete field coverage from
  *********                           a point source at Z=0)

     IQIN,ISOURC,DISTZ,NPTS_SRC9
             IQIN    charge of the incident beam (defaults to 0)
             ISOURC  =  9
             DISTZ   SSD (default to 100 cm)
         NPTS_SRC9   the number of discrete points at the SSD
                     defaults to $MAXPTS_SRC9 if NPTS_SRC9 > $MAXPTS_SRC9
                     or 1 if NPTS_SRC9 <= 0.

  Next record (if ISOURC=9)
  ************************

     Repeat for I=1,NPTS_SRC9

     X_SRC9(I),Y_SRC9(I),PROB_SRC9(I) (3F15.0)
              X_SRC9(I)     X coordinate of point I at the SSD (cm)
              Y_SRC9(I)     Y coordinate of point I at the SSD (cm)
              PROB_SRC9(I)  probability of a particle being at point I

     Note that PROB_SRC9(I) need not be normalized; they are
     automatically normalized in the source routine.
    For this source the particles start at Z_min_CM(1) and hence
    Z_min_CM(1) must be >= 0.0
-----------------------------------------------------------------------

  ISOURC = 10   PARALLEL CIRCULAR BEAM INCIDENT FROM THE SIDE
  ***********
                (NOTE: beam facing X-AXIS, I.E., UINC should be < 0.0   )
                (this source should only be used together with CM XTUBE )
                (for simulating the target of an X-ray tube.            )
                (XTUBE should always be the first CM in the geometry.   )

     IQIN,ISOURC,RBEAM,UINC,VINC,WINC
             IQIN    charge of the incident beam (defaults to 0)
             ISOURC  =  10
```

```
            RBEAM   radius of parallel beam in cm (defaults to max radius)
            UINC    incident X-axis direction cosine (UINC < 0.0)
            VINC    incident Y-axis direction cosine
            WINC    incident Z-axis direction cosine
                    Note: (UINC,VINC,WINC) get automatically normalized
                          defaults to (-1.0,0.0,0.0)
-----------------------------------------------------------------------

  ISOURC = 13   PARALLEL RECTANGULAR BEAM INCIDENT FROM THE SIDE
  ***********
               (Note beam facing X-axis, i.e., UINC should be < 0.0    )
               (this source should only be used together with CM XTUBE )
               (for simulating the target of an X-ray tube.            )
               (XTUBE should always be the first CM in the geometry.   )

     IQIN,ISOURC,YBEAM,ZBEAM,UINC,VINC
            IQIN    charge of the incident beam (defaults to 0)
            ISOURC  =  13
            YBEAM   half-width of parallel beam in cm (defaults to 0.2 cm)
            ZBEAM   half-height of parallel beam in cm (defaults to 0.2 cm)
            UINC    incident X-axis direction cosine (UINC < 0.0)
            VINC    incident Y-axis direction cosine
                    (incident Z-axis direction cosine WINC default to 0.0)
                    Note: (UINC,VINC) get automatically normalized
                          defaults to (-1.0,0.0,0.0)
-----------------------------------------------------------------------

  ISOURC = 15   NRC SWEPT BEAM WITH BEAM DIVERGENCE AND RADIAL INTENSITY
  ***********                                              DISTRIBUTION

     IQIN,ISOURC,GAMMA,ZFOCUS,RTHETAIN,THETAIN
            IQIN       charge of the incident beam (defaults to 0)
            ISOURC     =  15
            GAMMA      half angle of the cone swept by the beam (degrees)
            ZFOCUS     Z position of the apex of the cone (cm)
            RETHETAIN  radius at which THETAIN, the divergence angle of the
                       beam, is specified (cm).  RTHETAIN must be > 0.
            THETAIN    divergence angle of the beam (degrees).  If GAMMA
                       is not 0, then THETAIN can be set to 0; otherwise
                       it must be > 0.

      Note that particles are always incident at Z_min_CM(1), regardless of
      the value of ZFOCUS

  Next record (If ISOURC=15)
  ***********
      SPCNAM       FILENAME (with EXT) containing description of the radial
                   intensity distribution of the incident particles
                                          (maximum 256 characters)
```

```
            ------------------------------------------------------
            FILE FORMAT for SPCNAM:
              NRDIST
              (RDISTF(I),RPDF(I),I=1,NRDIST)
                    NRDIST     # radial bins
                    RDISTF(I)  upper radius of bin I (cm)
                    RPDF(I)    probability of particle being in bin I.
-------------------------------------------------------------------------

  ISOURC = 19   PARALLEL ELLIPTICAL BEAM FROM FRONT GAUSSIAN IN X AND Y
  ***********

     IQIN,ISOURC,RBEAM,UINC,VINC,WINC,sigma_src19,RBEAMY
             IQIN    charge of the incident beam (defaults to 0)
             ISOURC  = 19
             RBEAM   sigma of the 2-D gaussian distribution (RBEAM > 0)
                     in the X-direction in cm
                       or
                     -FWHM of 2-D gaussian distribution (RBEAM < 0) in
                     the X-direction in cm
                     Note: sigma of gaussian distribution is limited to
                           <RMAX_CM(1) for circular CM 1 and
                           <SQRT(2)*RMAX_CM(1) for square CM 1
             UINC    incident x-axis direction cosine
             VINC    incident y-axis direction cosine
             WINC    incident z-axis direction cosine
                     Note: (UINC,VINC,WINC) get automatically normalized
                           defaults to (0.0,0.0,1.0)
      sigma_src19    mean angular spread of particles about Z axis
                     in degrees (none if set <=0).  Overrides incident
                     direction cosines if set > 0, so that beam is assumed
                     to be centred on Z axis.
           RBEAMY    same as RBEAM but for Y-direction.  If set to 0,
                     then RBEAMY=RBEAM for a circular beam.
-------------------------------------------------------------------------

  ISOURC = 21   FULL PHASE-SPACE SOURCE
  ***********

     IQIN,ISOURC,INIT_ICM,NRCYCL,IPARALLEL,PARNUM,ISRC_DBS,RSRC_DBS,
                                              SSDSRC_DBS,ZSRC_DBS
             IQIN       dummy NOT USED. Set = 9 for this source by BEAMnrc
             ISOURC     = 21
             INIT_ICM   particles start at front surface of this CM
                        (INIT_ICM is actually read as a real)
             NRCYCL     Number of times to recycle each particle in a phase
                        space source.  Each particle in the phase space
                        file is used a total of NRCYCL+1 times before
                        going on to the next particle.
```

If NRCYCL is set <=0 then NRCYCL is automatically calculated to use the entire phase space file with no restarts. The calculated NRCYCL does not take into account particles that are rejected because they miss the geometry.
If NRCYCL is set > 0, then the user-input value is used.
If NCASE > no. of particles in the phase space file, then use of NRCYCL is essential for accurate statistics. If you are unsure of how many times to recycle, use the automatically-calculated value of NRCYCL. If this still results in many restarts (because of multiple passers being rejected and/or photons rejected because they fall outside the DBS splitting radius--see below)
then re-run the simulation with NRCYCL set manually to:

```
NCASE/(NNPHSP-NNPHSP*(NPASS_ph_sp+NFAT_ph_sp)/
            (NTOT_ph_sp+NPASS_ph_sp+NFAT_ph_sp) ) -1
```

where NNPHSP is the no. of particles in the file, NTOT_ph_sp is total no. of particles used (not including recycling), NPASS_ph_sp is total no. of multiple passers ignored (not including recycling), and NFAT_ph_sp is the no. of photons rejected (not including recycling because they fall outside the DBS splitting radius at the SSD (only if ISRC_DBS=1--see below)). These numbers are available in the .egslst file. Always round your calculated value of NRCYCL up.

IPARALLEL set >1 if you are distributing the job among IPARALLEL machines. IPARALLEL is used with PARNUM (see below) to partition a phase space source into IPARALLEL equal parts.

PARNUM For each of the IPARALLEL parallel jobs, PARNUM should have a different integer value in the range 1<=PARNUM<=IPARALLEL. The partition of the phase space source that is used for a particular job is then given by:

```
   (PARNUM-1)*(NNPHSP/IPARALLEL)<INPHSP<=
                          (PARNUM)*(NNPHSP/IPARALLEL)
```

where NNPHSP is the total number of particles in the phsp source and INPHSP is the particle no. chosen.

ISRC_DBS Set to 1 if you used directional bremsstrahlung splitting (DBS) in the BEAM simulation used to generate this phase space source and you wish to reject photons not aimed into the splitting field(which are fat). These fat photons compromise statistics. Set to 0 otherwise.

RSRC_DBS DBS splitting radius in BEAM simulation used to

```
                    generate this source (cm).  Only used if ISRC_DBS=1.
          SSDSRC_DBS  SSD at which RSRC_DBS was defined in the BEAM sim.
                      used to generate this source (cm). Only used if
                      ISRC_DBS=1.
          ZSRC_DBS    Z where the phase space source was collected in
                      the BEAM simulation used to generate this source(cm).
                      Only used if ISRC_DBS=1.

     Photons are projected from ZSRC_DBS to SSDSRC_DBS and if they will fall
     outside of RSRC_DBS (based on their trajectory) then they will
     be rejected.  This prevents fat photons from compromising statistics.

  Next record (If ISOURC=21)
  ***********
     SPCNAM        FILENAME (with EXT) contains phase space information
                   (maximum of 256 characters)
-------------------------------------------------------------------------


  ISOURC = 23   BEAM SIMULATION SOURCE
  ***********

     IQIN,ISOURC,INIT_ICM,ISRC_DBS,ALPHA24,BETA24,DIST24
             IQIN       dummy NOT USED. Set = 9 for this source by BEAMnrc
             ISOURC     = 23
             INIT_ICM   particles start at front surface of this CM
                        (INIT_ICM is actually read as a real)
             ISRC_DBS   Set to 1 if you are using directional bremsstrahlung
                        splitting (DBS) in the BEAM simulation source
                        and you wish to reject fat photons (not aimed into
                        the splitting field).  These fat photons compromise
                        statistics.  Set to 0 otherwise.
              ALPHA24   Angle of rotation of source plane about X-axis
                        in degrees.  Positive angle is clockwise rotation.
                        (-90 < ALPHA24 < 90)
               BETA24   Angle of rotation of source plane about Y-axis
                        in degrees.  Positive angle is counter-clockwise
                        rotation. (-90 < ALPHA24 < 90)
               DIST24   Distance of point of rotation above INIT_ICM.

     Note restriction that if ALPHA24~=0 and/or BETA24 ~=0, then INIT_ICM
     must be > 1.  This is because the rotation will result in some
     particles incident within INIT_ICM-1.  Also, both INIT_ICM and
     INIT_ICM-1 must be SLABS, SIDETUBE or FLATFILT, since these are the
     only CMs currently capable of determining initial regions for particles
     incident within them.

  Next record (If ISOURC=23)
  ***********
```

```
     the_beam_code, the_pegs_file, the_input_file (3A80)

       the_beam_code    The name of the BEAM code you are running as a
                        source (ie BEAM_sourceaccelname).  This must have
                        been compiled as a shared library
                        (libBEAM_accelname.so or BEAM_accelname.dll) and
                        exist in EGS_HOME/bin/config.
       the_pegs_file    The pegs data set used by the BEAM simulation source
                        (no .pegs4dat extension).  This must be in
                        HEN_HOUSE/pegs4/data or EGS_HOME/pegs4/data.
       the_input_file   The input file used to run the BEAM simulation
                        source (no .egsinp extension).  This must exist in
                        your EGS_HOME/BEAM_sourceaccelname directory.  It
                        must be a working input file and must be set up to
                        write a phase space file at a single scoring plane.
                        This plane becomes where particles are sampled from
                        for the second-stage BEAMnrc simulation (no phase
                        space file is scored, however).
-------------------------------------------------------------------------

  ISOURC = 24   FULL PHASE-SPACE SOURCE INCIDENT FROM USER-DEFINED ANGLE
  ***********

  Inputs identical to ISOURC = 21 with the following additional line of
  inputs after SPCNAM:

     ALPHA24,BETA24,DIST24

              ALPHA24   Angle of rotation of source plane about X-axis
                        in degrees.  Positive angle is clockwise rotation.
                        (-90 < ALPHA24 < 90)
               BETA24   Angle of rotation of source plane about Y-axis
                        in degrees.  Positive angle is counter-clockwise
                        rotation. (-90 < ALPHA24 < 90)
               DIST24   Distance of point of rotation above INIT_ICM.

     Note restriction that if ALPHA24~=0 and/or BETA24 ~=0, then INIT_ICM
     must be > 1.  This is because the rotation will result in some
     particles incident within INIT_ICM-1.  Also, both INIT_ICM and
     INIT_ICM-1 must be SLABS, SIDETUBE or FLATFILT, since these are the
     only CMs currently capable of determining initial regions for particles
     incident within them.

     The initial idea and much of the coding for Source 24 is courtesy of
     Patrick Downes at University of Cardiff, Wales.
-------------------------------------------------------------------------

  ISOURC =  31     BEAM CHARACTERIZATION MODEL,
  ************
```

```
   IQIN,ISOURC,CMSOU
           IQIN    charge of incident beam (not used)
           ISOURC  = 31
           CMSOU   particles start at the front surface of this CM

Next record (If ISOURC=31)
***********
   SPCNAM        FILENAME (with EXT) contains information on beam model
                 (maximum of 256 characters)
-------------------------------------------------------------------------

Next Record (IF ISOURC <21)     SOURCE ENERGY INPUT
***********                     ******************

   MONOEN (I8)
                    = 0 if monoenergetic beam (the default)
                    = 1 if energy spectrum to be used
                        Note: BEAMnrc sets MONOEN=2 for phase space inputs

 Next Record (IF MONOEN = 0)
 ***********
   EIN (F15.0)
               kinetic energy of the incident beam in MeV
               (defaults to 1.25), only for MONOEN=0

 Next Record (IF MONOEN = 1)
 ***********
   FILNAM(256A1)  FILENAME(WITH EXT) contains spectrum information
                  which must be in  NRC's ensrcV format.
           ------------------------------------------------------
           FILE FORMAT:
           SPEC_TITLE   (A80)
             NENSRC,ENMIN,IMODE (I10,F15.0,I5)
             (ENSRCD(IB),SRCPDF(IB),IB=1,NENSRC) (2F20.0)
                   NENSRC  # Energy bins in spectrum histogram
                   ENMIN   Lower energy of first bin
                   IMODE   =0 assuming counts/bin, =1 counts/MeV
                   ENSRCD(I),SRCPDF(I)  I=1,NENSRC
                           Top of energy bin and probability of
                           initial particle being in this bin.
                           Probability does not need to be normalized
           ------------------------------------------------------

 Next Record (IF MONOEN = 1)
 ***********
   IOUTSP (I5):
           = 0 no spectrum data in output summary
           = 1 include spectrum data in output summary
```

```
-------------------------------------------------------------------------

                    TRANSPORT CONTROL INPUT
                    ***********************
   Next Record
   ***********
     ESTEPIN,SMAX,ECUTIN,PCUTIN,IDORAY,IREJCT_GLOBAL,ESAVE_GLOBAL,IFLUOR

            ESTEPIN Dummy variable (used to be ESTEPE -- max. fractional
                    energy loss/electron step)
            SMAX    Dummy variable (used to be SMAX -- max. step length)
            ECUTIN  electron cut off in MeV - total energy
                    If this is > ECUT as input in the EGSnrc input section
                    (see below), then ECUT is set to ECUTIN.  Default
                    value for ECUT is AE.
            PCUTIN  photon cut off in MeV.  If this is > PCUT as input
                    in the EGSnrc input section (see below), then PCUT
                    is set to PCUTIN.  Default value for PCUT is AP.
            IDORAY  Dummy variable (used to turn Rayleigh scattering on/off)
      IREJCT_GLOBAL = 0 no electron range rejection
                    = 1 do electron range rejection--if residual
                        range to ECUTRR(IRL) is < DNEAR and
                        electron energy is < ESAVE_GLOBAL, terminate
                        history.  ECUTRR(IRL) may vary from component
                        module to component module and is calculated based
                        on the particle making it to the bottom of the
                        accelerator with energy > ECUT
                    = 2 as in =1, but use a non-calculated ECUTRR = ECUT(IRL)
                        this should be used if interested in more
                        than phase-space data at base of simulation
                =-1,-2 Same as above, but now Russian Roulette is played
                        with ALL electrons that can not escape the region
                        and are not fat. Only applicable with DBS.
       ESAVE_GLOBAL  energy below which an electron will be discarded
                    If E<ESAVE_GLOBAL & the electron cannot escape from the
                    current region with E>ECUTRR(IRL). This ignores brem
                    losses.
            IFLUOR  Dummy variable (used to turn X-ray fluorescence on/off)

        The dummy inputs are retained for compatibility with EGS4/BEAM input
        files.
-------------------------------------------------------------------------

  Next Record        Photon Forcing Controls
  ***********        **********************
      IFORCE,NFMIN,NFMAX,NFCMIN,NFCMAX
            IFORCE = 0 normal photon transport (the default)
                   = 1 force photon interaction in the geometry
            NFMIN  number of photon interaction/history at which to start
```

```
                photon-interaction-forcing (defaults to 1)--this
                option has been deleted and NFMIN is now always treated
                as 1.
          NFMAX  number of photon interaction/history after which to
                 stop forcing photon to interact (defaults to 1)
          NFCMIN number of CM to start photon interaction forcing
                 ( default to 1 )
          NFCMAX number of last CM in which photons forced to interact
                 ( default to max_cms )
                 If a particle passes thru NFCMIN to NFCMAX, it is
                 forced to interact there for the first NFMAX
                 interactions. The WEIGHT
                 of this photon is reduced and the remaining WEIGHT is
                 carried by another photon which will be transported
-------------------------------------------------------------------------

  Next record             SCORING PLANE INPUT
  ***********             ******************

    NSC_PLANES, (IPLANE_to_CM(I), I=1,NSC_PLANES)

       NSC_PLANES     number of scoring planes >=0
       IPLANE_to_CM   CM numbers corresponding to the scoring planes
                      fluence is scored at the back of a component module
                      phase space data written from same planes

       (Note only IPLANE_to_CM(1) is used for beam model analysis);

  Next record            SCORING ZONE TYPE/DIMENSIONS
  ***********            ***************************

  Repeat the next pair of lines for ISCORE=1,...,NSC_PLANES

     NSC_ZONES(ISCORE), MZONE_TYPE(ISCORE)
          NSC_ZONES     number of scoring zones within each scoring plane
                        (= 0: maximum number available with equal zone area)
          MZONE_TYPE    0 annular zones (default)
                        1 square (ring) zones
                        2 grid

  Next record (for NSC_ZONES(ISCORE)>0 and MZONE_TYPE = 0 or 1)
  ***********
     (RSCORE_ZONE(ISCORE,I), I=1,NSC_ZONES)  (up to 10/line)
          RSCORE_ZONE   outer radius of each scoring zone in order of
                           increasing radius (MZONE_TYPE = 0)
                        half width from origin of each scoring zone
                           in order of increasing width (MZONE_TYPE = 1)

  Next record (for NSC_ZONES(ISCORE)>0 and MZONE_TYPE = 2)
```

```
**********
   XMIN_ZONE, XMAX_ZONE, YMIN_ZONE, YMAX_ZONE, NX_ZONE, NY_ZONE
          XMIN_ZONE     lower x bound of grid area (cm)
          XMAX_ZONE     upper x bound of grid area (cm)
          YMIN_ZONE     lower y bound of grid area (cm)
          YMAX_ZONE     upper y bound of grid area (cm)
          NX_ZONE       number of grid zones in x direction
          NY_ZONE       number of grid zones in y direction
-------------------------------------------------------------------------

Next record           DOSE COMPONENTS CALCULATION INPUT
**********            ********************************
   ITDOSE_ON
       ITDOSE_ON  = 0 (DEFAULT) only total dose is calculated
                  = 1 total dose and dose components may be calculated

   There are 2 classes of components. First is selected as dose from
   particles or descendents of particular charge as they cross a
   specified boundary.  Second is based on bit selections in the
   variable LATCH, either inclusive or exclusive sets - i.e. depends
   on where particle has been or interacted.

Next record (if ITDOSE_ON=1)
**********
   ICM_CONTAM, IQ_CONTAM (2I5)
        All particles of type IQ_CONTAM (0=photons, 1=charged
        particles) are identified as contaminants when they enter
        the front of CM number ICM_CONTAM and their dose is scored as
        contaminant dose in all dose zones.
        If ICM_CONTAM = 0, no contaminant dose is scored.
        LATCH_OPTION = 1 is not allowed with ICM_CONTAM non-zero

Next record (if ITDOSE_ON=1)
**********
   LNEXC (I5)
         LNEXC: # of dose components scored which exclude dose from
                particles with certain LATCH bits set - i.e. which have
                not been in certain regions.
                LNEXC = 0 is allowed. LNEXC <= $MAXIT - 3

Next records (if LNEXC > 0)
**********
   (L_N_EXC(I,J), J=1, 31 ) (31I5) (repeat LNEXC times, line by line)
         L_N_EXC(I,J): Bit #s in LATCH for dose component I
               (will exclude dose from component I if these bits set)

Next record (if ITDOSE_ON=1)
**********
   LNINC (I5)
```

```
          LNINC: # of dose components scored for particles
                 from specified regions (with designated bit settings
                 in LATCH).  LNINC <= $MAXIT - LNEXC - 3

  Next records (if LNINC > 0)
  ************
     (L_N_INC(I,J), J=1, 31 ) (31I5) (repeat LNINC times, line by line)
           L_N_INC(I,J): Bit #s in LATCH for dose component I.
                         These are in two groups/line, separated by a zero.
                         Of the first group of bits, at least one must
                         be set to be in this dose component.
                         The second group need not be present, but if
                         it is, none of these bits can be set to be in
                         this dose component.

-------------------------------------------------------------------------

  Next record           INPUT FRONT SURFACE (Z) FOR CM 1
  ***********           *******************************

     Z_min_CM(1)  Z-coordinate of front surface for component module 1
                  This includes any air gap and defines front of model.
                  A common value will be 0.0 except for sources 8 & 9.
                  For most sources except for ISOURC=3, 21 & 31, this is
                  also the source plane on which the particles are incident
          Note that the front of all CMs is given w.r.t. z = 0.0, not
          w.r.t. Z_min_CM(1).

  Next record         Blank or dummy line indicating start of
  ***********         input for component modules


  Next records (many)       COMPONENT MODULE INPUT
  *************             **********************

  Component module parameters are input in order of their appearance in the
  code, that is, in the order they occur in $CM_LIST.  See the 'INPUT FROM
  UNIT 5' section in each CM subroutine for the list of input parameters.

   There are two lines before the input of parameters for each CM:
   one is ******************************************** ,
   the other is RMAX_CM, the outer boundary(radius or 1/2 of square) of CM

  -------------------------------------------------------------------------

                          EGSnrc INPUTS
                          *************
    (modified from the description in $HEN_HOUSE/src/get_inputs.mortran)
```

```
The input for parameters associated with EGSnrc follows a format
used by all the standard EGSnrc user codes.  The rest of the BEAMnrc
input has not been changed because the GUI's make the details of the
format irrelevant.

All input associated with selection of EGSnrc transport parameters
is not crucial for the execution as there are default values set.
Therefore, if some of the input options in this section are
missing/misspelled, this will be ignored and default parameters assumed.
As the transport parameter input routine uses get_inputs, a lot
of error/warning messages may be produced on UNIT 15, though.
If you don't have the intention of changing default settings,
simply ignore the error messages.

The delimiters are
             :start mc transport parameter:
             :stop mc transport parameter:

Currently, the following options are available (case does not matter and
           the internal variables are shown in [ ] brackets):


     Global ECUT=     Global (in all regions) electron transport cut
                      off energy (in MeV). If this input is missing,
                      or is < ECUTIN from the main BEAMnrc inputs
                      (See above) then ECUTIN is used for Global ECUT.
                      Global ECUT defaults to AE(medium).
                      [ ECUT ]
     Global PCUT=     Global (in all regions) photon transport cut
                      off energy (in MeV). If this input is missing,
                      or is < PCUTIN from the main BEAMnrc inputs
                      (See above) then PCUTIN is used for Global PCUT.
                      Global PCUT defaults to AP(medium).
                      [ PCUT ]
     Global SMAX=     Global (in all regions) maximum step-size
                      restriction for electron transport (in cm).
                      No SMAX restriction is necessary if the electron
                      step algorithm is PRESTA-II and the EXACT boundary
                      crossing algorithm (the default) is used.  In this
                      case, SMAX will default to 1e10.  However, if either
                       Electron-step algorithm= PRESTA-I
                       or
                       Boundary crossing algorithm= PRESTA-I,
                      then a step-size restriction is necessary, and
                      SMAX will default to 5 cm.
                      [ SMAXIR ]

     ESTEPE=          Maximum fractional energy loss per step.
                      Note that this is a global option only, no
```

```
                region-by-region setting is possible. If missing,
                the default is 0.25 (25%).
                [ ESTEPE ]
XImax=          Maximum first elastic scattering moment per step.
                Default is 0.5, NEVER use value greater than 1 as
                this is beyond the range of MS data available.
                [ XIMAX ]
Boundary crossing algorithm=
                There are two selections possible: EXACT and
                PRESTA-I.  PRESTA-I means that boundaries will
                be crossed a la PRESTA.  That is, with lateral
                correlations turned off at a distance given by
                'Skin depth for BCA' (see below) from the boundary
                and MS forced at the boundary.  EXACT means
                the algorithm will cross boundaries in a single
                scattering (SS) mode, the distance from a boundary
                at which the transition to SS mode is made is
                determined by 'Skin depth for BCA' (see below).
                Default is EXACT since PRESTA-I may result in
                significant dose overestimates when CHAMBER is
                used as a phantom, and EXACT will not significantly
                increase CPU time in most accelerators.
                [ bca_algorithm, exact_bca ]
Skin depth for BCA=
                Determines the distance from a boundary (in elastic
                MFP) at which the algorithm will go into single
                scattering mode (if EXACT boundary crossing) or
                switch off lateral correlations (if PRESTA-I boundary
                crossing). Default value is 3 for EXACT or
                exp(BLCMIN)/BLCMIN for PRESTA-I (see the PRESTA paper
                for a definition of BLCMIN). Note that if you choose
                EXACT boundary crossing and set Skin depth for BCA
                to a very large number (e.g. 1e10), the entire
                calculation will be in SS mode. If you choose
                PRESTA-I boundary crossing and make Skin depth for
                BCA large, you will get default EGS4 behaviours
                (no PRESTA)
                [ skindepth_for_bca ]

Electron-step algorithm=
                PRESTA-II (the default), the name is
                used for historical reasons
                or PRESTA-I
                Determines the algorithm used to take into account
                lateral and longitudinal correlations in a
                condensed history step.
                [ transport_algorithm ]
Spin effects=   Off, On, (default is On)
                Turns off/on spin effects for electron elastic
```

```
                scattering. Spin On is ABSOLUTELY necessary for
                good back-scattering calculations. Will make a
                difference even in 'well conditioned' situations
                (e.g. depth dose curves for RTP energy range
                electrons).
                [ spin_effects ]
Brems angular sampling= Simple, KM, (default is KM)
                If Simple, use only the leading term of the Koch-Motz
                distribution to determine the emission angle of
                bremsstrahlung photons. If KM, complete
                modified Koch-Motz 2BS is used (modifications
                concern proper handling of kinematics at low
                energies, makes 2BS almost the same as 2BN at low
                energies).
                [ IBRDST ]
Brems cross sections= BH, NIST, NRC, default is BH
                If BH is selected, the Bethe-Heitler bremsstrahlung
                cross sections (Coulomb corrected above 50 MeV)
                will be used. If NIST is selected, the NIST brems
                cross section data base (which is the basis for
                the ICRU radiative stopping powers) will be employed.
                Differences are negligible for E > ,say, 10 MeV,
                but significant in the keV energy range.  If NRC is
                selected, NIST data including corrections for
                electron-electron brems will be used (typically only
                significant for low values of the atomic number Z
                and for k/T < 0.005).
Triplet production= On or Off (default).  Turns on/off simulation
                of triplet production.  If On, then Borsellino's
                first Born approximation is used to sample triplet
                events based on the triplet cross-section data.
                [ itriplet ]
Bound Compton scattering=  On, Off, Simple or norej (default)
                If Off, Compton scattering will be treated with
                Klein-Nishina, with On Compton scattering is
                treated in the Impulse approximation.
                With Simple, the impulse approximation incoherent
                scattering function will be used (i.e., no Doppler
                broadening). With norej the actual total bound
                Compton cross section is used and there are no
                rejections at run time.
                Make sure to use for low energy applications,
                not necessary above, say, 1 MeV.
                [ IBCMP ]
Compton cross sections= Bound Compton cross-section data.  User-
                supplied bound Compton cross-sections in the file
                $HEN_HOUSE/data/comp_xsections_compton.data, where
                comp_xsections is the name supplied for this input.
                This is only used if Bound Compton scattering= Simple
```

```
                and is not available on a region-by-region basis
                (see below).  The default file (ie in the absence
                of any user-supplied data) is compton_sigma.data.
                [ comp_xsections ]
Radiative Compton corrections= On or Off (default). If on, then
                include radiative corrections for Compton scattering.
                Equations are based on original Brown & Feynman
                equations (Phys. Rev. 85, p 231--1952).  Requires
                a change to the user codes Makefile to include
                $(EGS_SOURCEDIR)rad_compton1.mortran in the
                SOURCES (just before
                $(EGS_SOURCEDIR)get_inputs.mortran).
                [ radc_flag ]

Pair angular sampling= Off, Simple or KM (Default is Simple)
                If off, pairs are set in motion at an angle m/E
                relative to the photon direction (m is electron rest
                energy, E the photon energy). Simple turns on
                the leading term of the angular distribution
                (this is sufficient for most applications),
                KM (comes from Koch and Motz) turns on using 2BS
                from the article by Koch and Motz.
                Always use Simple or KM.
                [ IPRDST ]
Pair cross sections= BH (default) or NRC.  If set to BH, then use
                Bethe-Heitler pair production cross-sections.  If set
                to NRC, then use NRC pair production cross-sections
                (in file $HEN_HOUSE/data/pair_nrc1.data).  Only
                of interest at low energies, where the NRC cross-
                sections take into account the asymmetry in the
                positron-electron energy distribution.
                [ pair_nrc ]
Photoelectron angular sampling= Off or On (Default is On)
                If Off, photo-electrons get the direction of the
                'mother' photon, with On, Sauter's furmula is
                used (which is, strictly speaking, valid only for
                K-shell photo-absorption).
                If the user has a better approach, replace the macro
                    $SELECT-PHOTOELECTRON-DIRECTION;
                The only application that
                Only situation encountered where this made a
                small difference was a big ion chamber (cavity size
                comparable with electron range) with high-Z walls
                in a low energy photon beam.
                [ IPHTER ]
Rayleigh scattering= Off, On, custom
                If On, turn on coherent (Rayleigh) scattering.
                Default is On. Should be turned on for low energy
                applications.
```

```
                If custom, user must provide media names and form
                factor files for each desired medium. For the rest
                of the media, default atomic FF are used.
                [ IRAYLR ]
ff media names = A list of media names (must match media found in
                PEGS4 data file) for which the user is going to
                provide custom Rayleigh form factor data.
                [ iray_ff_media($MXMED) ]
ff file names = A list of names of files containing the Rayleigh
                form factor data for the media specified by
                the ff media names = input above.  Full directory
                paths must be given for all files, and for each medium
                specified, iray_ff_media(i), there must be a
                corresponding file name, iray_ff_file(i).  For
                example files, see the directory
                $HEN_HOUSE/data/molecular_form_factors.
                [ iray_ff_file($MXMED) ]
Atomic relaxations= Off, On, eadl, simple
                 Default is eadl.  On defaults to eadl.
                 When simulating atomic relaxations:
                 - In photo-electric absorption events, the element
                   (if material is mixture) and the shell the photon
                   is interacting with are sampled from the
                   appropriate cross sections
                 - Shell vacancies created in photoelectric,
                   compton and electron impact ionization events
                   are relaxed via emission of fluorescent X-Rays,
                   Auger and Koster-Cronig electrons.
                  The eadl option features a more accurate treatment
                  of relaxation events and uses binding energies
                  consistent with those in of the photon cross
                  sections used in the simulation.  If using
                  mcdf-xcom or mcdf-epdl photon cross sections, you
                  cannot use the simple option and this will
                  automatically get reset to eadl.
                  Make sure to use eadl or simple for low energy
                  applications.
                  [ IEDGFL ]

Electron Impact Ionization= Off (default), On, casnati, kolbenstvedt,
                gryzinski, penelope.  If set to On or ik, then use
                Kawrakow's theory to derive EII cross-sections.
                If set to casnati, then
                use the cross-sections of Casnati (contained in the
                file ($HEN_HOUSE/data/eii_casnati.data).  Similar for
                kolbenstvedt, gryzinski and penelope. This is only of
                interest in kV X-ray calculations.
                Case-sensitive except for Off, On or ik options.
                [ eii_flag ]
```

```
Photon cross sections= Photon cross-section data.  Current options are
                si (Storm-Israel), epdl (Evaluated Photon Data
                Library), xcom (the default), pegs4, mcdf-xcom and
                mcdf-epdl:
                Allows the use of photon cross-sections other than
                from the PEGS4 file (unless the pegs4 option is
                specified).  Options mcdf-xcom and mcdf-epdl use
                Sabbatucci and Salvat's renormalized photoelectric
                cross sections with either xcom or epdl for all other
                cross sections.  These are more accurate but can
                increase CPU time by up to 6 %.
                Note that the user can supply their own cross-section
                data as well. The requirement is that the files
                photon_xsections_photo.data,
                photon_xsections_pair.data,
                photon_xsections_triplet.data, and
                photon_xsections_rayleigh.data exist in the
                $HEN_HOUSE/data directory, where photon_xsections
                is the name specified.
                Hence this entry is case-sensitive.
                [ photon_xsections ]
Photon cross-sections output= Off (default) or On.  If On, then
                a file $EGS_HOME/user_code/inputfile.xsections is
                output containing photon cross-section data used.
                [ xsec_out ]
Photonuclear attenuation= Off (default) or On
                If On, models the photonuclear effect. Current
                implementation is crude. Available on a
                region-by-region basis (see below)
                [ IPHOTONUCR ]
Photonuclear cross sections= Total photonuclear cross sections. User-
                supplied total photonuclear cross-sections in
                $HEN_HOUSE/data/photonuc_xsections_photonuc.data,
                where photonuc_xsections is the name supplied for
                this input (case sensitive). In the absence of
                any user-supplied data, or if photonuc_xsections
                is set to 'default', the default file is
                iaea_photonuc.data.
                [ photonuc_xsections ]

Atomic relaxations, Rayleigh scattering, Photoelectron angular
sampling, Bound Compton scattering and Photonuclear attenuation
can also be turned On/Off on a region-by-region basis.
To do so, put e.g.

Atomic relaxations= On in Regions   or
Atomic relaxations= Off in regions
                 in your input file. Then use the relevant one of:
Relaxations start region=
```

```
      Relaxations stop region=
               or
      Bound Compton start region=
      Bound Compton stop region=
               or
      Rayleigh start region=
      Rayleigh stop region=
               or
      PE sampling start region=
      PE sampling stop region=
               or
      Photonuclear start region=
      Photonuclear stop region=

            each followed by a list of one or more
            start and stop regions separated by commas.
       Example:
       Atomic relaxations= On in Regions
       Relaxations start region=  1, 40
       Relaxations stop region=  10, 99
                        will first turn off relaxations everywhere and
                        then turn on in regions 1-10 and 40-99.
                        Note that input is checked against min. and max.
                        region number and ignored if
                        start region < 1 or stop_region > $MXREG or
                        start region > stop region.


 ------------------------------------------------------------------------

                          Rejection Plane Inputs
                          **********************

 Used to define a rejection plane for use in conjunction with directional
 bremsstrahlung splitting (DBS, IBRSPL=2, see above).
 Inputs can exist without IBRSPL=2, but they will not be used.

 Inputs must appear between the delimiters:
 :Start DBS rejection plane:
 :Stop DBS rejection plane:

 Inputs are:

    Use a rejection plane= Off, On (default is Off)
                           Set to On if you want to define a rejection
                           plane.
                           [USE_REJPLN]
    Z(cm) from zero reference plane= Z position of reference plane.
                           [Z_REJPLN]
```

```
   Fat photons and electrons will be discarded if they are about
   to interact at Z>=Z_REJPLN.  Used to prevent correlated particles
   from being created close to a scoring plane, compromising
   statistics.

-------------------------------------------------------------------------

        Bremsstrahlung Cross Section Enhancement (BCSE) Inputs
        ******************************************************

Inputs for the BCSE variance reduction technique.

Inputs must appear between delimiters:
:Start BCSE:
:Stop BCSE:

Inputs are:

    Use BCSE= Off, On (default is Off)
            Set to On to use BCSE.
            [USE_BCSE]
    Media to enhance= A list of media in
            which to enhance the bremsstrahlung cross-section.  If
            none of the media is found in the accelerator, then no
            BCSE is done.
            [is_bcse_medium]
    Enhancement constant= Floating point factor by which bremsstrahlung
            cross-sections are enhanced. Typical values are in the range
            20 (megavoltage accelerators) -- 500 (x-ray tubes in
            mammography energy range).
            [BCSE_FACTOR_C]
    Enhancement power= Floating point number that us used for an energy
            dependent BCSE. If this input is <=0, then a constant
            BCSE factor is used that is given by BCSE_FACTOR_C. But if
            this input is >0, then the BCSE factor is computed on-th-fly
            using 1 + BCSE_FACTOR_C*(E(np)-rm)**BCSE_POWER_N.
            Typical values for BCSE_POWER_N are 2...4. Note that
            BCSE_FACTOR_C must be adjusted accordingly so that the above
            equations gives a factor of 20 (megavoltage accelerators) --
            500 (low energy x-ray tubes) for the maximum energy of the
            incident electron spectrum.
            If in doubt, just set to <=0 because the gain from an
            energy dependent BCSE is modest (~20%).

Note that if BCSE is used in conjunction with uniform bremsstrahlung
splitting (UBS) then
Russian Roulette is automatically turned on (IRRLTT=2--see above).
```

```
BCSE is most efficient when used in conjunction with DBS or UBS


------------------------------------------------------------------------

                        CUSTOMIZED USER INPUTS
                        **********************

This section contains inputs that are unique to the user.  In general
the user must modify beamnrc.mortran to read these inputs from the
.egsinp file.  Also, the GUI will not give access to these inputs.
If no custom inputs are required, then this section can be omitted
entirely.

Custom user inputs must appear in the .egsinp file between the delimiters:
     :Start user inputs:
     :Stop user inputs:

This section can appear either just before or just after the EGSnrc
inputs (see above).

Currently, the only custom user input hard-coded in beamnrc.mortran
is:

 PHSP OUTPUT DIRECTORY= /full directory path to where phase space files
                         are to be output, or blank
           This allows the user to specify a directory other than
           $EGS_HOME/BEAM_accelname in which to write phase space data.
           Useful if phase space files are large and, due to disk space
           limitations, must be written to a /temp area.  If left blank
           or omitted entirely, then phase space files are output to the
           default $EGS_HOME/BEAM_accelname directory.
           [ PHSP_OUTDIR ]

************************************************************************
```

3.1 Sample input files

Complete input files for simulation of a 10 MeV electron beam and a 16 MV photon beam can be found in the appropriate subdirectories of:
`$OMEGA_HOME/beamnrc/BEAMnrc_examples` in files `EX10MeVe.egsinp` and `EX16MVp.egsinp` respectively. The file `EXphantom.egsinp` gives an example of using BEAMnrc to do an efficient depth-dose calculation (but only for the central axis). These examples are described in separate reports[21, 22, 23] which are somewhat dated since one should now use the GUIs to run these examples.

The auxiliary script `test_BEAMnrc` (found on `$HEN_HOUSE/scripts`) automatically create the needed subdirectories on the user's area (`BEAM_EX10MeVe` *etc.*) and copy the relevant files

to them so that the user may execute BEAMnrc immediately (but the script only works on Linux/Unix and there is no corresponding replacement for Windows).

4 Source Routines

In general, the incident particles move in the direction of the z-axis. With the exception of ISOURC=3 (internal isotropic source), =10 and 13 (x-ray tube sources) or =21, 23, 24, 31 (phase space source, BEAMnrc simulation source, or source model), the particles start being transported on the `Z_min_CM(1)` plane. Conceptually, some of them originate at a point outside the accelerator model and are essentially transported through vacuum to the accelerator which starts at the `Z_min_CM(1)` plane.

One of the major features of BEAMnrc is that a phase space file can be used as a source file between any two `CMs` in the accelerator. By writing a suitable off-line routine to generate the appropriate phase space file (use the macros in `phsp_macros.mortran`), virtually any source can be simulated. This feature also allows the accelerator simulation to be broken up into components. For example, one might simulate the fixed components at the top of an accelerator and then feed the phase space file into a variety of calculations with different jaw or applicator settings.

For historic and compatibility reasons, the input to specify the source type and the charge of the incident beam are on a single line (as outlined in section 3). In particular, note that the first variable is the charge, `IQIN`, or a dummy in those cases where the charge is determined elsewhere (*eg.* for a phase space file input).

The following subsections describe the source routines available in BEAMnrc.

ISOURC=0 Parallel Circular Beam

ISOURC=1 Isotropic Point Source on Z-axis

ISOURC=3 Interior Isotropic Cylindrical Source

ISOURC=5 NRC Swept Beam

ISOURC=6 Parallel Rectangular Beam

ISOURC=7 Scanning Beam (sawtooth)

ISOURC=8 Scanning Point Source for MM50

ISOURC=9 Discrete Point Source for MM50

ISOURC=10 Parallel Circular Beam Incident from Side

ISOURC=13 Parallel Rectangular Beam Incident from Side

ISOURC=15 NRC Swept Beam with Radial Intensity Distribution and Radial Divergence

ISOURC=19 Elliptical Beam with Gaussian Distributions in X and Y, Parallel or with Radial Divergence

ISOURC=21 Phase Space Source

ISOURC=23 BEAMnrc Simulation Source

ISOURC=24 Phase Space Source Incident from User-specified Angle

ISOURC=31 Beam characterization Model

4.1 ISOURC=0: Parallel Circular Beam

IQIN, 0, RBEAM, UINC, VINC, WINC

The parallel circular beam is always assumed to be incident on the center of the front of the first CM (*i.e.* at `Z_min_CM(1)`). The input parameters are:

IQIN The charge of the incident beam (defaults to 0=photons)

RBEAM Radius of the beam in cm (maximum is `RMAX_CM(1)` for a circular first CM or $\sqrt{2}$`RMAX_CM(1)` for a square first CM, it defaults to maximum if set > maximum)

UINC X-axis direction cosine (defaults to 0)

VINC Y-axis direction cosine (defaults to 0)

WINC Z-axis direction cosine (defaults to 1, *i.e.* parallel to the z-axis)

UINC, VINC, and WINC are automatically normalized by (UINC^2+VINC^2+WINC^2). Note that it is possible to have `RBEAM` = 0; this is a pencil beam. Figure 6 below shows the parallel circular beam and its input parameters.

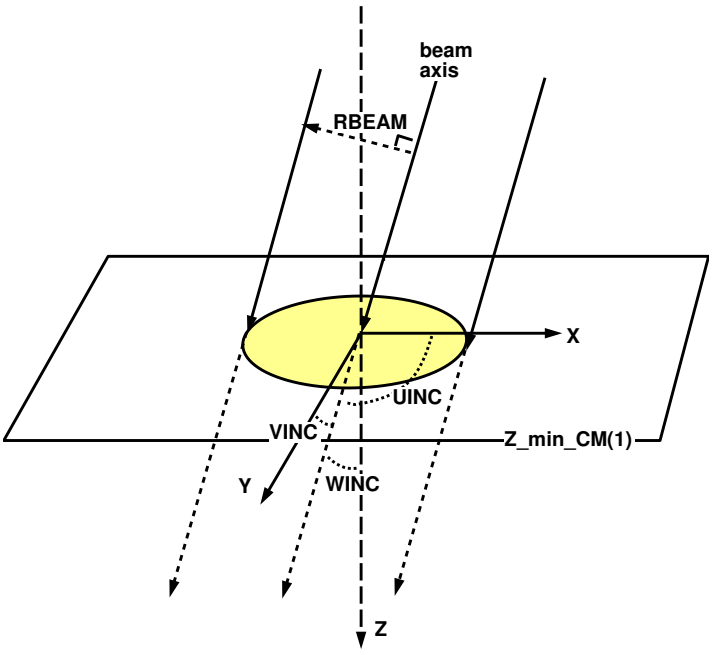


Figure 6: Parallel Circular Beam (ISOURC=0) showing the beam radius, `RBEAM`, and the direction cosines, UINC, VINC and WINC. Note that `RBEAM` is measured perpendicular to the beam central axis. The beam axis always intersects `Z_min_CM(1)` at (X=0,Y=0).

4.2 ISOURC=1: Isotropic Point Source on Z-axis

IQIN, 1, DISTZ, RBEAM, GAMMA, XINL, XINU, YINL, YINU

The isotropic point source is placed on the Z-axis. It is assumed to be incident on the first CM, but can be placed any distance above `Z_min_CM(1)`. The beam field can be circular or rectangular (see below). The circular beam is centred on the Z axis, while the rectangular beam can be placed anywhere on the top surface of CM 1.

DISTZ Distance of the source above `Z_min_CM(1)` in cm (defaults to 100 cm)

RBEAM Radius of circular field at `Z_min_CM(1)` (if > 0). If `RBEAM`=0, then `GAMMA` (see below) is used to define the circular field. If `RBEAM`<0, then `XINL`, `XINU`, `YINL`, `YINU` (see below) are used to define a rectangular field. Maximum value of `RBEAM` is `RMAX_CM(1)` (circular CM 1) or $\sqrt{2}$`RMAX_CM(1)` (square CM 1). `RBEAM` defaults to maximum if it is set $>$ maximum or if `RBEAM`=0.0 and `GAMMA` = 0.0.

GAMMA Half-angle of circular field at `Z_min_CM(1)` relative to the point source (Note that this is simply another way of specifying the radius of a circular field, and is only used if `RBEAM` = 0.0; defaults to give maximum radius [see description of `RBEAM` above] if it is set so that the radius would be $>$ maximum).

XINL, XINU, YINL, YINU Min. X, Max. X, Min. Y, Max Y dimensions of rectangular field (only used if `RBEAM`<0) in cm. The rectangular field is restricted to -`RMAX_CM(1)`$\leq$X$\leq$`RMAX_CM(1)` and -`RMAX_CM(1)`$\leq$Y$\leq$`RMAX_CM(1)`. The dimensions can be set to give a line or point "field" anywhere on the top of CM 1.

`DISTZ`, between the source and `Z_min_CM(1)` is assumed to be a vacuum. Figure 7 below illustrates the point source and its input variables.

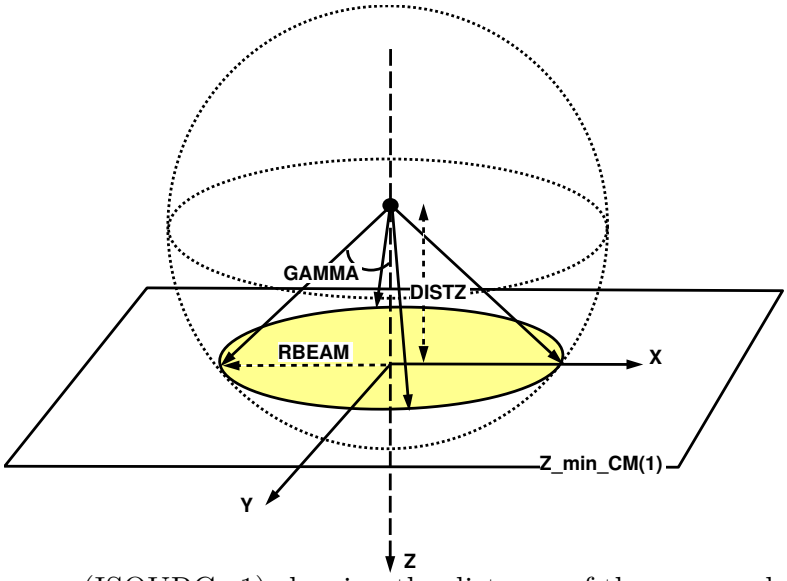


Figure 7: Point source (ISOURC=1) showing the distance of the source above `Z_min_CM(1)`, specified by `DISTZ`, and the two different ways of specifying a circular field at `Z_min_CM(1)`, `RBEAM` and `GAMMA`. A rectangular beam anywhere on the top surface of CM 1 can be specified by setting `RBEAM`<0 and using `XINL`, `XINU`, `YINL`, `YINU` to define the beam.

4.3 ISOURC=3: Interior Isotropic Cylindrical Source

IQIN, 3, RMINBM, RBEAM, ZSMIN, ZSMAX

The uniform radiating cylindrical isotropic source can be a ring centred on the Z-axis or a cylinder with its central axis parallel to the X-axis. The orientation is controlled by the sign of the input variable, `RBEAM` (see below). The source is contained within the geometry of one or more component modules (*i.e.* it is able to span adjacent CMs). Currently, this source can only be used inside `CONESTAK, SIDETUBE or FLATFILT` CMs because of the need to identify the initial regions. The input parameters for this source are:

RMINBM Inner radius of vertical ring (if `RBEAM` ≥ 0) *or* Z position of centre of horizontal cylinder (if `RBEAM` < 0)

RBEAM Outer radius of vertical ring (if ≥ 0) *or* negative radius of horizontal cylinder (if $<$ 0) (for vertical ring, maximum value is `RMAX_CM` of largest CM containing the source; for horizontal cylinder, maximum value keeps the radius entirely contained between `Z_min_CM(1)` and the bottom of the geometry; defaults to maximum if absolute value is $>$ maximum)

ZSMIN Z of top of vertical ring (if `RBEAM` ≥ 0) *or* minimum X of horizontal cylinder (if `RBEAM` < 0) (for vertical ring, minimum value is `Z_min_CM(1)`; for horizontal cylinder, minimum is `-RMAX_CM` of largest CM containing source; defaults to minimum if set $<$ minimum)

ZSMAX Z of bottom of vertical ring (if `RBEAM` ≥ 0) *or* maximum X of horizontal cylinder (if `RBEAM` < 0) in cm (for vertical ring, maximum is the bottom of geometry; for horizontal cylinder, maximum is `RMAX_CM` of largest CM containing source; defaults to maximum if set $>$ maximum)

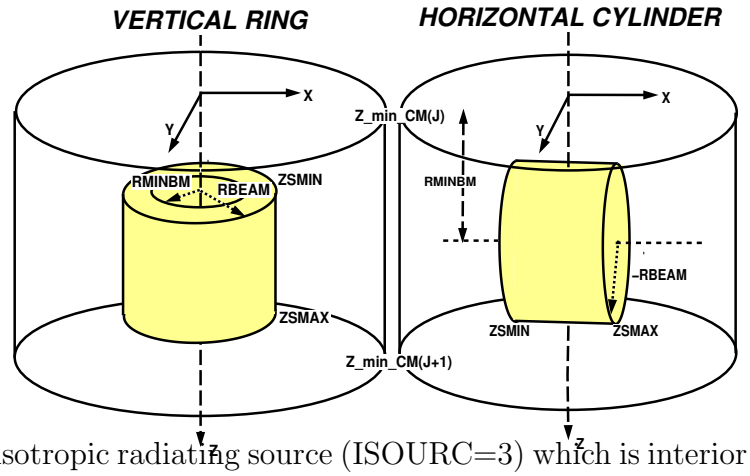


Figure 8: Uniform isotropic radiating source (ISOURC=3) which is interior to the accelerator model. For vertical ring configuration shown on the left, set `RBEAM` ≥ 0. For horizontal cylinder configuration on right set `RBEAM` < 0. The meaning of the input variables depends on the configuration. The source is shown within a single CM, however it can span any number of adjacent CMs.

Source 3 also has an option to split photons that are emitted into a user specified treatment field. This option, called directional source biasing (DSB), can increase the efficiency of Cobalt-60 treatment head simulations by factors on the order of 100 and, in general, can be used to increase the efficiency of any photon beam with an isotropically emitting source. A detailed explanation of DSB is given in section 6.4.

4.4 ISOURC=5: NRC Swept BEAM

IQIN, 5, GAMMA, RBEAM

The NRC swept beam is a parallel circular beam swept around the outside of an imaginary cone. The beam is assumed incident on the front of CM 1. The apex of the imaginary cone is always at (X=0,Y=0,Z=`Z_min_CM(1)`), and the cone angle is variable. Input parameters for this source are:

IQIN Charge of the incident beam (defaults to 0 = photons)

GAMMA Half-angle of the cone in degrees ($0^o \leq$ `GAMMA` $< 90^o$)

RBEAM Radius of the beam in cm (maximum value `RMAX_CM(1)` if the first CM is circular or $\sqrt{2}$`RMAX_CM(1)` if the first CM is square, defaults to maximum if set > maximum)

Figure 9 below shows the swept beam source and its input parameters.

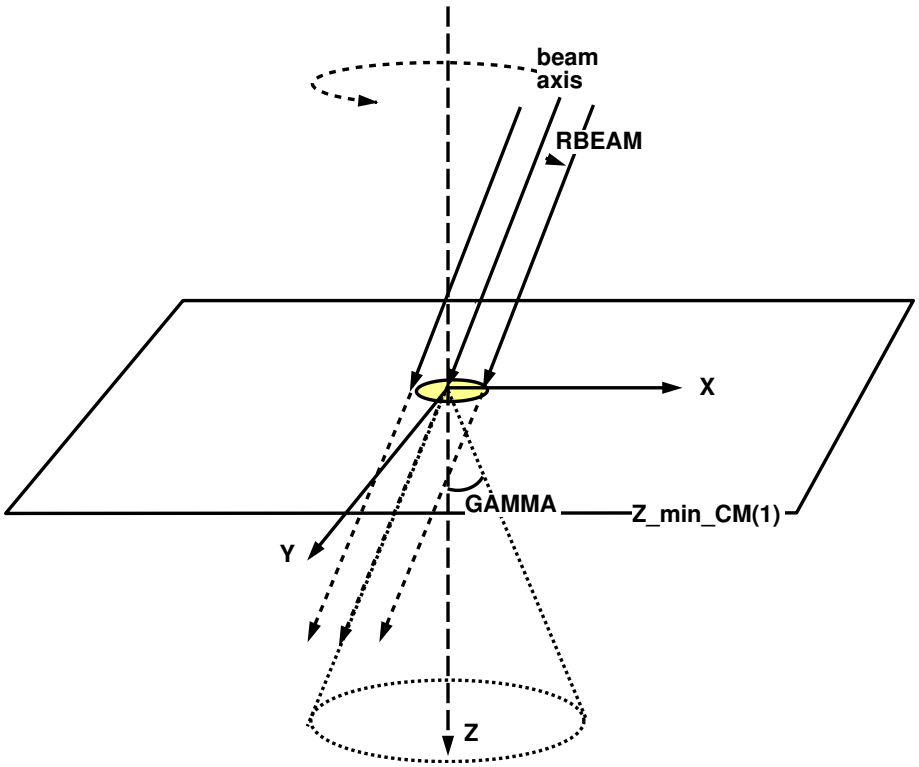


Figure 9: NRC swept beam source (ISOURC=5) showing the incident beam, with radius `RBEAM`, sweeping the surface of an imaginary cone with apex half-angle `GAMMA`. Note that it is the beam axis which actually sweeps the surface of the cone.

4.5 ISOURC=6: Parallel Rectangular Beam

IQIN, 6, XBEAM0, YBEAM0, XBEAM, YBEAM

The parallel rectangular beam is the rectangular equivalent of the parallel circular beam (ISOURC=0) with the exception that the beam cannot be oblique, it is always perpendicular to the front surface of the first CM and this beam can be offset from the Z-axis. Input parameters for the parallel rectangular beam are:

IQIN Charge of beam (defaults to 0 = photons)

XBEAM0 X position of beam centre in cm (defaults to 0 if $\texttt{XBEAM0}^2 + \texttt{YBEAM0}^2 > \texttt{RMAX\_CM(1)}^2$ when the first CM is circular or if $|\texttt{XBEAM0}|$ or $|\texttt{YBEAM0}| > \texttt{RMAX\_CM(1)}$ when the first CM is a square)

YBEAM0 Y position of beam centre in cm (defaults to 0 under same conditions as XBEAM0)

XBEAM Half-width of beam in X-direction in cm (defaults to keep $|\texttt{XBEAM0}| + |\texttt{XBEAM}| \leq \texttt{RMAX\_CM(1)}$)

YBEAM Half-width of beam in Y-direction in cm (defaults to keep $|\texttt{YBEAM0}| + |\texttt{YBEAM}| \leq \texttt{RMAX\_CM(1)}$

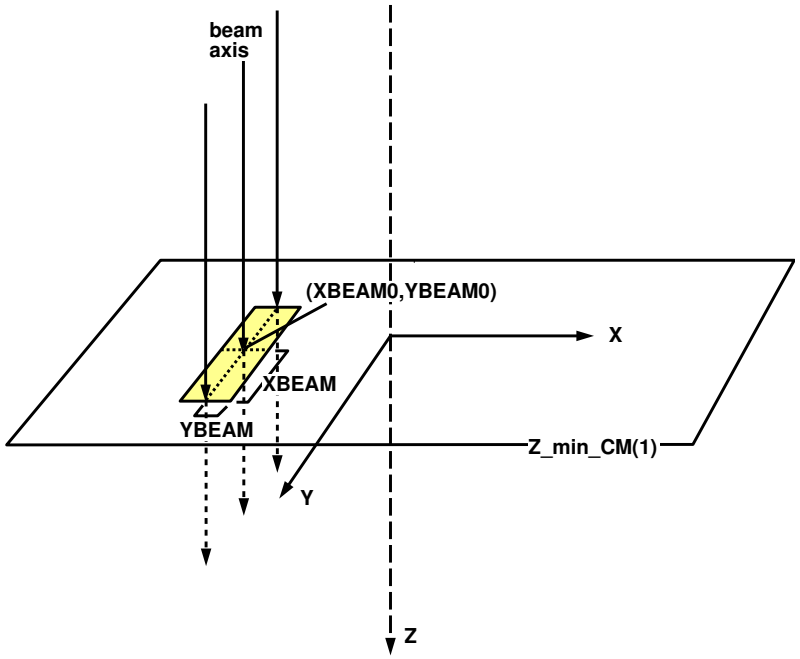


Figure 10: Parallel rectangular beam (ISOURC=6) showing the beam centre (XBEAM0,YBEAM0) and the X and Y half-widths (XBEAM,YBEAM).

4.6 ISOURC=7: Scanning Sawtooth Beam

IQIN, 7, FD_AT100, IRATIO_YXF, RBEAM

In the scanning sawtooth beam source, a circular parallel beam, incident on the front of the first CM, is scanned in a zig-zag pattern on the X-Y plane. The angle for the scan is selected as the particles leave the `Z_min_CM(1)` plane. This is an approximation of the real case where the beam is bent below this point usually. The source randomly selects points in the scan. The input parameters are:

IQIN Charge of the incident beam (defaults to 0 = photons)

FD_AT100 Scanning field size at SSD=100cm in cm (field is `FD_AT100` x `FD_AT100`). This is often much greater than the actual field size.

IRATIO_YXF The number of Y scans per X scan (defaults to 6.5 if set $\leq$ 0; also, 2x`IRATIO_YXF` is always rounded up to the nearest odd integer)

RBEAM Radius of the incident beam in cm (defaults to 0.01 cm if set $\leq$ 0; defaults to maximum of `RMAX_CM(1)` [circular CM 1] or $\sqrt{2}$`RMAX_CM(1)` [square CM 1] if set > maximum)

Note that the field is always scanned twice in the X direction; thus, the number of Y scans is 2x`IRATIO_YXF`. It is also important to note that `FD_AT100` defines the field size covered by the beam central axis; a beam with finite `RBEAM` will actually go outside `FD_AT100`. Figure 11 below shows the scanning beam and its various input parameters.

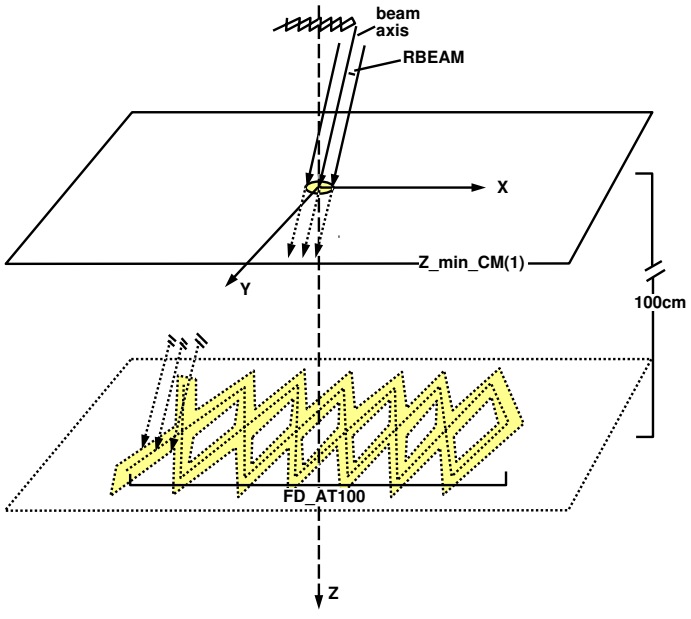


Figure 11: Scanning beam (ISOURC=7). The parallel circular beam of radius `RBEAM` is scanned in a zig-zag pattern in the X-Y plane to produce the zig-zag coverage at SSD=100cm. In the case shown here, `IRATIO_YXF` = 6.5; hence there are a total of 13 Y scans.

4.7 ISOURC=8: Scanned Point Source for MM50–Uniform Field

IQIN, 8, DISTZ, RBEAM, RBEAM0

This source is designed to simulate the source used in the MM50 linear accelerator for electron beams. The source behaves like a point source located at Z=0 scanned to produce a uniform distribution of particles at a user-specified distance from the source (`DISTZ`) over a field of user-specified radius (`RBEAM`). Uncertainty in the X-Y position of the source at Z=0 can be simulated using the beam spot radius, `RBEAM0`. If `RBEAM0` > 0, the particle distribution at `DISTZ` will be uniform out to `RBEAM` and tails off between `RBEAM` and `RBEAM + RBEAM0`. Note that `Z_min_CM(1)` must be ≥ 0 for this source. Input parameters are:

IQIN Charge of the incident beam (defaults to 0 = photon)

DISTZ The source-to-surface distance (SSD) at which uniform particle distribution is desired in cm (defaults to 100 cm if it is set ≤ 0)

RBEAM Radius of the field at `DISTZ` in cm (defaults to maximum value of `RMAX_CM(1)*DISTZ/Z_min_CM(1)` for circular CM 1 or $\sqrt{2}$`RMAX_CM(1)*DISTZ/Z_min_CM(1)` for square CM 1)

RBEAM0 Radius of the beam spot at Z=0 in cm (defaults to 0 if set < 0 and gets reset to max. value of `RBEAM - RBEAM` if `RBEAM+RBEAM0` $>$ max. value of `RBEAM`)

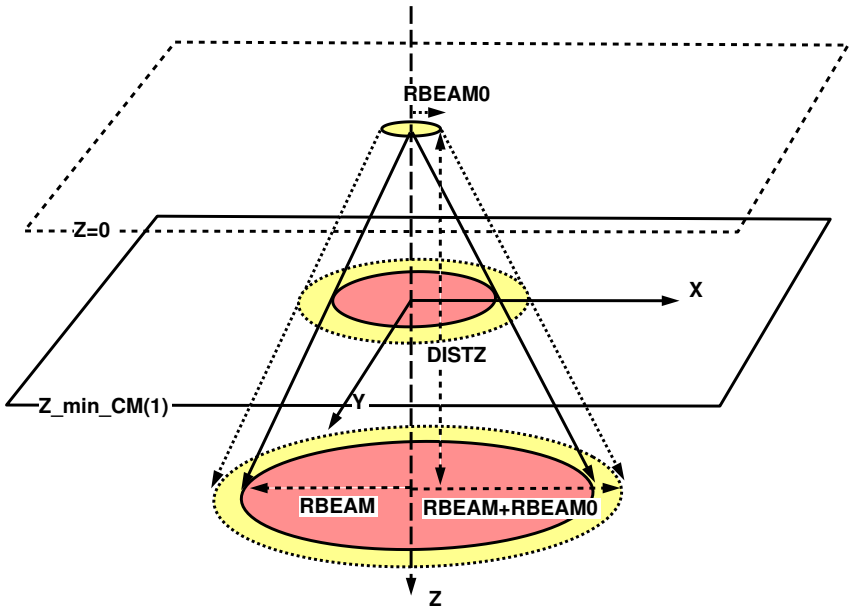


Figure 12: Scanned point source producing uniform field coverage (ISOURC=8). The figure shows a point source at Z=0 scanned to give uniform particle distribution in a field of radius `RBEAM` at a distance `DISTZ` below the source. The uncertainty of the position of the source at Z=0, is simulated using `RBEAM0`. Particles can originate anywhere within the circle defined by `RBEAM0` which results in a particle distribution at `DISTZ` which is uniform out to `RBEAM` and falls off between `RBEAM` and `RBEAM+RBEAM0`.

4.8 ISOURC=9: Scanned Point Source for MM50–Discrete Field

IQIN, 9, DISTZ, NPTS_SRC9
X_SRC9, Y_SRC9, PROB_SRC9

This source is designed to simulate the source used in the MM50 linear accelerator to produce photon beams. It behaves like a point source emanating pencil beams to a finite number of dwell points, providing discrete field coverage. The user specifies by the X,Y coordinates of the dwell points on a plane perpendicular to the Z-axis at a user-specified source-to-surface distance (SSD) and the probability of a particle being initially directed towards each point. The source is always considered to be at Z=0. Input parameters are:

IQIN Charge of the incident beam (defaults to 0 = photon)

DISTZ The source-to-surface distance (SSD) in cm (defaults to 100 cm if it is set ≤ 0)

NPTS_SRC9 The number of points used to cover the field (defaults to 1 if set ≤ 0)

X_SRC9(I) (I=1,...,NPTS_SRC9) The X coordinate of point I on plane $\perp$ Z at `DISTZ` in cm

Y_SRC9(I) (I=1,...,NPTS_SRC9) The Y coordinate of point I on plane $\perp$ Z at `DISTZ` in cm

PROB_SRC9(I) (I=1,...,NPTS_SRC9) The probability of a particle being incident on point I. Probabilities are automatically normalized.

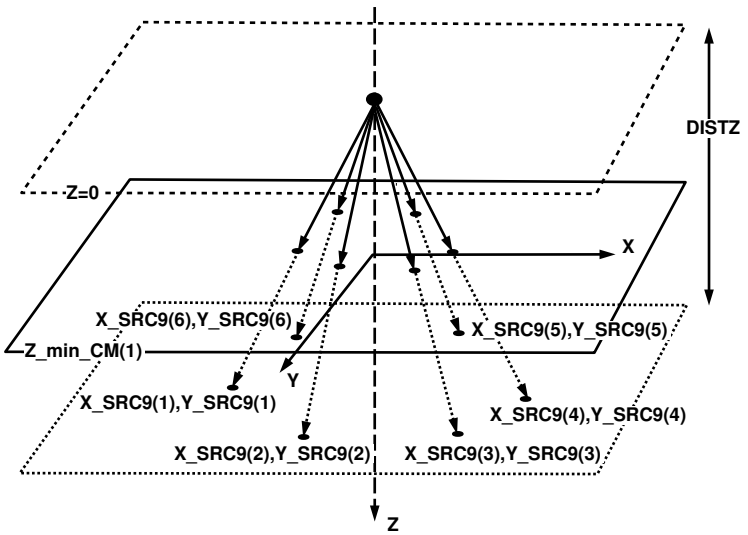


Figure 13: Scanned point source for discrete field coverage (ISOURC=9). The figure shows a point source at Z=0 producing 6 points scattered over the field (`NPTS_SRC9`=6). The user specifies the source-to-surface distance, `DISTZ`, the X and Y coordinates of the points at `DISTZ`, `(X_SRC9(I),Y_SRC9(I))`, and the probability of a particle being incident at each point, `PROB_SRC9(I)` (not shown).

4.9 ISOURC=10: Parallel Circular Beam Incident from Side

IQIN, 10, RBEAM, UINC, VINC, WINC

This parallel circular beam enters the first CM from the side and can only be used as the source for an XTUBE. The input parameters for this source are:

IQIN Charge of incident beam (defaults to 0 = photons)

RBEAM Radius of beam in cm (defaults to half the thickness of the first CM if it is greater than this and to 0 if it is set < 0)

UINC Incident X-axis direction cosine (should be < 0 so the beam faces the Z-axis; reset to -UINC if it is > 0)

VINC Incident Y-axis direction cosine

WINC Incident Z-axis direction cosine
(UINC, VINC, WINC default to -1,0,0 if $UINC^2+VINC^2+WINC^2 = 0$)

The direction cosines UINC, VINC and WINC are each automatically normalized by $(UINC^2+VINC^2+WINC^2)$.

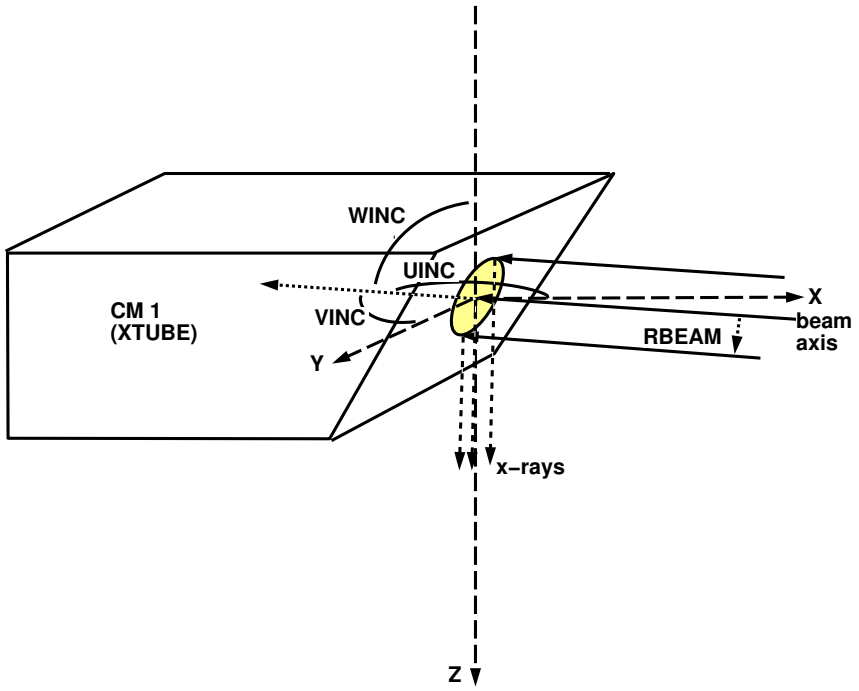


Figure 14: Parallel circular beam incident from the side (ISOURC=10). This source can only be used with an XTUBE CM as the first CM position.

4.10 ISOURC=13: Parallel Rectangular Beam Incident from Side

IQIN, 13, YBEAM, ZBEAM, UINC, VINC

This beam is the rectangular equivalent of ISOURC=10 (see previous source). The input parameters for this beam are:

IQIN Charge of incident beam (defaults to 0 = photons)

YBEAM Half-width of beam in cm (defaults to 0.2 cm if set < 0; defaults to half the thickness of the first CM if set $>$ than this)

ZBEAM Half-height of beam in cm (defaults to 0.2 cm if set < 0; defaults to half the thickness of the first CM if set $>$ than this)

UINC Incident X-axis direction cosine (should be <0 so that beam faces Z-axis; reset to -UINC if it is set > 0)

VINC Incident Y-axis direction cosine
(UINC, VINC default to -1,0 if $\text{UINC}^2+\text{VINC}^2 = 0$)

UINC and VINC are automatically normalized by ($\text{UINC}^2+\text{VINC}^2$). Note that the Z direction cosine, WINC, is always assumed to be 0 for this beam.

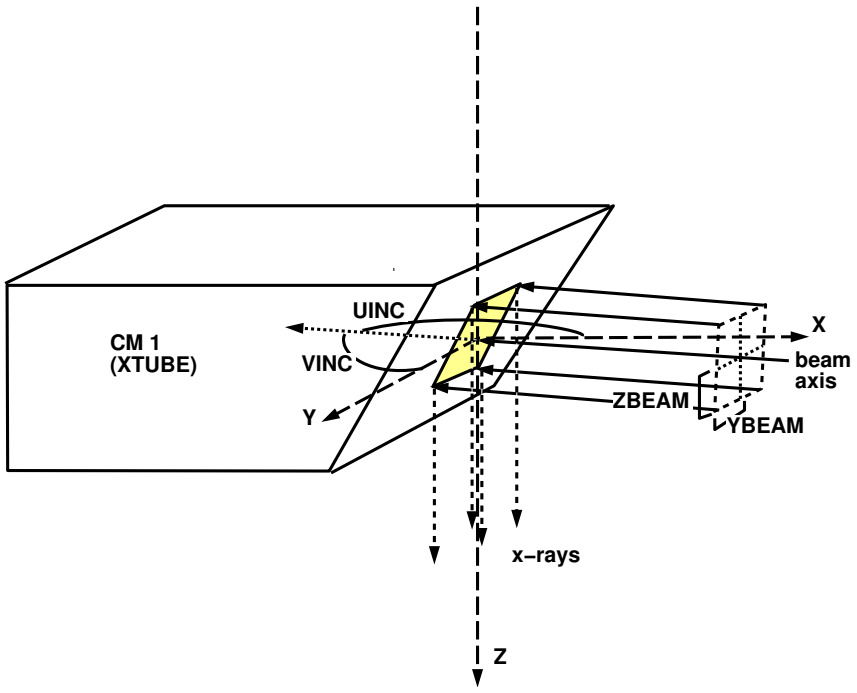


Figure 15: Parallel rectangular beam incident from the side (ISOURC=13).

4.11 ISOURC=15: NRC Swept Beam (Radial Variation, Divergence)

IQIN, 15, GAMMA, ZFOCUS, RTHETAIN, THETAIN
SPCNAM

This source is similar to the NRC Swept Beam (ISOURC = 5) with the difference that the beam itself has a radially varying intensity distribution and an angular divergence, both specified by the user. Also, the cone swept by the beam has its apex at a user specified Z position, given by (X=0,Y=0,Z=ZFOCUS), rather than the apex always being at (X=0,Y=0,Z=Z_min_CM(1)) as it is in source 5. As with most other sources, particles enter from the top of CM 1.

Beam divergence means the beam is diverging or spreading out about the axis of each element of the swept beam. At any given instant each elemental beam can be thought of diverging from a point on the elemental beam's axis. This point is specified by the angle (THETAIN) created by the elemental beam's axis and a line from the apex to a point at a specified radius (RTHETAIN) in the ZFOCUS plane. This angle's only role is to define the distance to the apex.

Input parameters for this source are:

IQIN Charge of the incident beam (defaults to 0 = photons)

GAMMA Half-angle of the cone swept by the beam in degrees ($0^o \leq$ GAMMA $< 90^o$)

ZFOCUS Z position of the apex of the swept cone in cm (Note: ZFOCUS can be greater than, equal or less than Z_min_CM(1)).

RTHETAIN The beam radius (in cm) at which the beam divergence angle, THETAIN, is defined. RTHETAIN must be > 0.

THETAIN The beam divergence angle (in degrees) at RTHETAIN. If GAMMA $\neq 0$, then THETAIN can be set to 0; otherwise, it must be > 0 (Note, however, that it can still be set to such a small angle that divergence is practically zero).

SPCNAM The name of file containing beam's radial intensity distribution about the axis of the elemental beam in a plane at Z = ZFOCUS. The file SPCNAM has a specific format:

```
NRDIST
(RDISTF(I),RPDF(I),I=1,NRDIST)
```

where:

NRDIST = # of radial bins in the distribution

RDISTF(I) = upper radius of bin I in cm

RPDF(I) = probability (not necessarily normalized) of finding a particle in radial bin I. These probabilities are multiplied by bin area before sampling.

Some sample radial intensity distribution files are found in
$OMEGA_HOME/beamnrc/radial_source_distributions.

The radial position of an incident particle, RIN, is chosen based on the radial intensity distribution. The divergence angle, THETAI, of the particle is then calculated from TAN(THETAI) = RIN/(RTHETAIN/TAN(THETAIN)). Note that the radial distribution is defined at Z = ZFOCUS (the apex of the swept cone) in a plane perpendicular to the Z axis. This is an approximation since, strictly speaking, the distribution should be defined in a plane perpendicular to the beam direction. However, for GAMMA of a few degrees, this approximation is valid.

In addition to simulating beams more realistically, this source can be applied to several useful cases. With GAMMA = 0 and THETAIN ≈0, it can be used to model an arbitrary radial intensity distribution from a parallel beam incident normally on the front face (*eg.*, the distribution could be ring shaped). With GAMMA = 0 one could mimic a point source at a distance from ZFOCUS of RTHETAIN/tan(THETAIN) along with a uniform intensity profile out to the beam radius. One should use source 1 for the uniform case, but could use source 15 to get an arbitrary radial distribution from a point source.

Figure 16 below shows the swept beam source with radial distribution and divergence.

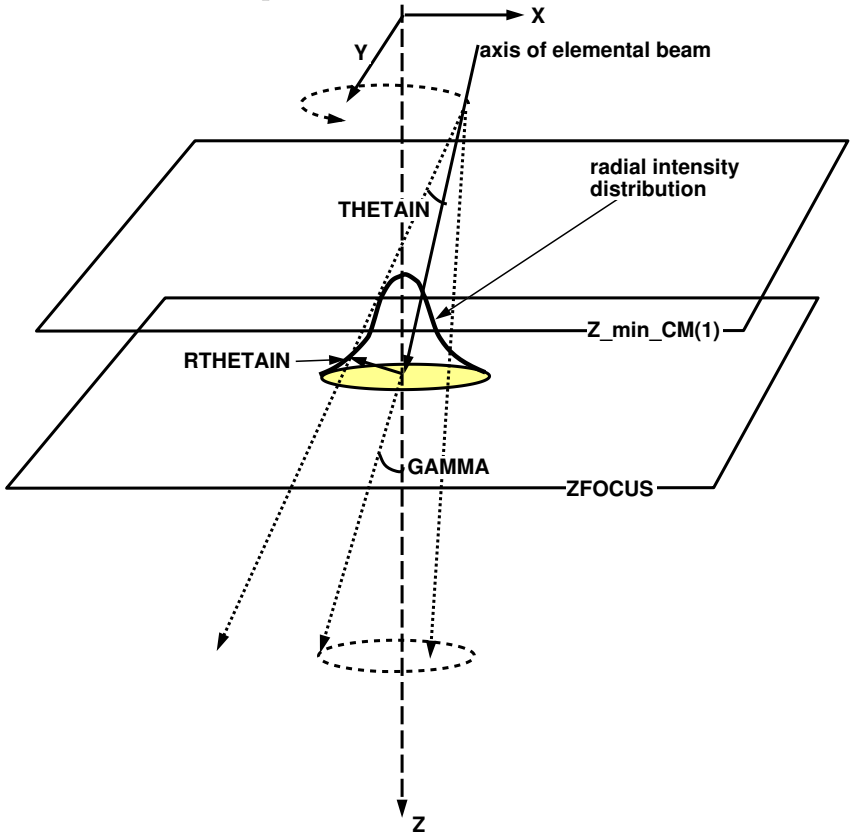


Figure 16: The beam sweeps on the surface of a cone with half-angle GAMMA and with the apex defined at ZFOCUS. The beam itself has a user-defined radial intensity distribution (in the case shown, a gaussian), defined at ZFOCUS and in a plane perpendicular to the Z axis. The beam then diverges from this original distribution as if it originated at an imaginary point back along the axis of each elemental beam. The position of this imaginary point is defined by the divergence angle, THETAIN, and the radius (at ZFOCUS) at which the divergence angle is defined, RTHETAIN.

4.12 ISOURC=19: Elliptical Beam with Gaussian Distributions in X and Y, Parallel or with Angular Spread

This is an elliptical beam where the ellipse is defined by gaussian intensity distributions in X and Y. The beam can either be parallel, with direction cosines specified by the user, or it can have an angular spread from the Z-axis, specified by a mean angular spread.

IQIN The charge of the incident beam (defaults to 0=photons)

RBEAM Standard deviation (σ) in the X-direction (if set > 0) in cm, or -FWHM (FWHM:full-width half-maximum) of the gaussian distribution in the X-direction (if set < 0) in cm. σ is automatically limited to `RMAX_CM(1)` for circular CM 1 or $\sqrt{2}$`RMAX_CM(1)` for square CM 1. If the user enters `RBEAM`=0, the source collapses to a pencil beam.

UINC X-axis direction cosine (defaults to 0)

VINC Y-axis direction cosine (defaults to 0)

WINC Z-axis direction cosine (defaults to 1, *i.e.* parallel to the z-axis)

sigma_src19 the mean angular spread about the Z-axis. If `sigma_src19`>0, then `UINC`, `VINC`, `WINC` automatically take their default values.

RBEAMY σ (if set > 0) or -FWHM of the gaussian distribution in the Y-direction in cm. If set = 0, then `RBEAMY` is set = `RBEAM`, resulting in a circular beam with a gaussian radial distribution.

Note that in the case of a circular beam (`RBEAMY`=`RBEAM`) the radial distribution of particles is also gaussian with σ equal to $\sqrt{2}$ times the σ of the X and Y gaussian distribution.

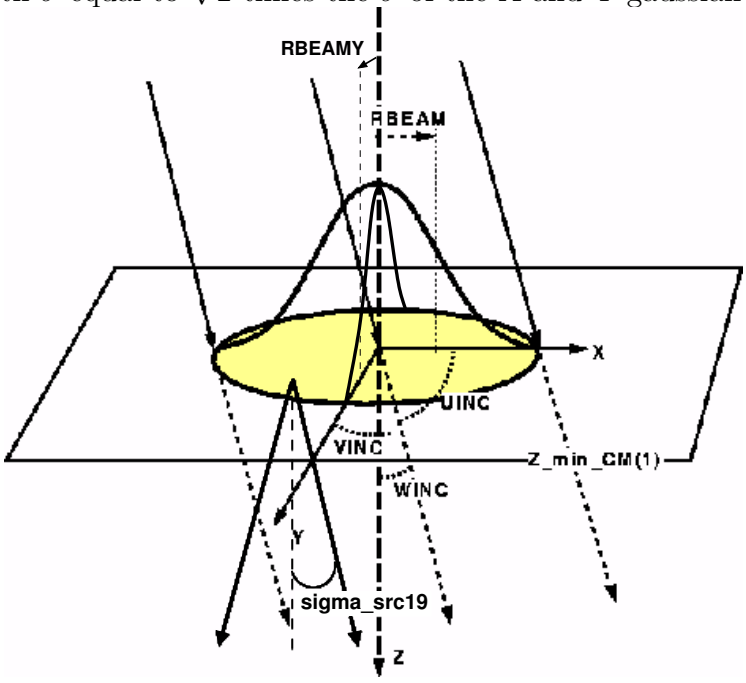


Figure 17: Elliptical Beam with Gaussian Distributions in X and Y (ISOURC=19). The shape of the ellipse is defined by `RBEAM` and `RBEAMY` which define the the σ (if set >0) or -FWHM of the gaussian intensity distributions in the X- and Y-directions respectively. The beam either has a mean angular spread about the Z-axis, `sigma_src19`, or, if `sigma_src19`$\leq$0, it is a parallel beam with incident direction cosines specified by `UINC`, `VINC`, `WINC`.

4.13 ISOURC=21: Phase Space Source

IDUMMY, 21, INIT_ICM, NRCYCL, IPARALLEL, PARNUM, ISRC_DBS, RSRC_DBS, SSDSRC_DBS, ZSRC_DBS
SPCNAM

This source routine allows a phase space file generated at any scoring plane to be used as a source in its own right. Unlike the sources described thus far, which are assumed to be incident on the first CM, phase space sources can be incident on any CM. They are particularly useful for doing repeated simulations of lower portions of an accelerator model (*i.e.* when the configuration of the lower part of the accelerator is changing, but repeated simulation of the top part is unnecessary), or simulating beams made up of several different particle types. Note that the `LATCH` values are passed on when using this source routine and it is necessary to number `LATCH` bits consistently between the two simulations (*i.e.* that generating the input file and the current simulation). Also, dose and fluence resulting from a phase space source are normalized by the number of initial particles in the original, non-phase space source (*i.e.* the source that generated the phase space source) and not by the number of particles incident from the phase space source itself. More about this normalization and how it is accomplished appears in section 7 below.

This source can also be used to simulate any incident beam shapes not covered by the sources described above. This is done by creating the appropriate phase space file for the general source by using a stand alone user-written program and the `phsp_macros.mortran` macros.

Input parameters for a phase space source are:

INIT_ICM Component module on which the phase space source is incident (defaults to 1 if it is set to < 1 or $>$ the number of `CMs` in the model)

NRCYCL The number of times to recycle each incident particles before moving on to the next particle in the phase space source. Thus, each particle will be used a total of `NRCYCL` + 1 times before moving on to the next one. If set ≤ 0, then BEAMnrc will automatically calculate a value of `NRCYCL` based on the number of incident histories and the number of particles in the phase space file that should prevent restarting of the phase space source. If set > 0, the user-input value is used. See below for details.

IPARALLEL No longer necessary if using BEAMnrc's built-in parallel processing functionality for submitting parallel jobs. In previous versions of BEAMnrc, `IPARALLEL` = the number of parallel jobs. See below for more details.

PARNUM No longer necessary if using BEAMnrc's built-in parallel processing functionality for submitting parallel jobs. In previous versions of BEAMnrc, `PARNUM` was set to an integer value in the range 1$\leq$`PARNUM`$\leq$ `IPARALLEL`, with a different value of `PARNUM` for each of the parallel jobs. See below for more details.

ISRC_DBS Set to 1 if directional bremsstrahlung splitting (DBS) was used in the BEAMnrc simulation to generate this source **and** you wish to reject photons directed outside the splitting field (which are fat). Set to 0 otherwise.

RSRC_DBS Radius of the DBS splitting field (in cm) used in the BEAMnrc simulation to generate this source. Only needed if ISRC_DBS = 1.

SSDSRC_DBS SSD at which RSRC_DBS was defined in the BEAMnrc simulation that generated this source (in cm). Only needed if ISRC_DBS = 1.

ZSRC_DBS Z value where this phase space source was scored in the BEAMnrc simulation (cm). This will be at the back of a component module (CM). Note the restriction that ZSRC_DBS is $\leq$ SSDSRC_DBS.

SPCNAM Filename (with extension) of the phase space source. The full directory path to the file must be specified. In the case of an IAEA-format phase space source, you must supply the full name of the file containing the phase space data (*i.e.* the .IAEAphsp file). The .IAEAheader file will be assumed to be in the same directory. For more information on IAEA-format phase space data, see Section 7.4.

Value of NRCYCL

NRCYCL is an essential input for phase space sources because phase space data are often sparse, making it necessary to re-use particles in order to obtain adequate statistics. However, NRCYCL can only help reduce the inherent uncertainty in the dose calculation. The overall uncertainty will also be governed by the latent variance of the phase space file (see the report on history by history statistics in BEAMnrc and DOSXYZnrc[17]). In addition to recycling, particles are also re-used whenever a phase space source is restarted (which happens automatically when the simulation has used all particles in the source). However, restarting may result in an underestimate of uncertainties [17], making it important to choose a value of NRCYCL that both ensures adequate sampling of the phase space source and also prevents restarting.

If you are unsure of the value of NRCYCL to use, set NRCYCL$\leq$ 0 and BEAMnrc will automatically calculate its value based on the number of histories and the total number of particles in the phase space file. Even with an automatically-calculated value of NRCYCL, the phase space source may still restart, either because the algorithm used to calculate NRCYCL has determined that setting NRCYCL$>$ 0 will cause the phase space source to be undersampled, or because some incident particles were multiple-passers (i.e., had crossed the phase space scoring plane more than once) and were rejected from the simulation. If the phase space source has only been restarted once and only a small fraction of it has been re-used on the second pass, the effect on uncertainty will not be significant. However, if a significant portion of the source has been re-used on the second pass, or if the source has been restarted more than once, we recommend re-running the simulation with a new value of NRCYCL given by:

$$\frac{\texttt{NCASE}}{[\texttt{NNPHSP} - \texttt{NNPHSP} * (\texttt{NPASS\_ph\_sp} + \texttt{NFAT\_ph\_sp}) / (\texttt{NTOT\_ph\_sp} + \texttt{NPASS\_ph\_sp} + \texttt{NFAT\_ph\_sp})]} - 1$$

where NCASE is the number of histories, NNPHSP is the total number of particles in the phase space source, NTOT_ph_sp is the total number of particles used from the phase space source in the previous simulation (not including recycling), NPASS_ph_sp is the number of particles rejected from the previous simulation because they were multiple-passers (not including recycling), and NFAT_ph_sp is the number of photons (not including recycling) rejected because

they fall outside the directional bremsstrahlung splitting field radius at the SSD (only if ISRC_DBS=1–see below for more details). NNPHSP, NTOT_ph_sp, NPASS_ph_sp and NFAT_ph_sp can be found in the .egslst file from the previous simulation.

Note that, even with NRCYCL> 0, the input variable NCASE (see section 10.7) still controls the number of histories simulated. The simulation will stop as soon as NCASE histories have been run, regardless of whether the current particle has been recycled the full NRCYCL times or not.

Inputs IPARALLEL and PARNUM
Note: the input variables IPARALLEL and PARNUM are no longer necessary if you are using the built-in parallel processing functionality in BEAMnrc. In previous versions of BEAMnrc, these inputs were essential for dividing a phase space source into IPARALLEL equal partitions. Each job used a different partition given by:

$$(\texttt{PARNUM}-1)*\left(\frac{\texttt{NNPHSP}}{\texttt{IPARALLEL}}\right)<\texttt{INPHSP}\leq\texttt{PARNUM}*\left(\frac{\texttt{NNPHSP}}{\texttt{IPARALLEL}}\right)$$

where NNPHSP is the total number of particles in the phase space source and INPHSP is the particle number used. The Unix script, pprocess, used for parallel job submission with previous versions of BEAMnrc, set the values of IPARALLEL and PARNUM automatically. You only need to be concerned with these inputs if you are submitting parallel jobs one by one, manually creating an input file for each of them.

Inputs re DBS
If directional bremsstrahlung splitting (DBS) was used in the BEAM simulation used to generate this source, then it is recommended that you use the inputs ISRC_DBS, RSRC_DBS, SSDSRC_DBS and ZSRC_DBS to prevent fat photons from compromising dose or fluence statistics in the current simulation. RSRC_DBS and SSDSRC_DBS, the radius and SSD of the DBS splitting field respectively, are available from the DBS inputs for the BEAMnrc simulation used to generate the phase space source. ZSRC_DBS will be equal to the Z position of the back of the component module where this source was scored (this information is available in the .egslst file from the BEAMnrc simulation that generated the source). When ISRC_DBS is set to 1, photons in the source are projected along their trajectory from ZSRC_DBS to SSDSRC_DBS. If they fall outside RSRC_DBS at SSDSRC_DBS then they are not used in the simulation. In the context of the BEAMnrc simulation that generated this source, these photons are fat (but this information is not stored in the phase space file). Note that

Charged particles are never rejected with this technique, which means that if you do not want fat charged particles to compromise dose statistics (especially near the surface of a phantom) then you must use the electron splitting option in DBS.

For more information about DBS, see section 6.3.4 of this manual.

3-D and 4-D IAEA phase space sources Source 21 is capable of handling IAEA format phase space sources in which the (X,Y,Z) position of each particle is scored (3-D) and/or in

which the fractional monitor unit index (MU) associated with a particle is scored (4-D). This functionality has been adapted from modifications by Lobo & Popescu[24].

Handling of 3-D phase space data is essential if using accelerator phase space data acquired on a non-planar surface (such as the data supplied by Varian for their TrueBeam accelerators) or if using phase space data output by a DOSXYZnrc simulation (see the DOSXYZnrc Users Manual[15]). If data is scored in 3-D, this is automatically detected by the source 21 algorithm. Particles are incident within a component module (or multiple CMs), and the particle Z-position read from the phase space source overrides the top of INIT_ICM (see above) as the incident Z-position. Particles must be incident within CMs that handle internal sources: SLABS, FLATFILT or SIDETUBE. If, during a simulation, a particle is found to be incident within a CM that does not match one of these types, the simulation stops with an error message.

MU is automatically scored in IAEA-format phase space files generated by accelerator simulations having one or more synchronized CMs with time-varying opening coordinates (see Section 15 below). There is also an option to include MU in the 3-D phase space files generated using DOSXYZnrc with a synchronized phase space source or synchronized BEAMnrc simulation source (see the DOSXYZnrc Users Manual[15]). MU gives a time dimension to the phase space data. Hence, this data is termed 4-D. If MU is scored in an IAEA phase space file being used as a source, this is automatically detected by the source 21 algorithm. Instead of choosing a random value for MU at the beginning of each history, a BEAMnrc simulation using a 4-D phase space source uses the MU's read from the source to set any time-varying opening coordinates of synchronized CMs in the accelerator. This allows the CMs to be synchronized with those in the upstream simulation(s) used to generate the phase space source.

For more information related to phase space sources, see the section 7 on full phase space files.

4.14 ISOURC=24: Phase Space Source Incident from User-specified Angle

IDUMMY, 24, INIT_ICM, NRCYCL, IPARALLEL, PARNUM, ISRC_DBS, RSRC_DBS,
SSDSRC_DBS, ZSRC_DBS
SPCNAM
ALPHA24, BETA24, DIST24

This source is similar to the phase space source (21) with the exception that the user can specify a point of rotation on the Z-axis above INIT_ICM and rotation angles about the X- and Y-axes. Input parameters are the same as ISOURC=21 with additional inputs:

ALPHA24 Angle of rotation of source (phase-space) plane about an axis parallel to the X-axis (degrees). Positive angle is clockwise rotation. -90 degrees < ALPHA24 < 90 degrees.

BETA24 Angle of rotation of source plane about an axis parallel to the Y-axis (degrees). Positive angle is counter-clockwise rotation. -90 degrees < BETA24 < 90 degrees.

DIST24 Distance of point of rotation above INIT_ICM on the Z-axis (cm).

Note that if either ALPHA24 or BETA24 is non-zero INIT_ICM must be > 1, since any rotation of the source will result in particles incident from within INIT_ICM-1. In this case, INIT_ICM and INIT_ICM-1 must be SLABS, FLATFILT, or SIDETUBE, since these are currently the only CMs capable of handling particles incident within them.

The initial idea and much of the coding of ISOURC=24 is courtesy of Patrick Downes at University of Cardiff, Wales.

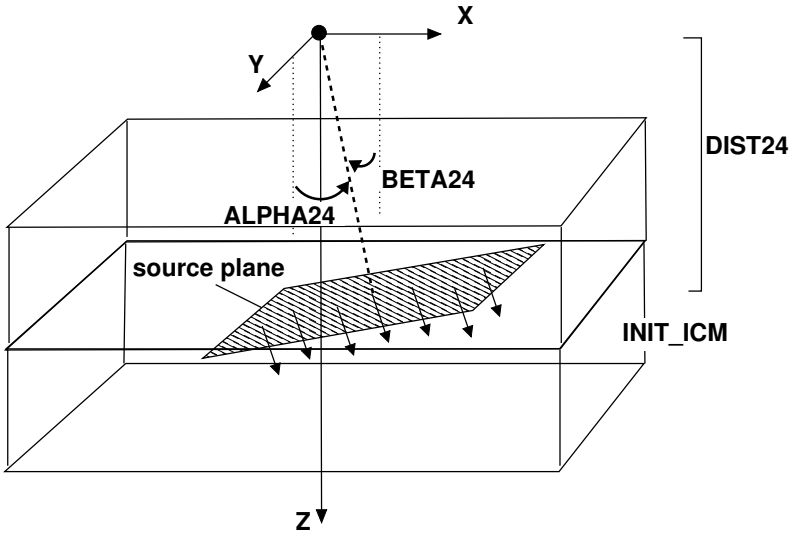


Figure 18: Phase-space source incident from user-specified angle (ISOURC=24). Similar to the standard phase-space source (ISOURC=21) except that the user can specify a point of rotation on the Z-axis above INIT_ICM, DIST24, and angles of rotation about axes parallel to the X-axis, ALPHA24, and Y-axis, BETA24.

4.15 ISOURC=23: BEAM Simulation Source Incident from User-specified Angle

IDUMMY, 23, INIT_ICM, ISRC_DBS, ALPHA24, BETA24, DIST24
the_beam_code, the_pegs_file, the_input_file

This source allows an accelerator that has been compiled as a shared library to be used as a source in a BEAM simulation. In function, it is similar to the phase-space source incident from a user-specified angle, `ISOURC`=24, but it eliminates the need to store an intermediate phase space file since the BEAM simulation reads phase-space data at the scoring plane in the simulation source directly. The simulation source runs simultaneously with and under the control of the BEAM simulation using it, so primary histories in the simulation source are run each time new phase-space data is required by the BEAM simulation using it. Note that this also means that any need to recycle phase space data is eliminated, another important advantage of the simulation source.

For information about compiling an accelerator as a shared library source, the user is urged to refer to Section 4.10.1 in the DOSXYZnrc Users Manual[15].

Input parameters for source 23 are:

INIT_ICM CM no. at the top of which the source is incident.

ISRC_DBS Set to 1 if directional bremsstrahlung splitting (DBS) is used in the simulation source and you wish to reject fat photons outside the splitting field. This is recommended since these photons can compromise statistics in the downstream BEAM simulation.

ALPHA24 Angle of rotation of the source plane about an axis parallel to the X-axis (degrees). Positive angle is clockwise rotation.

BETA24 Angle of rotation of the source plane about an axis parallel to the Y-axis (degrees). Positive angle is counter-clockwise rotation.

DIST24 Distance of point of rotation on the Z-axis above `INIT_ICM` (cm).

the_beam_code The name of the accelerator to be used as a simulation source. This must include the `BEAM_` prefix (*i.e.* `BEAM_accelname`). The accelerator must have been compiled as a shared library on the architecture that the BEAM simulation is running on.

the_pegs_file The name of the PEGS4 data set used by the simulation source (*i.e.* may be different than that used by the BEAM simulation using the simulation source).

the_input_file The input file for the simulation source. This must exist in the directory `BEAM_accelname`, where `accelname` is the name of the accelerator being used as the simulation source. Also, this input file must specify one phase space scoring plane, and the input parameter `IO_OPT` must be set to 1 to output phase-space data at this scoring plane (see Section 10.4. No phase-space file will be output, but the phase-space data at this scoring plane will be read directly by the BEAM simulation using the source.

Note that `the_input_file` will specify a value for `NCASE` (no. of histories) but this will be ignored since the number of histories run is dictated by the number of incident particles required by the simulation using this source.

The definitions of the parameters specifying the incident direction, `ALPHA24`, `BETA24` and `DIST24`, are the same as those for `ISOURC`=24 and shown in Figure 18. Note that if `ALPHA24` and `BETA24` are not both 0, then `INIT_ICM` must be >1 since some particles will be incident from within `INIT_ICM`-1 and `INIT_ICM` and `INIT_ICM`-1 must be of type(s) SLABS, FLATFILT or SIDETUBE.

Unlike the phase-space sources, `ISOURC`=21 and `ISOURC`=24, the only parameter that needs to be set to reject fat photons from a simulation source using directional bremsstrahlung splitting (DBS) is `ISRC_DBS`=1. This is because information about whether a photon is fat (directed outside the splitting field) is available directly from the simulation source, so there is no need to reconstruct the splitting field and the photon's path relative to it.

In the case of parallel BEAM runs using a simulation source, each parallel job starts its simulation source with different random number seeds, ensuring that incident particles are not identical. This is in contrast to phase-space sources, where each parallel job uses a different partition, or "chunk", of the phase-space data (see Section 13.2). For more information about parallel runs see Section 13.

4.16 ISOURC=31: Phase Space Reconstructed Using Beam Models

IDUMMY, 31, CMSOU, SPCNAM

This source routine reconstructs the phase space parameters using the beam data derived from a beam characterization model. The reconstructed phase space sources can be incident on any CM. Use of beam characterization models can save CPU time for the accelerator head simulation and also result in significant reduction in disk space requirement for phase-space storage. For more information related to beam reconstruction using beam characterization models, see NRCC report "Beam Characterization: a Multiple-Source Model" by Ma and Rogers (1995).

Input parameters for a beam model source are:

CMSOU Component module on which the phase space source is incident (defaults to 1 if it is set to < 1 or $>$ the number of `CMs` in the model)

SPCNAM Filename (with extension) of the beam model parameters.

By default, this source is not included in the accelerator models. If you want to include them you must do several things.

- Copy the files `beammodel_macros.mortran` and `beammodel_routines.mortran` from `$OMEGA_HOME/progs/beamdp` to the directory with you accelerator.
- Update the `sources.make` file on that same area to include the above two files. If you are using a standard accelerator model, you may just copy `$OMEGA_HOME/beamnrc/sources.make.MS` to your accelerator subdirectory and rename it as `sources.make`.
- `make` the accelerator

5 Monoenergetic vs Energy Spectrum Sources

In any of the above sources, with the exception of the phase space and beam simulation sources (ISOURC=21,23,24), incident beam energy can be either monoenergetic or described by an energy spectrum. Inputs associated with incident beam energy are:

MONOEN Set to 0 if the beam is monoenergetic (default); set to 1 if the beam energy is described by a spectrum

EIN (required if `MONOEN` = 0) The kinetic energy of a monoenergetic beam in MeV; defaults to 1.25 MeV if EIN is set ≤ 0

FILNAM (required if `MONOEN` = 1) The filename (with extension) containing information describing the incident beam's energy spectrum (more on spectrum file format below)

IOUTSP (required if `MONOEN` = 1) Set to 0 for no summary of the incident beam energy spectrum in the `.egslst` output file (default); set to 1 if a summary is desired

The energy spectrum file, `FILNAM`, has a specific format:

```
SPEC_TITLE
NENSRC, ENMIN, IMODE
ENSRCD(I), SRCPDF(I) (I = 1 to NENSRC)
```

where:

SPEC_TITLE is an 80-character spectrum title

NENSRC = # of energy energy bins in the spectrum histogram

ENMIN = lower energy of first bin in MeV

IMODE Set to 0 for histogram counts/bin; set to 1 for counts/MeV

ENSRCD(I) = upper energy of bin I in MeV

SRCPDF(I) = probability of finding a particle in bin I (SRCPDF need not be normalized)

The code randomly distributes incident particle energies equally across each energy bin.

In `$HEN_HOUSE/spectra/egsnrc/` there are a collection of spectra developed at NRC over the years for various applications. We will happily add any other spectra sent to us in the correct format, and with some semi-adequate documentation.

6 Variance Reduction in BEAMnrc

There is an extensive discussion of variance reduction in BEAM in the original BEAM paper [3].

6.1 Range Rejection

This section contains a brief description of how range rejection is performed and describes the input variables used to control range rejection. Range rejection is used to save computing time, in the process introducing approximations in the simulation (see below). Thus, it really falls under the category of an "approximate efficiency improvement technique" (AEIT) as opposed to being a true variance reduction technique (VRT).

The basic method for range rejection is to calculate the range of a charged particle and terminate its history (depositing all of its energy at that point) if it cannot leave the current region with energy > ECUTRR. ECUTRR is the range rejection cutoff energy which may vary from region to region depending on the type of range rejection used (see below). To determine the range to ECUTRR, BEAMnrc calculates the range from ECUTRR to AE (using EGSnrc macros and EGSnrc-calculated tables of range to AE as a function of electron energy) for each region at the beginning of the simulation (zero if ECUTRR = AE). Then, before each charged particle step during the simulation, this value is subtracted from the particle's range to AE (which is always calculated by EGSnrc). Ranges are calculated using restricted stopping powers and, thus, represent the longest possible ranges to ECUTRR.

When the range rejection control variable IREJCT_GLOBAL is set to 2, range rejection is performed on a region-by-region basis. In this case ECUTRR in each region is simply equal to the value of ECUT in the region. If the range to ECUTRR is less than the perpendicular distance to the nearest region boundary, the history is terminated and energy is deposited in the current region. On the other hand, when IREJCT_GLOBAL is set to 1, ECUTRR is the minimum energy that a charged charged particle can have as it leaves the current region and still reach the bottom of the accelerator with an energy greater than the global ECUT. ECUTRR is automatically calculated for each region at the beginning of the simulation. Similar to IREJCT_GLOBAL=2, if the range to ECUTRR is less than the perpendicular distance to the nearest region boundary, the particle is terminated and energy deposited in the current region. IREJCT_GLOBAL=1 can save more time than IREJCT_GLOBAL=2 but is generally only useful for the case where the user is only interested in phase space data at the bottom of the accelerator, since the higher values of ECUTRR higher up in the accelerator may lead to inaccuracies in scored quanties (*e.g.*, dose) in these regions. Note, one can increase the time saved by IREJCT_GLOBAL=2 by judicious setting of ECUT in different regions throughout the accelerator.

Range rejection introduces an approximation because, in terminating a charged particle's history and depositing all of its energy in the current region, it is assumed that any bremsstrahlung photons that would have been created by the particle, do not leave the current region. The user can minimise inaccuracies resulting from this approximation using the input variable ESAVE_GLOBAL defining the maximum charged particle energy (in MeV) at which range rejection is considered. The choice of ESAVE_GLOBAL depends on the incident

beam energy and the materials that it is passing through. ESAVE is treated internally on a region by region basis, but only in the CM SLABS does the user currently have the ability to assign individual values to each region (via ESAVEIN, this is because this CM is used for bremsstrahlung targets and we thought we might need more control).

Rejection of particles based on range to ECUTRR, as described above, is defined in the BEAMnrc macro, $USER-RANGE-DISCARD. This is implemented immediately after EGSnrc's intrinsic range rejection routine, which is based on particle range to AE, the lower energy limit for cross section and stopping power data, and which, in general, is not as stringent a rejection criterion.

6.2 Photon Forcing

BEAMnrc offers an option whereby the user can force photons to interact in specified CMs within a simulation. This option is useful for improving statistics of scattered photons when photon interactions are sparse (*eg.* in thin slabs of material or in material with low density). One of the main purposes of implementing this option was to study the generation of contaminant electrons in a photon beam.

Briefly, a photon forced to interact in a CM is "split" into a scattered photon whose weight is equal to the probability of interaction and an unscattered photon carrying the remaining weight. The unscattered photon proceeds as if an interaction did not take place and is transported directly through the forcing zones with no further interaction. The scattered photon can be forced again in the forcing zone depending on how many interactions are allowed to be forced.

The input variables used to control photon forcing are:

IFORCE Set to 0 for no forcing (the default); set to 1 for photon forcing

NFMIN Photon interaction # at which to begin forcing (defaults to 1). Currently, the option to start photon forcing at any interaction other than the first one in the forcing CMs has been disabled. Thus, NFMIN is effectively 1 regardless of what is input.

NFMAX Photon interaction # after which to stop forcing (defaults to 1)

NFCMIN CM # at which to start forcing (defaults to 1)

NFCMAX CM # beyond which to stop forcing (defaults to the # of CMs in the accelerator)

Briefly, a photon forced to interact in a CM is "split" into a scattered photon whose weight is equal to the probability of interaction and an unscattered photon carrying the remaining weight. The unscattered photon proceeds as if an interaction did not take place and is transported directly through the forcing zone (*i.e.*, the forcing CMs) with no further interaction. This process is repeated along the path of the scattered photon (and its descendants) until a total of NFMAX interactions have been forced. In each case, the weight of the scattered photon is equal to the weight of the parent photon times the probability of interaction, and the weight of the unscattered photon is equal to the weight of the parent

photon times (1 - the probability of interaction). Note that, depending on the setting of NFMAX, this can lead to a broad distribution of particle weights at the bottom of the forcing zone.

The number of times interactions have been forced is passed onto higher-order photons arising from non-forced events (*e.g.*, bremsstrahlung, positron annihilation). Thus, if the parent photon has not yet undergone NFMAX forced interactions, or if the parent is a charged particle, these higher-order photons will be forced to interact the remaining number of times (*i.e.* NFMAX - # of times parent particle forced) as long as they are within the forcing zone. This does not affect any remaining forced interactions of the parent photon. This feature is useful for getting good statistics on bremsstrahlung photon interactions and makes Photon Forcing a powerful tool for improving statistics when used in conjunction with bremsstrahlung splitting (see next section).

Note that forcing in regions with low mass can lead to a few secondary electrons created outside the forcing region having much larger weights than those created inside the forcing region and thereby distort results which are sensitive to these weight variations.

6.3 Bremsstrahlung Photon Splitting and Russian Roulette

Bremsstrahlung photon splitting offers the user another variance reduction technique which improves the statistics of bremsstrahlung photons resulting from electron interactions. The technique has been described for general use in EGS in references [25, 26]. BEAMnrc offers two bremsstrahlung splitting techniques, uniform bremsstrahlung splitting (UBS) and directional bremsstrahlung splitting (DBS). Prior to 2013, BEAMnrc also supported selective bremsstrahlung splitting (SBS). However, this has been superseded by the more efficient DBS. UBS has been optimised in BEAMnrc with the addition of the Russian Roulette feature (not required for DBS).

6.3.1 Uniform Bremsstrahlung Splitting

Input variables associated with uniform bremsstrahlung splitting (UBS) are:

IBRSPL Set to 1 for uniform bremsstrahlung splitting

NBRSPL The splitting number. Also applied to higher-order bremsstrahlung and annihilation photons if Russian Roulette is on (see section 6.3.3 below).

Each bremsstrahlung event produces NBRSPL photons, each having a weight equal to $\frac{1}{\text{NBRSPL}}$ times the weight of the electron that underwent the bremsstrahlung event. The energies and directions of each photon are sampled individually according to the relevant probability distributions. The energy of the primary electron is decremented by the energy of just one of these photons. This must be done in order to preserve energy straggling and means that, while energy is not conserved for a single history (the energy would have to be decremented by the average energy of the photons created to achieve this), it is conserved "on average" over many histories.

The splitting number, NBRSPL, is not applied to higher-order bremsstrahlung and annihilation photons unless Russian Roulette is turned on (see section 6.3.3 below). This prevents wasting simulation time by tracking many higher-order photons of vanishing weight.

Uniform bremsstrahlung splitting in BEAMnrc is handled by the EGSnrc system using an efficient internal bremsstrahlung splitting algorithm. So, other than input, the only coding in BEAMnrc related to UBS involves turning off splitting for higher-order events.

6.3.2 Selective Bremsstrahlung Splitting

Selective bremsstrahlung splitting (SBS) has been superseded by directional bremsstrahlung splitting (DBS) (see section 6.3.4 below) and is no longer supported.

6.3.3 Charged Particle Russian Roulette

Note that directional bremsstrahlung splitting (DBS), described in section 6.3.4 below does not make use of this charged particle Russian Roulette input.

When using UBS, following all of the secondary charged particles created by the "split" photons increases the CPU time required for simulations. If the primary interest is in secondary electrons or their effects (*eg.* dose deposition), the extra computing time is obviously acceptable. But if, as is often the case, the main interest is in the bremsstrahlung photons themselves, one can reduce the CPU time while still preserving the variance reduction advantages of bremsstrahlung splitting by using a Russian Roulette technique with any charged particles generated by the split photons.

The possible settings of the Russian Roulette input, IRRLTT, are:

= 0 Russian Roulette off. No splitting of higher-order bremsstrahlung annihilation photons.

= 1 Option discontinued. This defaults to IRRLTT=2 (see below).

= 2 Russian Roulette on. Higher-order bremsstrahlung and annihilation photons are split with splitting number NBRSPL in UBS.

Russian Roulette is implemented by giving secondary charged particles resulting from split photons a survival threshold. The survival threshold is always the inverse of the photon splitting number. Thus, in the case of UBS, the threshold is fixed and is equal to $\frac{1}{NBRSPL}$. Then a random number is chosen for each charged particle. If the random number is less than the survival threshold, the charged particle survives, and its weight is increased by a factor of NBRSPL. Otherwise, the charged particle is eliminated. Secondary charged particles subject to Russian Roulette are electrons resulting from Compton events and photoelectric events and electrons and positrons resulting from pair production.

Note that if Russian Roulette is turned on, then higher-order bremsstrahlung and annihilation photons are also split. This is because any charged particle surviving Russian Roulette has a weight higher than the photon that created it. If radiative products from this surviving

charged particle are not split, then their high weight may interfere with the statistics of the original split bremsstrahlung photons. Also, splitting of higher-order bremsstrahlung and annihilation photons does not greatly increase computing time when Russian Roulette is on because most of the secondary charged particles have been eliminated.

For a more complete description of Uniform Bremsstrahlung Splitting and Russian Roulette, see the BEAM paper [3].

6.3.4 Directional Bremsstrahlung Splitting (DBS)

DBS inputs

Directional bremsstrahlung splitting was introduced into BEAMnrc in 2004 and results in greater efficiency than SBS[18]. Our results have shown that using DBS in a photon beam can result in fluence efficiencies up to 8 times higher than with SBS (up to 20 times higher than with UBS) and dose efficiencies up 6 times higher than with SBS (up to 26 times higher than with UBS). The actual improvements will depend upon the energy and other details of the photon beam being simulated.

DBS began from the same philosophy as SBS, in which bremsstrahlung photons aimed into a field of interest (encompassing the treatment field) are split at the time of creation, while those aimed away from the field are not. Beyond that, however, the two algorithms are completely different.

The input parameters for DBS are:

`IBRSPL` Set to 2 for directional bremsstrahlung splitting

`NBRSPL` Splitting number. Set to ≈1000 if you are using charged particle splitting (see below) or ≈5000 if not. Note that `$MXSTACK` in the file `beamnrc_user_macros` must be increased to avoid overflows and to be safe, should probably be an order of magnitude larger than `NBRSPL`.

`FS` The radius of the DBS splitting field in cm. This must at least encompass the entire treatment field and should go sufficiently beyond the edges of the treatment field that the contribution of fat photons from outside the splitting field (see below for more details) to dose below `SSD` is negligible. For a 10×10 cm2 field, we recommend `FS`=10 cm.

`SSD` The Z value (in cm) at which the above `FS` is defined.

`ICM_DBS` The component module (CM) number in which electron (charged particle) splitting is to take place. Set this equal to the CM number modeling the flattening filter in your accelerator. Note that electron splitting is enabled in FLATFILT and PYRAMID CMs, so one of these, likely FLATFILT, must be used to model the flattening filter. If `ICM_DBS` is set to 0, then electron splitting is turned off.

`ZPLANE_DBS` The plane number in CM no. `ICM_DBS` at which electron splitting will take place. Planes correspond to layer boundaries within the CM. Set this equal to the back surface of the flattening filter (i.e., `ZPLANE_DBS`=no. of layers in flattening filter + 1).

IRAD_DBS Set to 1 to redistribute the NBRSPL split charged particles in a radially-symmetric manner about the Z axis. This can improve charged particle statistics if your beam is radially symmetric.

ZRR_DBS The Z position of the Russian Roulette plane used in conjunction with charged particle splitting (see below for more details). Place this several mm above the splitting plane, still within the flattening filter.

Use a rejection plane [USE_REJPLN] Options are ON or OFF. If Use a rejection plane= On then define a *rejection plane* which discards fat photons and fat charged particles if they interact below the rejection plane. This prevents correlated split photons from being created close to the scoring plane and degrading the statistics of the scored quantities. Our Monte Carlo studies show that eliminating these fat particles has a negligible effect on fluence scoring[27] and dose scoring[18] for typical field-sizes of interest. The rejection plane is meant to be placed in the air layer between the bottom of the last component in the accelerator geometry and the scoring plane. This input must appear between the delimiters :Start DBS rejection plane: and :Stop DBS rejection plane: at the end of the input file. See below for more details.

Z(cm) from zero reference plane [Z_REJPLN] The distance, in cm, between the rejection plane and the plane where Z=0. If the scoring plane is at an SSD = 100 cm, then a typical Z_REJPLN could be between 50 and 90 cm. This input must appear between the delimiters :Start DBS rejection plane: and :Stop DBS rejection plane: at the end of the input file. See below for more details.

Both the USE_REJPLN and Z_REJPLN differ from other DBS-related inputs in that they must appear between delimiters :Start DBS rejection plane: and :Stop DBS rejection plane: in the .egsinp file. For example, the following input defines a rejection plane at Z=22.5 cm:

```
:Start DBS rejection plane:

 Use a rejection plane= On
 Z(cm) from zero reference plane= 22.2

:Stop DBS rejection plane:
```

This section must appear in the .egsinp file somewhere below the input for the last component module. Note that the BEAM GUI automatically reads/writes the rejection plane inputs in the correct format, so in practice the user does not need to concern themselves with it.

Outline of the DBS algorithm

If a charged particle undergoes a bremsstrahlung or annihilation event, then DBS splits this event NBRSPL times. The resultant photons all have their weight multiplied by $\texttt{NBRSPL}^{-1}$. DBS then loops through these split photons and for each one, determines whether or not it is aimed into the splitting field defined by FS and SSD. If it is, then the photon is kept and is considered "non-fat" (low-weight). If not, then Russian Roulette is played on the photon

by comparing a random number to a survival threshold of NBRSPL^{-1}. If the random number is less than this number, then the photon is kept and its weight is multiplied by NBRSPL and it is considered a "fat" (high-weight) photon. Splitting is not limited to bremsstrahlung and annihilation events, however. If one of these fat photons undergoes a Compton event, then it is also split NBRSPL times and the same Russian Roulette scheme as above is applied to photons not aimed into the splitting field. Moreover, the basic DBS algorithm eliminates all but a few charged particles by:

1. playing Russian Roulette (survival probability NBRSPL^{-1}) with all electrons resulting from a split Compton event
2. requiring "non-fat" photons (ie low-weight photons that are actually aimed into the splitting field) to survive Russian Roulette (survival probability NBRSPL^{-1}) before they can undergo pair production, photoelectric or Compton events. This condition is relaxed if the event is about to happen in a gas ($\rho < 1.2 \times 10^{-2}$ g/cm^3) to prevent fat photons (ie survivors of Russian Roulette) from being generated in the air just above the splitting field and then Compton scattering into the field, thus compromising the photon statistics.

It can be seen that, other than non-fat charged particles generated in the air just above the splitting field due to the exception in (2) above, those few charged particles that do survive will have a high weight (ie will be fat).

DBS results in many non-fat photons inside the splitting field and few fat photons outside the splitting field. All of the non-fat photons inside the splitting field will have the same low weight (NBRSPL^{-1} times the weight of the incident particles). This is desirable since a large variation of weights inside the field was one factor compromising the efficiency of SBS.

DBS as described thus far is very efficient for photon fluence/dose, however it will result in only a few fat charged particles reaching the SSD. If you are interested in the charged particle contribution to dose (as in most realistic cases), you must use the electron (really, charged particle) splitting function to "recover" the charged particles.

The inputs for electron splitting are ICM_DBS, ZPLANE_DBS, IRAD_DBS and ZRR_DBS. ICM_DBS and ZPLANE_DBS define a CM number and a plane number (which will correspond to a layer boundary) within that CM at which all fat charged particles are to be split NBRSPL times (with their weights multiplied by NBRSPL^{-1}). If you have set IRAD_DBS=1, then these split charged particles will be redistributed in a radially-symmetric manner about the beam axis. Radial redistribution can result in better statistics as long as the beam has radial symmetry above the electron splitting plane. ZRR_DBS defines a "Russian Roulette plane" which is always above the electron splitting plane. Below this Russian Roulette plane, DBS is carried out in the following manner:

1. electrons resulting from a split Compton event are not subject to Russian Roulette
2. non-fat photons undergo pair production, photoelectric and Compton events
3. If a fat photon (i.e., one not aimed into the splitting field) undergoes a pair production or photoelectric event, then the event is split NBRSPL times to create NBRSPL (photoelectric) or 2*NBRSPL (pair production) non-fat charged particles.

Together, use of the splitting and Russian Roulette planes ensures that all charged particles reaching the SSD are non-fat (and that there are many of them).

DBS parameter selection

We have found that, although optimal settings of the DBS splitting parameters depend on the details of the accelerator being simulated, some generalizations can be made. `NBRSPL`, the splitting number, should be set ≈1000 for peak efficiency if you are also using electron splitting. If electron splitting is off, then peak efficiency occurs at higher splitting numbers (≈5000). The splitting field radius `FS`, must at least enclose the entire beam field. However, it should also go sufficiently beyond the edges of the beam field that the dose contribution below the SSD from fat photons (from outside the splitting field) is negligible (more about this below). The efficiency of DBS decreases with increasing `FS`, so it is desirable to select the smallest `FS` possible that still provides adequate coverage. We have found that for a 10×10 cm^2 field, a splitting field radius of 10 cm goes sufficiently beyond the edges without a significant sacrifice in efficiency. If you are in doubt about how far beyond the edges of the beam field the splitting field should go, it is okay to overestimate by up to 5 cm, and the gain in efficiency will still be significantly greater than UBS or SBS.

If you are using electron (charged particle) splitting, then you must select the locations of the splitting and Russian Roulette planes. We have found that the optimum location for the splitting plane is at the very back (ie downstream surface) of the flattening filter, with the Russian Roulette plane slightly above this. Because of the way electron splitting is coded, the splitting plane is specified by a CM number, `ICM_DBS`, and then a plane number, `ZPLANE_DBS`, within that CM. Thus, `ICM_DBS` should correspond to the CM modeling the beam flattening filter. Currently, splitting is only enabled in the FLATFILT and PYRAMID CMs, so this means that your flattening filter must be modeled with one of these, likely FLATFILT. If you are using the BEAM GUI, input of `ICM_DBS` is made easier by only presenting the user with a list of the FLATFILT and PYRAMID CMs in the accelerator. Planes within a CM correspond to layer boundaries in the geometry, so to place the splitting plane at the very back of the flattening filter, set `ZPLANE_DBS`=(no. of layers in flattening filter + 1). Again, GUI input makes selection of `ZPLANE_DBS` easier by showing the user a list of possible planes (along with their Z positions) within CM number `ICM_DBS`.

Finally, the user must select `ZRR_DBS`, the Z position (in cm) of the Russian Roulette plane. `ZRR_DBS` should be several millimeters above the splitting plane, but its exact value is not critical, since, with the eception of a sharp drop in efficiency when the `ZRR_DBS` is < 1 mm above the splitting plane, the overall variation in efficiency with distance between the Russian Roulette and splitting planes is small. In a simulated 6 MV beam (10×10 cm^2 field) from an SL25 accelerator, Russian Roulette was optimized with the splitting plane at Z=15.66 cm (the back of the flattening filter), and the `ZRR_DBS`=15.5 cm.

Augmented range rejection with DBS

When using DBS with electron splitting and charged particle range rejection in the simulation, the user has the option to use an augmented range rejection scheme which is more efficient. If you are using the BEAMnrc GUI, this is selected by clicking on the "Augmented range rejection" checkbox in the DBS input frame. If you are editing the `.egsinp` file di-

rectly, then this option is selected by changing the IREJCT_GLOBAL input (see Section 6.1) to -IREJCT_GLOBAL (*e.g.* if you had originally set IREJCT_GLOBAL=2, for region-by-region range rejection, then set IREJCT_GLOBAL=-2 for augmented region-by-region range rejection with DBS). Augmented range rejection is identical to the standard range rejection selected (see Section 6.1) with the exception that all non-fat charged particles (which will exist only if electron splitting is being used with DBS) that cannot make it to the nearest region boundary with energy > ECUT are subject to Russian Roulette (survival probability=1/NBRSPL) regardless of whether or not their energy is greater than the range rejection cutoff energy (ESAVE_GLOBAL). Recall that in the standard schemes, range rejection is not considered if the particle energy is > ESAVE_GLOBAL. The augmented scheme does not influence bremsstrahlung production, since particles that survive the Russian Roulette (with their weight increased by a factor of NBRSPL) still have a chance to undergo bremsstrahlung events. Preliminary studies in a simulated 6 MV photon beam have shown that augmented range rejection can reduce the CPU time by ∼20%. Note that if you have edited the .egsinp file and set IREJCT_GLOBAL<0 but are not using DBS, then standard range rejection corresponding to IREJCT_GLOBAL=|IREJCT_GLOBAL| is done.

When DBS is used in BEAM, an additional dose component (see Section 9) is output to the .egslst and .egsplot files which is the total dose minus the dose due to fat particles (or "total-fat"). The total-fat dose is included because the statistics of the total dose may be compromised by a few fat particles which do not significantly contribute to the dose. However, if the total dose is significantly higher than the total-fat dose over many dose zones, then the contribution from fat particles is significant, and you should consider re-running your simulation with the number of fat particles reduced. Fat charged particles are eliminated by turning charged particle splitting on, and the number of fat photons is reduced by using a larger splitting field radius, FS.

DBS gives rise to additional considerations when it is used in a BEAM simulation to generate a phase space file that is then used as a source in a second, downstream simulation (using BEAM, DOSXYZ, or another EGSnrc user code). Since the fat or non-fat status of a particle is not explicitly stored in the phase space file, we cannot use this information to exclude dose (or other scored quantity such as fluence) from fat particles in the downstream simulation. To overcome this, phase space sources in BEAM (see section 4.13) and DOSXYZ (see the DOSXYZnrc User's Manual[15]) have additional inputs which allow the user to reject photons that are not aimed into the DBS splitting field at the SSD (ie fat photons). Note that there is no such option to reject charged particles, so you must have used electron splitting to generate the phase space source if you are concerned about fat charged particles compromising statistics.

For a full discussion of the DBS algorithm and optimization of splitting parameters, we urge you to read the recent paper submitted to Medical Physics[18]. In addition to the DBS algorithm outlined above, this paper describes the various "smart" subroutines that DBS uses to eliminate the CPU-intensive task of generating and playing Russian Roulette with split photons aimed outside of the splitting field (and these routines are very important to the efficiency gains).

6.4 Directional Source Biasing (DSB)

Directional source biasing (DSB) was introduced in 2014 to improve the efficiency of isotropically-radiating sources in BEAMnrc. Its development was motivated by the reevaluation of Co-60 as a source in intensity modulated radiotherapy[], the potential for use of Co-60 treatment heads together with magnetic resonance imaging (MRI) in image guided radiotherapy (IGRT)[], and the ongoing interest in calculating beam quality correction factors, k_Q's, for ion chambers.

Since DSB is designed for isotropically-radiating sources, it must be used with `ISOURC`=3, the only such source in BEAMnrc (see Section 4.3). In addition, DSB uses many elements of the directional bremsstrahlung splitting (DBS) algorithm. Thus, directional bremsstrahlung splitting must be turned on by setting `IBRSPL`=2 (see Section 6.3.4 above).

The inputs exclusive to DSB are:

`i_dsb` Set to 1 for directional source biasing. Note that this input is only available with `ISOURC`=3.

`splitcm_dsb` Upon entering component module number `splitcm_dsb` primary photons will be split and radially redistributed. Note that this means that `splitcm_dsb` should be the first CM in the treatment head in which rotational symmetry does not exist. Set to 0 if there is no rotational symmetry in the treatment head geometry or if you do not wish to use it to increase the efficiency of the DSB routine. See below for more information.

`dsb_delta` The minimum linear distance, projected to the SSD, between primary photons when they are split and radially-redistribued at the top of the CM number `splitcm_dsb`. `dsb_delta` is used to set up a series of radial bins where all photons directed into a bin are split by the same number. See below for more information.

Inputs that are shared with directional bremsstrahlung splitting are:

`NBRSPL` Photon splitting number. In DSB is the total number of times that primary photons are split.

`FS` The radius of the DSB splitting field in cm. This should encompass the treatment field. Only primary photons directed into the field defined by `FS` and `SSD` (see below)

`SSD` The Z value (in cm) at which DSB splitting field radius, `FS`, is defined.

`ICM_DBS` The number of the component module (CM) in which electron (charged particle) splitting is to take place. This input has the same meaning as it does in directional bremsstrahlung splitting (see Section 6.3.4 above), and, indeed electron splitting is performed in the same way as it is in directional bremsstrahlung splitting. Electron splitting must be used if you are interested in charged particle contamination. Given the low energy of contaminant electrons in a Co-60 beam, for optimum electron fluence efficiency, electron splitting should take place at the bottom of the last field-defining

CM in the treatment head model. For example, in the Co-60 treatment head simulation used to implement and test DSB [28], ICM_DBS is the CM number corresponding to the PYRAMIDS CM used to model an adjustable collimator. Set ICM_DBS=0 (default) for no electron splitting.

ZPLANE_DBS This input functions the same as it does in electron splitting with directional bremsstrahlung splitting. When fat contaminant charged particles cross plane number ZPLANE_DBS in CM number ICM_DBS, they are split NBRSPL times. Planes correspond to layer boundaries within the CM, and ZPLANE_DBS is is usually set to the plane number defining the back surface of ICM_DBS.

IRAD_DBS Set to 1 to redistribute the NBRSPL split charged particles in a radially-symmetric manner about the Z axis. In a megavoltage photon beam making use of directional bremsstrahlung splitting, electrons are usually split in a portion of the accelerator with radial symmetry, so contaminant electron statistics can be improved by redistributing split particles in this way. However, in Co-60 treatment head simulations, electron splitting may be performed in a CM that does not have radial symmetry (*e.g.* a field-defining aperture), in which case the user should not opt for radial redistribution of split charged particles and IRAD_DBS should be set to 0 (the default).

ZRR_DBS The Z position of the Russian Roulette plane used in electron splitting. This input has the same function as it does in electron splitting with directional bremsstrahlung splitting (See Section 6.3.4 above). Thin contaminant charged particles below a Z position defined by ZRR_DBS are not subject to Russian Roulette. This should be placed within CM no. ICM_DBS several mm above the splitting plane defined by ZPLANE_DBS.

A schematic of the directional source biasing algorithm (without electron splitting) is shown in Figure 19 below. In practice, it is time consuming to play Russian Roulette with the split primary photons not directed into the splitting field, defined by FS and SSD, so BEAMnrc employs an algorithm which preferentially generates only those photons that are directed into the splitting field plus one fat photon representing the statistics of photons directed away from the field.

Further CPU time can be saved by utilizing the radial symmetry at the top of many treatment heads to reduce the number of primary photons that must be tracked there. To do this, the DSB algorithm divides the splitting field into radial bins, with minimum radii, r_1, r_2,...,r_{nbin}. Photons directed into bin i will be split i times with the split photons radially distributed about the beam central axis as soon as they enter CM no. splitcm_dsb. Thus, above splitcm_dsb, the number of primary photons that must be generated and tracked is reduced, using Russian Roulette, by a factor of i. It follows that the statistical weight of photons above splitcm_dsb is i/NBRSPL, while, after the splitting and radially redistributing the photons on entering splitcm_dsb, the weight is reduced by a factor of i to become 1/NBRSPL.

The minimum radii, r_i, of the bins are determined by the user input, dsb_delta, defining the minimum linear distance, projected to SSD, between split, radially redistribued photons:

$$r_i = \frac{\texttt{dsb\_delta}}{2sin\left(\frac{\pi}{i}\right)} \ , \ i = 2, 3, ..., \texttt{nbin} \qquad (1)$$

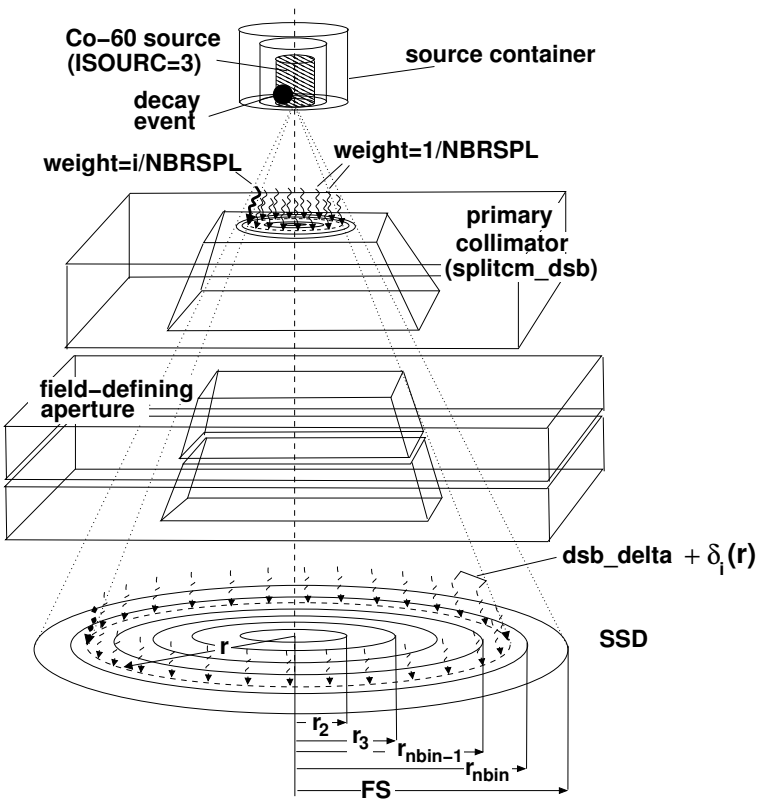


Figure 19: A schematic of directional source biasing (DSB) in a Co-60 treatment head simulation. The splitting field is defined by the user inputs, FS and SSD and contains the treatment field. If there is radial symmetry above CM number splitcm_dsb, the number of primary photons tracked in this region can be reduced by dividing the splitting field into nbin radial bins. The number of split primary photons directed into bin i, defined by minimum radius, r_i, is reduced by a factor of i and the weight of each photon is i/NBRSPL. Upon entering splitcm_dsb, photons are split i times and radially redistributed about the beam central axis, and their weight is decreased by a factor of i to become 1/NBRSPL. The individual r_i are determined by the user input, dsb_delta, defining the minimum linear distance between radially redistributed photons (*i.e.* for photons directed exactly at r_i).

where nbin is the total number of bins. Note that r_1=0. The number of bins, nbin, is equal to the number of times that photons are split when directed into the bin with minimum radius, r_{nbin}, and maximum radius, FS:

$$\texttt{nbin} = \frac{\pi}{asin\left(\frac{\texttt{dsb\_delta}}{2\texttt{FS}}\right)} - 1 \tag{2}$$

Note that the minimum value of dsb_delta is restricted by the requirement nbin≤NBRSPL. However, NBRSPL is usually very large ($\sim$ 20,000), and a more practical limit on the minimum dsb_delta is imposed by the maximum number of bins, $DSB_MAX_BIN, defined in $OMEGA_HOME/beamnrc/beamnrc_user_macros.mortran. Currently, $DSB_MAX_BIN is set to 1000.

Since dsb_delta defines the minimum linear distance, projected to SSD, between split, radially-redistributed photons, the actual linear distance between redistributed photons di-

rected towards radius, r, in bin i is `dsb_delta`+$\delta_i(r)$, where $\delta_i(r)$ increases linearly with r from 0 at r=r_i to $\alpha_{i,max}$`dsb_delta`, with $\alpha_{i,max}$ <1, as r→r_{i+1}.

In order to avoid having to reevaluate which radial bin a primary photon is directed into every time Russian Roulette is played on it or any of its higher-order descendents, the internal variable, `iphat`, is used When `iphat` was originally introduced with directional bremsstrahlung splitting, it took on only values of 0 or 1 depending on whether a particle was thin or fat. With DSB, however, `iphat` now takes on values of 1,2,...,`nbin`,`NBRSPL`. Above `splitcm_dsb`, `iphat` keeps an instantaneous record of what radial bin a particle (or its descendent) is directed into (`iphat`≤`nbin`) or, if set to `NBRSPL`, indicates that the particle is fat. Below the top of `splitcm_dsb`, `iphat` will be either 1, indicating a thin particle, or `NBRSPL`, indicating a fat particle. This redefinition of `iphat` allows quick calculation of Russian Roulette survival probabilities (*i.e.* 1/`iphat`).

The actual CPU time saved by reducing the number of photons tracked in the radially symmetric portion of the treatment head as described above depends on the design of the treatment head. If only a small portion of the head has radial symmetry, then the CPU time saved will be only a fraction of the overall efficiency improvement due to DSB.

If you are interested in contaminant electrons with DSB then you must make use of the electron splitting scheme implemented in directional bremsstrahlung splitting. Typical electron splitting inputs for a Co-60 treatment head simulation using DSB will be different from those for a megvoltage photon beam using directional bremsstrahlung splitting, and these differences are described in more detail near the beginning of this section.

If electron splitting is turned on, then DSB can also make use of directional bremsstrahlung splitting's augmented range rejection scheme (see Section 6.3.4 above). In this scheme, if a non-fat charged particle cannot make it to the next region boundary with energy > `ECUT`, then it is subject to Russian Roulette, with survival probability `iphat`/`NBRSPL`, regardless of whether or not its energy is < the range rejection cutoff energy, `ESAVE` (see Section 6.1). If the particle is not eliminated in this way, then it is subject to standard range rejection as described in Section 6.1.

DSB performance and parameter selection

Figure 20 below shows the relative efficiency of all doses > 0.5D_{max} calculated in a water phantom (0.5 cm × 0.5 cm × 0.5 cm voxels) using a BEAMnrc simulation of a typical Co-60 treatment head (10 cm × 10 cm field) as a source as a function of the photon splitting number, `NBRSPL`, selected in DSB. The Co-60 treatment head model is the same as that simulated by Mora et al[29] in 1999. Efficiency is expressed relative to the efficiency obtained with no variance reduction. The dotted line shows the maximum efficiency that can be obtained without DSB using varying values of `ECUT` in the treatment head, with some loss of accuracy. Electron contaminant dose is included.

Efficiency with DSB is optimized for `NBRSPL` ∼20,000 where it is a factor of ∼400 greater than that with no variance reduction and a factor of ∼40 greater than what can be achieved by varying `ECUT` in the treatment head. Efficiency does not show significant variation for `NBRSPL` > 20,000. Therefore, setting `NBRSPL`=20,000 should be sufficient for typical Co-60

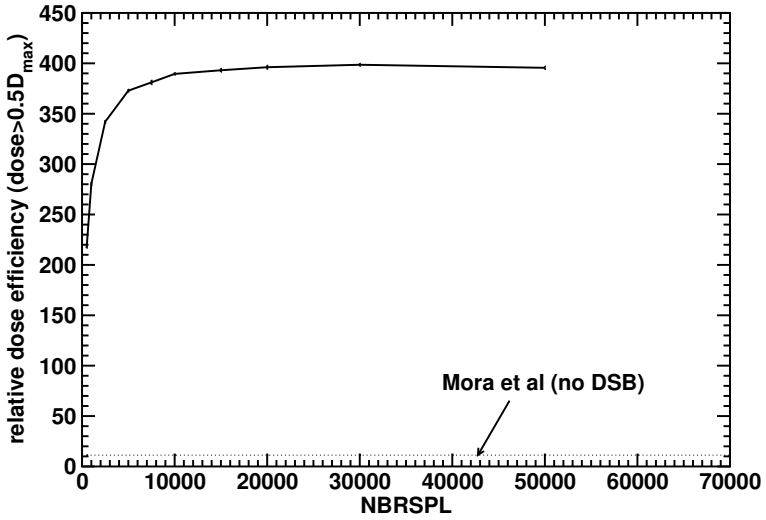


Figure 20: Efficiency of dose calculation in a water phantom (0.5 cm × 0.5 cm × 0.5 cm voxels) using a Co-60 treatment head simulation (10 cm × 10 cm field) as a source as a function of DSB photon splitting number, `NBRSPL`. Efficiency is evaluated for all doses > $0.5D_{max}$. Efficiency is relative to that with no variance reduction. The dotted line shows the maximum efficiency that can be achieved by varying `ECUT` in the treatment head, with some loss in accuracy. Electron contamination is included.

treatment head simulations.

Efficiency has also been studied as a function of the DSB parameter, `dsb_delta`, the minimum linear distance, projected to `SSD`, of radially redistributed photons when they are split upon entering `splitcm_dsb`. Within the range 1.0 cm ≤ `dsb_delta` ≤ 2.5 cm, photon and contaminant electron fluence effiency was not found to vary significantly. Thus, a setting of `dsb_delta`=1.5 cm is suggested.

Note that the time required to obtain 0.1% dose uncertainty using the simulated Co-60 treatment head[29] without DSB and using varying `ECUT` to improve efficiency is ∼100 hrs on 15 CPUs (1.8 GHz Opteron). The factor of ∼40 improvement in efficiency obtained using DSB means this calculation can be done in ∼2.5 hrs.

Since DSB potentially generates many incident primary photons per decay event (primary history), incident photon data is stored in an array. A new decay event is simulated only when all of the data in the array has been used.

For more detailed information about DSB, see the directional source biasing paper[28].

6.5 Bremsstrahlung Cross Section Enhancement (BCSE)

The Bremsstrahlung Cross Section Enhancement (BCSE) variance reduction technique is introduced into BEAMnrc in 2007 to improve the efficiency of simulations that involve x-ray production from bremsstrahlung targets (e.g. x-ray tubes and clinical linear accelerators), both in the kilovoltage and megavoltage range. The technique is discussed in detail in reference[27]. This section of the manual contains a brief description of the technique and

the input variables used to control it.

BCSE is most valuable in low-energy applications where bremsstrahlung production is rarest, and in 4π geometries where DBS is not applicable. Except for the few restrictions listed in section 6.5.3 below, BCSE is compatible with all other variance reduction techniques already used in BEAMnrc (e.g. range rejection, UBS and DBS). BCSE can be used alone or in conjunction with UBS or DBS; however, the largest efficiency gains are achieved when BCSE is optimally combined with UBS or DBS. For typical simulations of interest in medical physics, the efficiency gains can be up to five orders of magnitude over those obtained with analog simulations, and up to a full order of magnitude over those obtained with optimized splitting alone.

6.5.1 BCSE inputs

The input paramters associated with BCSE are currently passed to the code using delimiters in the input file (the same way the EGSnrc `MC transport parameters` are entered). The following is a template the user should copy to the end of the input file to be able to use BCSE (make sure there is no space between the equal sign and the text preceding it).

`:Start BCSE:`

`Use BCSE= ON` (could be `OFF`, internal variable `USE_BCSE` is 0 or 1 for `OFF` or `ON`, respectively).

`Medium to enhance= med` (`med` is the name of the medium in which to enhance the bremsstrahlung cross section (internal variable `BCSE_MEDNAME`). The medium must exist in the accelerator, otherwise BCSE will not be done.

`Enhancement constant= C`, where `C` is a real number $\geq$0 (internal variable `BCSE_FACTOR_C`)

`Enhancement power= N`, where `N` is a real number (internal variable `BCSE_POWER_N`)

`Russian Roulette= ON` (could be `OFF`. If the focus of the simulation is on the photons, it is strongly recommended to set Russian Roulette `ON`. (internal variable `IRRLTT` which is 0 if `OFF` and 2 if `ON`.)

`:Stop BCSE:`

The `Enhancement constant` (`C`) and `Enhancement power` (`N`) inputs determine the bremsstrahlung cross-section enhancement factor, `BCSE_FACTOR`, which is given by:

$$\texttt{BCSE\_FACTOR} = \begin{cases} \texttt{C} & \text{if N is set } \leq 0 \\ 1 + \texttt{C} \times E^{\texttt{N}} & \text{if N is set } > 0 \end{cases} \tag{3}$$

where E is the kinetic energy of the electron undergoing a bremsstrahlung event.

While an energy-dependent enhancement factor (lower part of Equation 3) can achieve a further $\sim$20% increase in efficiency over a constant enhancement factor (Equation 3 upper part), it requires a judicious choice of both `C` and `N`. Thus, unless you are prepared to do a detailed study of simulation efficiency, we recommend setting `N`$\leq$0 and using a constant

enhancement factor, BCSE_FACTOR=C. The discussion on optimizing BCSE below assumes you are using a constant enhancement factor.

6.5.2 Simulation optimization with BCSE

For maximum efficiency gains, BCSE should be used with either UBS or DBS. BCSE+UBS is suitable for 4π-geometry simulations (e.g. use of miniature x-ray sources in brachytherapy) whereas BCSE+DBS is suitable for directional geometry simulations (kilovoltage x-ray systems and megavoltage clinical linear accelerators). When BCSE is used with either UBS or DBS, the following two steps can be used to optimize the simulation for production runs.

Step 1: The user chooses an optimum BCSE_FACTOR depending on the simulation type. Recommended values of BCSE_FACTOR are listed in table 1. Getting the maximum efficiency gain is not very sensitive to the exact choice of BCSE_FACTOR because the complementary NBRSPL of the splitting technique should pick up any little difference and get the efficiency gain very close to its maximum. For justification of this argument, see reference[27].

Table 1: Recommended bremsstrahlung cross section enhancement factor (BCSE_FACTOR).

simulation type	incident electron energy range	recommended BCSE_FACTOR
4π geometry (brachytherapy)	kilovoltage range	~500
x-ray tubes	mammography range	~500
x-ray tubes	diagnostic range	~200
x-ray tubes	orthovoltage range	~100
clinical linear accelerators	megavoltage range	~20

Step 2: The user determines the complementary NBRSPL (this applies to both UBS and DBS) by applying Kawrakow's model from reference[30] as follows:

• Perform a few short runs with BCSE_FACTOR from step 1, each with a different NBRSPL value.

• Calculate the efficiency of each run, $\varepsilon_{N_i} = \frac{1}{T_i \ s_i^2}$, where N_i is the splitting number used for run i, T_i is the simulation CPU time for run i, and s_i^2 is the average statistical variance on the scored quantity of interest for run i.

• Fit N_i/ε_{N_i} versus $(N_i - 1)$ to the following quadratic equation:

$$\frac{N_i}{\varepsilon_{N_i}} = A_0 + A_1(N_i - 1) + A_2(N_i - 1)^2 \tag{4}$$

where $A_i, i = 0, 1, 2$ are polynomial coefficients.

• Calculate the optimum splitting number NBRSPL using NBRSPL $= \sqrt{A_0/A_2}$.

Production runs then use BCSE_FACTOR from step 1, and NBRSPL from step 2 for maximum simulation efficiency.

To use BCSE alone (i.e. without UBS or DBS), table 1 should not be used. Instead, the user should perform a few short runs with different values of BCSE_FACTOR (typically much larger than those in table 1), then follow the remainder of step 2 above with BCSE_FACTOR replacing N_i.

6.5.3 Restrictions

The following are restrictions and cautionary remarks that should be considered when using BCSE.

• The user can enhance the bremsstrahlung cross section in one medium only (which may exist in single or multiple geometric regions). This is not an intrinsic restriction, but a restriction due to present coding.

• If more than one geometric region is made of the same medium (e.g. target and jaws made of tungsten), and the user wants to use BCSE for one or more of these regions (e.g. target only), the user needs to: (1) Duplicate the data of the particular medium of interest in the PEGS4 data file (copy and paste), (2) Assign the duplicate data set a different medium name, and (3) Use one medium name for the geometric region(s) where BCSE will be used and the other material name for the all other geometric regions where BCSE will not be used.

• When BCSE is used with UBS, Russian Roulette cannot be ON for one technique and OFF for the other. It must be either ON for both or OFF for both.

• BCSE cannot be used if the electron beam incident on the bremsstrahlung target has variable electron weights.

• BCSE cannot, at present, be combined with Directional Source Biasing (DSB).

6.5.4 Outline of the BCSE algorithm

When BCSE is used *without* UBS or DBS, BEAMnrc implements the following algorithm[27]:

1. The bremsstrahlung cross section in the medium of interest, BCSE_MEDNAME, which is typically the bremsstrahlung target in an x-ray tube or a clinical linear accelerator, is scaled up by a user-supplied factor BCSE_FACTOR.

2. The weight of the resulting bremsstrahlung photons in the enhanced medium is reduced by a factor 1/BCSE_FACTOR.

3. With a probability of 1/BCSE_FACTOR, the energy of the charged particle is decremented by the amount given to the bremsstrahlung photon.

4. Higher generations of charged particles are created through photoelectric, Compton and pair production interactions of first-generation low-weight photons. If the user turns Russian Roulette ON, these higher-order charged particles are eliminated throughout the full geometry

with a survival probability 1/BCSE_FACTOR. The weight of the surviving charged particles is increased by a factor BCSE_FACTOR. If Russian Roulette is OFF, higher-order charged particles are tracked individually.

5. Fat photons would be created through relaxation events after electron impact ionization (both in the enhanced medium and elsewhere), through bremsstrahlung events outside the enhanced medium, and through positron annihilation events. In each of these events, photons are split, according to the UBS and DBS algorithms, into BCSE_FACTOR photons, each with a reduced weight of 1/BCSE_FACTOR.

When BCSE is used *with* UBS or DBS, BEAMnrc implements an algorithm similar to the one above, except that the splitting number is made equal to NBRSPL only when a bremsstrahlung event is about to take place in the enhanced medium, and reset to (BCSE_FACTOR · NBRSPL) for all other aspects of the BCSE and splitting algorithms. See reference[27] for more on the implementation details.

7 Phase Space Files

This section describes the phase space files output by BEAMnrc and the utilities available for processing them.

7.1 Description of Phase Space Files

A phase space file contains data relating to particle position, direction, charge, *etc.* for every particle crossing a scoring plane. Phase space files can be output for each scoring plane in an accelerator. The input parameter IO_OPT controls whether or not phase space files are created. For more information on IO_OPT see section 10.4 dealing with this variable below.

The phase space files are binary and, thus, byte order (big endian or little endian) will depend on which type of architecture they were written on. The utility program readphsp can be used to swap bytes between these two formats.

The first record in a phase space file is different from the others and contains the following information.

MODE_RW The file mode: it can be either 'MODE0' or 'MODE2' depending on whether ZLAST is included in the phase-space parameters.

NPPHSP The total number of particles in the file.

NPHOTPHSP The total number of photons in the file.

EKMAXPHSP The maximum kinetic energy of the particles stored in the file.

EKMINPHSPE The minimum electron kinetic energy (MeV).

NINCPHSP The number of particles incident from the original (non-phase space) source used to generate the phase space file.

Where a phase space file is generated by a simulation using a phase space file as the source (ISOURC=21), in the output phase space file, NINCPHSP, the equivalent number of particles incident from the original non-phase space source, is equal to

$$\texttt{NINCPHSP} = \frac{\texttt{IHSTRY} + (\texttt{NRCYCL}+1)\,(\texttt{NPASS\_ph\_sp} + \texttt{NFAT\_ph\_sp})}{\texttt{NPPHSP}_{\texttt{source}}} * \texttt{NINCPHSP}_{\texttt{source}}$$

where IHSTRY is the number of histories run, NPASS_ph_sp is the number of particles rejected from the source because they were multiple passers, NFAT_ph_sp is the number of fat photons rejected from the source (only if directional bremsstrahlung is used–see section 6.3.4), and NRCYCL is the number of times each particle in the source is recycled (see section 4.13). Thus, regardless of the number of upstream phase space sources, the value of NINCPHSP in a phase space file is always traceable back to the number of histories from the primary, non-phase space source. Since quantities scored during a simulation that uses a phase space source are normalized by NINCPHSP, they, too, are traceable back to the original, primary source. For example, consider a model of a ^{60}Co unit which is broken into 2 components. In the first part, the source capsule is modelled and a phase space file created with all the particles leaving the surface of the capsule. Here all outputs are normalized per photon from the ^{60}Co capsule. In the second stage, the phase space file from the first part is used as input to a model of the collimator system. Here again the outputs are normalized per photon from the ^{60}Co capsule, thus automatically maintaining the "natural" normalization.

Each record in a phase space file contains the following information about the particle scored (in this order):
`LATCH, E, X, Y, U, V, WT, [ZLAST]`
where:

LATCH contains the particle charge, IQ, the number of times the particle has crossed the scoring plane, NPASS, and information which allows the particle's history to be traced (see section 8 below). Note that because it is also used to store IQ and NPASS, the value of LATCH written to the phase space file is not the same as that during transport.

E is the particle total energy (kinetic plus rest mass) in single precision. This is set negative if this is the first particle scored from a new primary (ie non-phase space source) history.

X is the particle X-position (cm).

Y is the particle Y-position (cm).

U is the X-direction cosine.

V is the Y-direction cosine.

WT is the particle's weight; WT also carries the sign of W, the Z-direction cosine, W.

ZLAST For scored photons, this is the Z-position of last interaction; for charged particles, it is the Z-position where the charged particle or its ancestor was set in motion by a photon (*i.e.*, it does not flag the creation site of delta rays). If a particle does not interact, then ZLAST stores Z-position on entering the simulation (*i.e.*, ZIN). This quantity is in brackets because its inclusion in the phase space file depends upon the setting of the input variable IZLAST (see section 10.6 on IZLAST).

The magnitude of the Z-direction cosine, W, is determined from $W = \sqrt{1-(U^2+V^2)}$. The particle's charge is stored in bits 29/30 of LATCH and recovered when read in.

We set E negative for the first particle scored by each new primary (non-phase space) history in order to be able to group scored quantities (energy deposited, fluence, etc) according to primary history when a phase space file is used as a source. This is necessary to account for correlations between incident particles and ensures a correct estimate of the uncertainty on the scored quantities [17]. The negative E marker is propagated to phase space files generated using a phase space source, so that, even in these second-generation files, particles can be grouped according to the original primary histories. When a negative E is read from a phase space source, the primary history counter is incremented and the particle's energy is set to |E|.

If you are using an old phase space file without negative E markers as a source, scored quantities will be grouped according to incident particle instead of primary history. BEAMnrc will output a warning that uncertainties may be underestimated because correlations between incident particles cannot be accounted for. The underestimates in uncertainty caused by not taking correlations into account may be significant in cases where variance reduction techniques result in many correlated particles scoring quantities in the same volume (dose) or area (fluence) [17]. An example of a case where this could happen is when directional bremsstrahlung splitting is used with a large splitting number (see section 6.3.4).

When BEAMnrc writes a phase space file, it opens the file using
`ACCESS = 'direct', FORM = 'unformatted', RECL = 'length'`.
The `FORM='unformatted'` statement ensures that the file will be stored in binary format. Specification of the record length, 'length', depends on the machine being used. For most architectures, 'length' is the number of bytes/record (*i.e.*, 28 or 32 with ZLAST). However, given that some architectures specify record length in 4-byte units, 'length' is represented internally as 7*$RECL_FACTOR or 8*$RECL_FACTOR (with ZLAST), where $RECL_FACTOR is 4 or 1, depending on architecture. Note that $RECL_FACTOR is defined in
`$HEN_HOUSE/lib/${my_machine}/machine.macros`.

The format of phase space files opened with
`ACCESS = 'direct', FORM = 'unformatted'` is called 'MODE0' if ZLAST is not scored and 'MODE2' if ZLAST is scored. The 'MODE0' or 'MODE2' identifier is the first quantity written to the first record of a phase space file.

Phase space files have extension `.egsphsp#` where # is the number of the scoring plane in the accelerator. By default, the maximum number of scoring planes is 3, but this can be adjusted as described in section 2.11.

BEAMnrc makes use of macros stored in `$HEN_HOUSE/utils/phsp_macros.mortran` to read and write the phase space files. Currently, we have optimised the writing macros to store

up and write 1000 particles at a time. This has been found to save network time. Certainly, there are other read/write optimisations that could be made, however, in a normal accelerator simulation, only a small fraction of the simulation time is taken up reading from and writing to phase space files. The source for these macros is kept separately to preserve commonality between the methods used by all Mortran applications (BEAMnrc, DOSXYZnrc, BEAMDP, etc) to access phase space files. In addition, they may be used in the user's own applications.

7.2 Maximum Size of Phase Space Files

Note that the variable written to header storing the number of particles in a phase space file, `NPPHSP`, is a 4-byte integer. This puts a practical limit to the number of particles that can be stored in a phase space file of 2^{31}-1. To put this into perspective, a file without `ZLAST` containing this many particles would be ∼60 GBytes, and a file storing `ZLAST` would be ∼69 GBytes. It is theoretically possible to remove this limit on phase space file size and either:

- not have the number of particles in the file reflected in `NPPHSP` in the header, which will require other modifications to codes using this phase space file as a source since, currently, BEAMnrc (and other EGSnrc user codes) compares the number of histories run to `NPPHSP` in order to determine whether the source needs to be restarted from the beginning or not, or
- redefine `NPPSHP` as a 4-byte real variable, in the process losing precision in terms of the actual number of particles stored in the file.

Given these trade-offs, and short of redefining the format of the phase space header, we prefer to keep 2^{31}-1 as a hard limit and suggest that if you require more data in a phase space source that you run a BEAM shared library source instead (see Section 2.3.1).

7.3 Directory for Phase Space Output

By default, phase space files from an accelerator simulation are written to the directory `$EGS_HOME/BEAM_myaccel` (*i.e.* the accelerator directory). However, you do have the option to output phase space files to a different directory. This is particularly useful if you are writing large phase space files and, due to disk space limitations, you want to write them directly to a `/temp` storage directory on another machine (provided it is connected with NFS to the machine(s) you are running on).

BEAMnrc provides two different ways to change the phase space output directory:

In the `$OMEGA_HOME/beamnrc/beamnrc_user_macros.mortran` file, you can redefine the macro `$DIRECTORY-FOR-PHSP`, which is currently set to `$EGS_HOME/BEAM_myaccel`, to to be the directory you want to write phase space files to. Note that you must provide the full directory path in single quotes (*i.e.* `'/full path to directory'`). An example is given in the `beamnrc_user_macros.mortran` file. Once you have made this change, you must recompile your accelerator to put it into effect.

BEAMnrc also has a built-in custom user input that you can use to set the variable PHSP_OUTDIR, which redefines the output directory for phase space files. The input line is:

```
Phsp output directory= /full path to phase space output directory
```

This must appear in the accelerator .egsinp file in the custom input block delimited by :Start user inputs: and :Stop user inputs:. The custom input block can be placed either just before or just after the EGSnrc transport parameters (see Section 12 for more about custom inputs). If the input is left blank or omitted entirely, then the phase space output directory defaults to that defined by the $DIRECTORY-FOR-PHSP macro in beamnrc_user_macros.mortran (see above). Note that the GUI does not give you access to this input. This method of changing the output directory for phase space files has the advantage that the accelerator does not need to be recompiled whenever the output directory is changed and that different input files for the same accelerator can specify different phase space output directories.

7.4 IAEA-format phase space data

Provided that the user's system has a working C++ compiler (determined automatically on BEAMnrc installation), then BEAMnrc has the capability to read/write phase space data in IAEA format. This allows the user to add to or make use of data from the IAEA's online accelerator phase space database at:
www-nds.iaea.org/phsp/phsp.htmlx

7.4.1 IAEA format

A complete description of the IAEA phase space format is given in IAEA's technical report INDC(NDS)-0484[31], which is available from the online phase space database indicated above. The format is quite flexible in terms of the amount of data that is stored. This section gives a brief description of the IAEA data that is output and read by BEAMnrc.

Phase space data in IAEA format is contained in 2 files: the data file (with extension .IAEAphsp) and the header file (with extension .IAEAheader). IAEA-format header and phase space data files output by BEAMnrc have the names inputfile.#.IAEAheader and inputfile.#.IAEAphsp, where # is the scoring plane number.

The IAEA header file output by BEAMnrc contains the following data:

$CHECKSUM: The size of the .IAEAphsp in bytes. This is used when opening an IAEA (to use it as a source or to add more data to it) to make sure that there are no errors in the file.

$RECORD_CONTENTS: A block indicating the data that is stored in the .IAEAphsp file. A "1" beside a variable indicates that this is stored in the .IAEAphsp file. This block also indicates how many extra long integers and extra floating point variables are stored. In the case of an IAEA-format phase space file written by BEAMnrc, LATCH is always stored as an extra long integer variable. In addition, if the input IZLAST=1

(See Section 10.6) then ZLAST, the Z of the last interaction site for photons and the creation site for secondary charged particles, is stored as an extra floating point variable.

$RECORD_CONSTANT: Values which are set to a constant value. In the case of an IAEA file written by BEAMnrc, the Z of the scoring plane is stored here (note that this information is not available in a standard BEAMnrc phase space file).

$RECORD_LENGTH: The length of a phase space record in bytes.

$BYTE_ORDER: "1234" for little endian machines, "4321" for big endian machines.

$ORIG_HISTORIES: No. of primary histories used to generate the phase space data.

$PARTICLES: Total no. of particles represented in phase space data.

$PHOTONS: No. of photons represented in phase space data.

$STATISTICAL_INFORMATION_PARTICLES: Total weight, min. weight, max. weight, average total energy, min. total energy and max. total energy for each particle type

$STATISTICAL_INFORMATION_GEOMETRY: min. and max. X and Y of all particles scored.

There are other fields in the header but they are not currently used by BEAMnrc.

The phase space data file (.IAEAphsp) output by BEAMnrc consists of a record of data for each particle scored, where a record consists of:

type: particle type and sign of Z-direction cosine, W (CHAR*1)

E: total energy of the particle (REAL*4). If this is the first particle scored from a primary history, then, similar to standard BEAMnrc format phase space data, E is set negative.

X: X position of the particle (REAL*4)

Y: Y position of the particle (REAL*4)

U: X-direction cosine (REAL*4)

V: Y-direction cosine (REAL*4)

WT: statistical weight of the particle (REAL*4)

LATCH: value of LATCH variable (INTEGER*4)

ZLAST: Z of last interaction site of photons or site of creation of secondary charged particles (REAL*4). This is only output if the input variable IZLAST=1 (See Section 10.6).

MU: Fractional monitor unit index associated with the particle (REAL*4). MU is a number on [0,1] chosen at the beginning of each primary history and used to determine the opening coordinates (field) for synchronized component modules (CMs). It is scored automatically if there is/are one or more synchronized CMs with time-varying opening coordinates in the accelerator. Scoring of MU allows synchronization between the simulation generating the file and any downstream simulations, also having synchronized CMs, that use the file as a source. See Section 15 for more information about synchronized CMs.

Thus, each record is 29 bytes long without ZLAST and 33 bytes long if ZLAST or MU is output, and 37 bytes if both ZLAST and MU are output.

Similar to standard BEAMnrc phase space files, a negative energy (E) marker is used to indicate the first particle scored from a new primary history. This allows quantities scored (dose, fluence, etc) by correlated particles to be grouped for statistical analysis when the IAEA phase space file is used as a source. On reading a negative energy value, the primary history counter in BEAMnrc is incremented by one, and E is set to its absolute value.

Note that, unlike standard BEAMnrc phase space files, LATCH is not modified to carry the particle charge and NPASS before being written to the phase space file. The charge is stored in type and if this particle has already crossed the scoring plane (NPASS=1) then it is simply not written to the file.

7.4.2 Writing IAEA phase space data

By setting the input variable IO_OPT=4 (See Section 10.4) you will output the IAEA phase space data and header files at each scoring plane.

7.4.3 Reading IAEA phase space data

If you are using IAEA-format space data as a source (*i.e.* ISOURC=21. See Section 4.13 above.) then you must specify the full name of the phase space data (.IAEAphsp) file. BEAMnrc will automatically detect the .IAEAphsp extension and read the data in IAEA-format (with an output message indicating this to the user). Note that the header file is assumed to be in the same directory as the phase space data file.

As with BEAMnrc format phase space sources, the Z-direction cosine, W is calculated from $\sqrt{U^2 + V^2}$. The sign of W is determined from type, the CHAR*1 variable storing particle type. First, type is converted to INTEGER*2, and then its sign is applied to W, after which the integer form of type is set to its absolute value.

Note that BEAMnrc automatically handles 3-D IAEA format phase space sources in which the (X,Y,Z) position of each particle is stored and 4-D phase space data in which fractional monitor unit index (MU) is stored. In terms of 3-D phase space data, there are restrictions on which component modules the source can be incident within. See section 4.13 above for more details.

7.5 readphsp

There are several programs and routines in place for processing phase space files. One of the most basic of these is the program `readphsp` located on `$OMEGA_HOME/progs/readphsp`. `readphsp` is invoked by:

```
readphsp inputfile outputfile
```

where `inputfile` includes the `.egsphsp#` extension and is the name of the phase space file to be converted, and `outputfile` is the name of the file which will contain the converted data (and must also include any extension you want to add). Note that `readphsp` retains compatibility with phase space files having `.egs4phsp#` extensions generated by EGS4 versions of BEAM.

`readphsp` gives the user various file conversion options (*eg.* from direct access mode to sequential access mode). It can also convert phase space files into a format readable by the CERN program PAW provided that the necessary libraries are present (see below).

`readphsp` can be used as a byte swapping tool so that phase space files generated on another type of machine become compatible with the type of machine that `readphsp` is being run on.

The `readphsp` program also allows extraction of a subset of phase space data from a file based on charge and/or maximum number of particles.

Before converting phase space files `readphsp` prints a summary of the file contents. If this is nonsense, then it is very likely the file was generated on a machine with a different byte order. The user should use the option of `readphsp` to swap the bytes of the phase space data to make it compatible with the current machine before otherwise manipulating phase space data. When using `readphsp` for byte swapping, the name of the output file can be the same as that of the input file however the swapped data will overwrite the original data.

Files related to `readphsp` are on `$OMEGA_HOME/progs/readphsp`.

Makefile Directs compilation of `readphsp`. Sources required files such as `$HEN_HOUSE/lib/config/machine.macros` (where `config` is the name of your configuration) for the record length factor (`$RECL-FACTOR`) for 4-byte phase space data on your particular configuration, and `$OMEGA_HOME/beamnrc/phsp_macros.mortran` for phase space reading/writing macros that `readphsp` makes use of. You must modify `Makefile` if you want to compile `readphsp` with PAW libraries (see below) so that phase space data can be converted into PAW format. Clear instructions for doing this are given in the `Makefile`.

readphsp.mortran the `MORTRAN` source file.

libpawlib.a, libpacklib.a (optional) Libraries required for the PAW format conversion. These are not included with the distribution because they are proprietary from CERN, but they are available for free from CERN for those working on medical research.

dummy.f which is a routine required to compile `readphsp` when there are no PAW libraries present.

Normally, `readphsp` is compiled as part of the BEAMnrc installation. However, it can be compiled separately by going into `$OMEGA_HOME/progs/readphsp` and typing `make`. This will leave behind the executable, `readphsp*`, in the `$HEN_HOUSE/bin/config` directory.

Note that `readphsp` does not work with IAEA-format phase space data.

7.6 BEAMDP

Another important program for processing phase space files is the `BEAMDP` program. For more details about running `BEAMDP` to derive energy, planar fluence, mean energy and angular distributions from a phase-space file please see the report "BEAMDP as a General-purpose Utility"[13].

8 Tracking a Particle's History using LATCH

The `LATCH` variable, associated with each particle in a simulation, is a 32-bit variable used to track the particle's history. In the input files there is an opportunity to define a mapping from geometric regions to bits (*i.e.* define bit regions) using the `IREGION_to_BIT` variable. Thus, *eg.* it is possible that bit 5 corresponds to geometric region 3, and more importantly, one bit, say 3, can correspond to multiple geometric regions, *eg.* 1,5,8. This is convenient for accelerator structures comprising multiple regions. All regions which are not explicitly associated with a bit number by the user are assigned bit number 23 by default. Each bit of the `LATCH` variable is designated as follows:

bit 0 Set to 1 if a bremsstrahlung or positron annihilation event occurs in the history; 0 otherwise(not used for `LATCH_OPTION` = 1).

bit 1-23 Used to record the bit region where a particle has been and/or has interacted (Note that the bit set for a region is determined by `IREGION_TO_BIT` for that region)

bit 24-28 When downshifted by 24 bits (see paragraph below), these store the bit region in which a secondary particle is created; if no bit number is stored, the particle is a primary particle. Note these bits are not used if `LATCH_OPTION` = 1.

bit 29-30 Store the charge of a particle when `LATCH` is output to a phase space file (see section 7 on phase space files). During a simulation, bit 30 is used to identify a contaminant particle but this information is not output to the phase space file. Set to 1 if the particle is a contaminant particle; 0 otherwise. Note that if `LATCH` is not inherited (*i.e.* when `LATCH_OPTION` = 1), bit 30 loses its meaning.

bit 31 Set to 1 if a particle has crossed a scoring plane more than once when `LATCH` is output to a phase space file (see section 7 on phase space files above)

For secondary particles, recording the bit region in which they were created in bits 24-28 is equivalent to multiplying the bit region number by 2^{24}, or 16777216. Thus, to retrieve the

bit region of origin of a secondary particles, the LATCH value of the particle must be divided by 16777216 (*i.e.* taking the value INT(LATCH/16777216)).

The user controls the protocol for setting LATCH using the LATCH_OPTION input variable. The possible settings of LATCH_OPTION are:

LATCH_OPTION = 1 (Non-Inherited LATCH Setting): LATCH bits 1-23 store bit regions where a particle has been. Secondaries do not inherit LATCH values from the primaries that created them (*i.e.*, bits 1-23 of a secondary particle carry no information about the the transport history of its ancestors). This option must NOT be used if ICM_CONTAM is non-zero since the ICM_CONTAM option needs bit 30.

LATCH_OPTION = 2 (Comprehensive LATCH Setting –default): Bits 1-23 store bit regions where a particle has been. LATCH values are passed on to secondary particles from the primaries that created them. Thus, bits 1-23 for a secondary particle include all bit regions in which the secondary particle has been plus those in which its ancestors have been up to the point where the secondary was created. Bits 24-28 are used to record the bit region where a secondary particle is created. Bit 0 is set if a bremsstrahlung photon is involved in a particle's history.

LATCH_OPTION = 3 (Comprehensive LATCH Setting 2): Similar to LATCH_OPTION=2, but, for a photon, bits 1-23 record the bit regions in which the particle has interacted, rather than simply where it has been. Secondary particles resulting from these interactions have bits 1-23 set for the bit regions in which they were created.

To clarify further the setting of LATCH bits under various LATCH_OPTION values, fig 21 and table 2 summarize the situation for a simple photon accelerator. Two electrons enter the simulation and produce bremsstrahlung photons in the target (bit region 1). One photon goes all the way to the scoring plane without interacting and the second undergoes a pair production event in the flattening filter (bit region 3). The electron escapes from the flattening filter an gets to the scoring plane and the positron annihilates in the flattening filter and one of the resulting 511 keV photons gets to the scoring plane. The user has also defined a contamination scoring plane just above the JAWS (*i.e.* ITDOSE_ON=1, ICM_CONTAM=4 (just above the JAWS) and IQ_CONTAM=-1).

For LATCH_OPTION=1 (non-inherited), bits 1-23 are set for bit regions where a particle has been. Note that all particles have bit 23 set because they have all been in air, which defaults to bit region 23 unless the user explicitly sets it to some other bit region number. Bits 24-28 store nothing because the bit regions where secondary particles are created are not recorded. Contaminant particles are not marked, so bit 30 is not used, and bremsstrahlung events are not recorded in bit 0.

For LATCH_OPTION=2 (comprehensive inherited where been), bits 1-23 for a particle are set for bit regions where it and all of its ancestors have been. Thus, all particles have bit 1 set since they are all descendants of the bremsstrahlung interaction in bit region 1. Particles 2 and 3 both have bit 3 set since they are descendants of the pair event in bit region 3. Again, bit 23 is set for all particles because they (and their ancestors) all pass through air. In addition, bit 0 is set for all particles because they (or their ancestors) are products of a bremsstrahlung event (in bit region 1). Bits 24-28, when downshifted by 24 bits, store the

bit region where secondary particles were created. Thus, these bits store the number "3" for particles 2 and 3 and the number "1" for particle 1. Finally, bit 30 is used to identify contaminant particles and their descendants, where, in this example, contaminant particles are defined as any charged particle crossing the contaminant plane shown. Thus, bit 30 for particle 3, an electron reaching the scoring plane from above the contaminant plane, is set.

LATCH_OPTION=3 (comprehensive inherited where interacted) is similar to LATCH_OPTION=2 except that bits 1-23 are set for bit regions in which photons (and ancestral photons) have interacted, as opposed to regions they have simply passed through. Note that charged particles interact at every step (although only a small fraction of these interactions are explicitly simulated!) and so they can be treated the same as with LATCH_OPTION=2. In the example shown, the only difference in bit settings between LATCH_OPTION=2 and LATCH_OPTION=3 is that bit 23 is not set for particles 1 and 2, because these photons have not interacted in air, nor are they descendants of photons that have interacted in air.

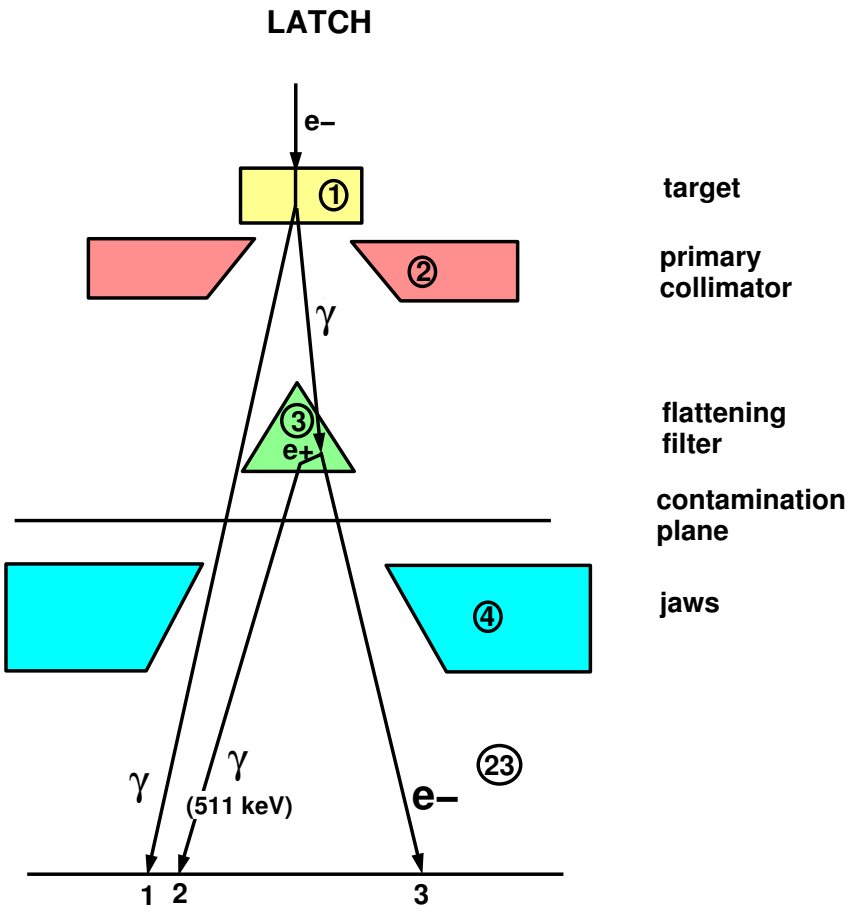


Figure 21: Simple photon accelerator model showing 3 particles reaching the phase space file after 2 electrons are incident. Contamination is defined as charged particles crossing the contamination plane. Table 2 shows the bit settings in LATCH as the particles reach the scoring plane. The bit numbers associated with the geometric regions (bit regions) are shown in circles.

Table 2: Bit settings in LATCH for the simple example shown in fig 21. In the case of bits 24-28, the table gives the numbers stored after downshifting by 24 bits.

Latch option	Particle	Bit 0	1	2	3	4	23	24-28	30
1 non-inherited	1	0	1	0	0	0	1	0	0
	2	0	0	0	1	0	1	0	0
	3	0	0	0	1	0	1	0	0
2 comprehensive	1	1	1	0	0	0	1	“1”	0
inherited	2	1	1	0	1	0	1	“3”	0
where been	3	1	1	0	1	0	1	“3”	1
3 comprehensive	1	1	1	0	0	0	0	“1”	0
inherited	2	1	1	0	1	0	0	“3”	0
where interacted	3	1	1	0	1	0	1	“3”	1

9 Calculating Dose Components

The ability to trace a particle's history using LATCH also allows doses to be broken down into their components. In any dose zone, BEAMnrc is able to break dose down in 2 ways: dose from contaminant particles (identified on the basis of their charge only); dose arising only from particles with a certain user-specified combination of LATCH bit settings (this is called "bit filtering").

The input variables associated with dose component calculations are:

ITDOSE_ON 0 to calculate total dose only (default); 1 to calculate dose components

The following variables are associated with contaminant dose calculation and are only required if ITDOSE_ON = 1.

ICM_CONTAM contaminant particles are identified upon entering the top of CM number ICM_CONTAM, where $1 \leq$ ICM_CONTAM $\leq$ total number of CMs. If ICM_CONTAM is set to 0, no contaminant dose will be calculated.

IQ_CONTAM The charge of the contaminant particles (0 for photons and 1 for charged particles); all particles with this charge will be marked as contaminant particles (by setting LATCH bit 30) upon entering the front of CM number ICM_CONTAM.

Contaminant dose is scored in every dose zone and is the dose due to particles having the contaminant charge entering CM number ICM_CONTAM and their descendants. For example, if IQ_CONTAM = 1, all the charged particles entering CM number ICM_CONTAM and their descendants will contribute to contaminant dose. Note that if LATCH_OPTION = 1 LATCH values are not transferred to descendants (secondaries), and contaminant dose calculations will be meaningless. Thus, the contaminant dose option is automatically turned off if LATCH_OPTION = 1.

The following variables are associated with bit filtering of dose and are only required if ITDOSE_ON = 1:

LNEXC number of dose components using an exclusive LATCH bit filter. Note that 0 <= LNEXC <= $MAXIT - 3, where $MAXIT can be changed from its default value of 12 (as described in section 2.11).

L_N_EXC(I,31) (I = 1,2,...,LNEXC) Exclusive bit filter for dose component I. An exclusive bit filter consists of a sequence of up to 31 bit numbers (range 1-31) on a single record separated by commas. Particles with any of the bits set are excluded from dose component I.

LNINC number of dose components using an inclusive/exclusive LATCH bit filter. Note 0 <= LNINC <= $MAXIT - LNEXC - 3.

L_N_INC(I,31) (I = 1,2,...,LNINC) Inclusive/exclusive bit filter for dose component I+LNEXC (inclusive/exclusive components are listed after exclusive components). An

inclusive/exclusive bit filter consists of up to 31 bit numbers (range 1-31) split into 2 groups input on one record. Bit numbers are separated by commas, with ",0," denoting the end of the first (inclusive) group and the beginning of the second (exclusive) group. Dose component I+LNEXC then includes contributions from particles with `LATCH` having any of the bits in the first group set and none of those in the second group set. For example, if the input for `L_N_INC(I,31)` is 2,3,5,0,1,4, then dose component I will include the contributions from the particles with **any** of `LATCH` bits 2, 3 or 5 set **and** neither of bits 1 and 4 set. The status of other `LATCH` bits will have no effect on this dose component. If the inputs are 1,2,3,0,0,...,(equivalently 1,2,3,) the dose component will include the contributions from the particles with any of `LATCH` bits 1, 2 or 3 set, and there are no exclusive bit settings.

Note that bit 0 of `LATCH` can not be used for the above dose component calculations.

Bit filtering of dose provides a particularly powerful tool for determining dose contributions. In view of the information stored in `LATCH` (see section above), dose contributions can be separated according to:

1. What regions particles have passed through/interacted in
2. Whether the particle is a primary or secondary
3. Where a secondary was created
4. Whether or not the particle is a contaminant
5. Any combination of the above

It should be noted that if directional bremsstrahlung splitting (DBS) is used in a BEAM simulation then an additional dose component, the total dose minus the dose due to fat particles, is automatically output to the `.egslst` and `.egsplot` files. see section 6.3.4 for more details.

10 Other Input Variables

This section provides descriptions of main BEAMnrc input variables not covered above.

10.1 IWATCH

`IWATCH` controls output to the screen (interactive run) or to the `.egslog` file (batch run) during beam execution. The possible settings are:

= 0 (default) On completion of each batch, outputs information about the batch (eg elapsed time, CPU time, elapsed/CPU time, random number used to begin batch, # of random seeds used in the batch, total number of histories run up to and including the current batch, the total number of particles scored in the first phase space file)

= **1** Outputs batch information plus, after every particle interaction, outputs complete information about the particle(s) involved (eg interaction type, particle type, position of particle on stack, particle energy, X-Y-Z position of particle, U-V-W direction cosines, `LATCH` value of particle, region # of particle); also informs user when a particle is being discarded, when a particle is passing from one CM to another

= **2** Similar to 1 only complete particle information is output at every step; also outputs total dose in a region whenever energy is deposited there

= **3** Similar to 2 only particle and dose information are output whenever dose is deposited.

= **4** Outputs same information as 0 plus `.egsgph` and `egsgeom` files for graphical representation of the accelerator and particle paths using `EGS_Windows`[16].

10.2 ISTORE

ISTORE controls how BEAMnrc reads and writes the state of the random number generator (RNG) used in simulations. The possible settings of ISTORE are:

= **0** (default) BEAMnrc stores the state of the RNG for the 1st history in each batch in the file `.egsrns`

= **1** BEAMnrc stores the state of the RNG for every history in the .egsrns file. The random number state of the current history overwrites that of the previous history.

= **-1** BEAMnrc reads the state of the RNG for the 1st history from the `.egsrns` file. If there was an error during a history that caused the history to stop, then you can use `ISTORE` = -1 to rerun the problem history, provided that you first stored the state of the RNG for each history using `ISTORE` = 1.

10.3 IRESTART

IRESTART controls how BEAMnrc treats the current run - either as new or as a continuation of a previous run. The possible settings of IRESTART are:

= **0** (default) BEAMnrc initiates a new run, deleting all of the output files from previous runs of the same name if present (`.egslog`, `.egslst`, `.egsrns`, `.egsphsp1`, *etc.*).

= **1** Restart of a previous run; BEAMnrc reads fluence and dose data from the previous run, the number of histories already run, the time taken by the previous run, and the state of the random number generator from the last batch of the previous run from the `.egsdat` file; fluence and dose and their statistics are averaged with those from the previous run; phase space output is appended to that from the previous run; note that the number of histories to run in the present run is given by NCASE but the reported total CPU time and number of histories analysed in the output include the histories and CPU time from the previous run.

= 2 BEAMnrc creates the `.egsinp` file and then exits without running the simulation.

= 3 BEAMnrc reads dose and fluence data from a previous run in the `.egsdat` file, performs statistical analysis on the data and outputs results to `.egslst`; no simulation is run.

= 4 For analysis of dose and fluence in a simulation that has been split up into parallel jobs. See section 13 page 133 below on parallel processing with BEAMnrc for more details.

10.4 IO_OPT

`IO_OPT` allows user control of phase space output. Some aspects of this variable have already been covered in section 7 on phase space files. However, it also controls data analysis of simplified source models. The possible settings for `IO_OPT` are:

= 0 (default) Phase space output at each scoring plane

= 1 No phase space output at scoring planes

= 2 No phase space output, but perform data analysis for simplified source models (see BEAMDP manual).

= 3 Phase space output for the first 100,000 particles across each scoring plane and perform data analysis for simplified source models

= 4 IAEA-format phase space output at each scoring plane

10.5 IDAT

`IDAT` controls output of the `.egsdat` file, which contains data from a previous run so that BEAMnrc is able to restart this run if required. The `.egsdat` files from parallel jobs are required for recombining to obtain the final results (see section 13). Possible settings of `IDAT` are:

= 0 (default) Output energy deposited and fluence data, number of histories completed, elapsed time and the current state of the random number generator to the `.egsdat` file upon completion of every batch

= 1 Do not output a `.egsdat` file

See section 2.10 for a more complete description of the `.egsdat` file.

10.6 IZLAST

IZLAST controls whether or not ZLAST is recorded in phase space files. ZLAST is the Z-position of last interaction for photons and is the Z-position of where an electron or its ancestor was set in motion by a photon (i.e. it does not flag the creation site of delta rays). The possible settings of IZLAST are:

= 0 (default) BEAMnrc does not score ZLAST in phase space files; this means that phase space files are read and written in compressed "MODE0"

= 1 BEAMnrc scores the Z position of last interaction in phase space files; phase space files are read and written in compressed "MODE2." "MODE2" files are approximately 14% larger than "MODE0" files; when plotting phase space data with paw, ZLAST replaces the radial position of a photon (r)

= 2 Same as option 1 and, in addition, BEAMnrc writes the X-Y-Z positions of the last site of interaction for photons into the .egsgph file. These last sites of interaction can be viewed in 3-D using EGS_Windows[16]. Note that IWATCH=4 must not be turned on at the same time since particle tracks are also written to the .egsgph file.

Note that the utility BEAMDP has an option to plot graphs of ZLAST in 2-D.

10.7 NCASE

NCASE is the number of histories for this run. Minimum value is 100. Default is 100 if NCASE is set < 100. The number of histories per batch is equal to NCASE/($NBATCH), where $NBATCH is the number of output batches. $NBATCH is set in the beamnrc_user_macros.mortran file and is currently equal to 10. The total number of histories after this run will be the sum of NCASE and the number of histories from the previous run.

Starting with BEAMnrc, the breaking of the run into batches is for saving intermediate data and diagnostics. It is no longer associated with statistical uncertainties.

Starting with BEAM99, NCASE and other variables related to the number of histories, such as IHSTRY, were made INTEGER*8 variables. This allows >2x10^9 total histories, and is particularly useful for restarts and parallel runs, where a large number of total histories can be accrued quite easily. However, INTEGER*8 may cause problems for some Fortran compilers. For a more detailed description of this problem and how to fix it if it does occur, see section 18 on known bugs/restrictions in BEAMnrc.

10.8 IXXIN, JXXIN

The input values of IXXIN and JXXIN are seeds used to initialize the RANLUX[32, 33] or RANMAR[34, 35] random number generator. Within BEAMnrc, IXXIN is limited to the range 0<IXXIN≤31328 (defaults to 1802), and JXXIN has the range 0<JXXIN≤30081 (defaults to 9373). These ranges/defaults were originally designed for RANMAR, however,

if you are using RANLUX (the new default) then IXXIN is the "luxury level" of the random number generator and must be in the range 0<IXXIN≤4, otherwise, it will automatically be set to the default luxury level of 1 (Note: this means that the BEAM default value for IXXIN of 1802 will ultimately get reset to 1 by RANLUX). Also, with RANLUX, the input JXXIN can have a maximum value of 1073741824, however you will have to change the BEAMnrc code to allow this maxium.

Note that IXXIN and JXXIN are only used to initialize the random number generator, and, during a simulation, they no longer reflect the values of the seeds that are actually used to generate the random numbers. During restarts (IRESTART = 1), the state of the random number generator at the end of the previous run is read from the .egsdat file and is used at the beginning of the restart. Thus, a restarted run with a total of, eg, 10000+10000 histories should generate results identical to a single run of the same simulation with 20000 histories. Also note that when running parallel simulations with BEAMnrc (see section 13), IXXIN and/or JXXIN must have different values for each of the individual jobs that make up the simulation. A property of the random number generators being used is that this simple change guarantees random number sequences which are independent.

To switch from the default RANMAR to RANLUX in your BEAM code, either go into $HEN_HOUSE/specs/beamnrc.spec and change the line:

```
RANDOM = $(EGS_SOURCEDIR)ranmar
```

to:

```
RANDOM = $(EGS_SOURCEDIR)ranlux
```

(recommended if you plan on using RANLUX as your default random number generator for all accelerators), or compile an individual accelerator using the command

```
make RANDOM=$HEN_HOUSE/src/ranlux
```

10.9 TIMMAX

TIMMAX is the maximum CPU time in hours allowed for this run. TIMMAX defaults to 0.99 hours if it is set = 0. At the end of each batch, an estimate is made of how much time is needed to complete another batch. If the next batch cannot be completed within TIMMAX for the current run, then BEAMnrc terminates the run and analyses the results for the shortened run. The restart feature (see section 10.3 on IRESTART) can be used to continue the run if needed and if IDAT (section 10.5) is set to 0 (*i.e.* the .egsdat files are created). Note that the total CPU time reported for a simulation is the sum of the CPU time so far for this run and the CPU time from the previous run, but TIMMAX refers to the present run's time only.

10.10 ECUTIN

The electron transport cutoff energy, ECUT, defines the minimum total energy below which electrons are no longer followed. As soon as an electron's total energy falls below ECUT, its history is terminated and its energy deposited in the current region. The time required for

a given calculation is strongly dependent on the value of ECUT and thus it is important to use as high a value as possible.

The global value of ECUT (in MeV) can be defined through either the Global ECUT input in the EGSnrc input parameters (see Section 11.1 below) or the BEAMnrc input, ECUTIN, with the greater of the two values used. If the Global ECUT input from the EGSnrc input parameters is omitted altogether, then ECUTIN is used.

The user can also override the global ECUT within individual regions of component modules (see CM descriptions below) provided that the value of ECUT for a region is > the global ECUT.

Note that AE for the PEGS4 data set used is the lower limit on the value of ECUT used in a given region. The selection of AE also requires some care and is discussed in section 16.2.

Selection of ECUT is complex in general and is dependent on what is being calculated[36, 26]. For therapy beams, ECUT can be quite high since low-energy electrons contribute little to dose in phantom. For what we consider detailed work, we have used ECUT = 0.700 MeV but much higher may be possible. However, if the dose in the monitor chamber is an important part of the calculation, lower values of ECUT may be required.

As a general rule of thumb for calculations of dose distributions, ECUT should be chosen so that the electron's range at ECUT is less than about 1/3 of the smallest dimension in a dose scoring region. This ensures energy is transported and deposited in the correct region although for electrons which are moving isotropically, this can be a very conservative requirement.

If the global value of ECUT is left blank or set to 0, then it defaults to AE of the default medium, unless the default is VACUUM, in which case it defaults to the maximum AE of all media in the accelerator.

10.11 PCUTIN

PCUT is the photon equivalent of ECUT. As soon as a photon's energy falls below PCUT transport of the photon is terminated and its energy is deposited in the current region.

The global value of PCUT (in MeV) is defined using either the Global PCUT input in the EGSnrc input section (see Section 11.2 below) or the BEAMnrc input, PCUTIN. Again, the greater of the two values is used, and, if the Global PCUT input is omitted from the EGSnrc inputs altogether, then PCUTIN is used. Similar to ECUT, the user can override the global value of PCUT within individual regions of component modules provided that PCUT for that region is > than the global PCUT.

The exact value of PCUT is not critical in the sense that low values do not take much more time. A value of 0.01 MeV should generally be used.

The global value of PCUT defaults to AP of the default medium, unless this is VACUUM, in which case it defaults to the maximum AP of all media in the accelerator.

10.12 ESTEPIN, SMAX, IDORAY, IFLUOR

These are dummy inputs which used to define the maximum fractional energy loss per electron step (ESTEPIN), the maximum step length (SMAX), a switch for turning on Rayleigh scattering (IDORAY), and a switch to turn on K-shell X-ray fluorescence (IFLUOR). Now all of these transport parameters are handled in the EGSnrc inputs (see section 11), however, the dummy inputs are retained for compatibility with older BEAM input files.

10.13 ICM_SPLIT, NSPLIT_PHOT, NSPLIT_ELEC

These inputs are used to split photons and electrons by an arbitrary splitting number upon entering a CM. The CM number is specified by ICM_SPLIT, with the restriction 1<ICM_SPLIT≤MAX_CMs, and NSPLIT_PHOT and NSPLIT_ELEC are the splitting numbers for photons and electrons, respectively. If the splitting number is ≤1 then particles are not split. Once a particle has been split, the resultant particles carry weight 1/NSPLIT_PHOT or 1/NSPLIT_ELEC. A particle is only split upon entering the user-specified CM from the top (*i.e.* W(NP) > 0) and if it has not been split by this option before.

Splitting at an arbitrary plane was designed primarily for improving statistics in dose calculations in a phantom, in which case particles are split upon entering the CM modeling the phantom. Use of this option near the top of an accelerator may produce undesirable correlations, without much gain in efficiency.

The ICM_SPLIT option operates in conjunction with any kind of bremsstrahlung splitting and photon forcing. In the case where ICM_SPLIT=NFCMIN (*i.e.* the arbitrary splitting plane corresponds to the plane at which photon forcing is to begin), the photons are split BEFORE they are forced to interact.

11 EGSnrc Monte Carlo Transport Parameters

EGSnrc allows the user extensive control over the Monte Carlo (MC) parameters governing transport physics in BEAMnrc. The default settings of these parameters in BEAMnrc are currently identical to those for EGSnrc applications in general and will result in accurate simulations at both high (MV) and low (kV) energies. However, if the user is interested only in MV simulations, then CPU time can be saved by adjusting some of these parameters without sacrificing accuracy. These parameters and their recommended settings for MV applications will be noted in the subsections below.

EGSnrc inputs generally appear at the end of a BEAMnrc input file between the delimiters :start mc transport parameter: and :stop mc transport parameter: and have the format:

```
PARAMETER NAME = parameter value
```

If you are using the BEAMnrc GUI, then the EGSnrc parameters (including defaults) are written to the input file when you save it.

If an MC transport parameter does not appear in the input file, then its default value is

used.

The following subsections describe each EGSnrc MC transport parameter. For more information, see the EGSnrc manual[1]. The actual internal variable name associated with each input appears in brackets.

11.1 Global ECUT (ECUT)

`Global ECUT` defines the global electron cutoff energy (`ECUT`) in MeV. If `ECUTIN` in the main BEAMnrc inputs is > `Global ECUT`, or if `Global ECUT` is omitted from the EGSnrc input section, then the global value of `ECUT` is set to `ECUTIN`. See section 10.10 for a more detailed discussion of `ECUT`.

11.2 Global PCUT (PCUT)

`Global PCUT` defines the global photon cutoff energy (`PCUT`) in MeV. If `PCUTIN` in the main BEAMnrc inputs is > `Global PCUT`, or if `Global PCUT` is missing from the EGSnrc input section, then the global value of `PCUT` is set to `PCUTIN`. See section 10.11 for a more detailed discussion of `PCUT`.

11.3 Global SMAX (SMAXIR)

`Global SMAX` defines the maximum electron step length in cm. If the default EGSnrc electron step electron algorithm (see section 11.8) and the exact boundary crossing algorithm are used, then no restriction on maximum step length is needed. However, if using PRESTA-I (the EGS4 standard) as the electron step algorithm or the boundary crossing algorithm, then `Global SMAX` must be set to a reasonable value (eg 5 cm) to ensure proper electron transport in low density materials (air). `Global SMAX` defaults to 5 cm when the PRESTA-I boundary crossing algorithm (BCA) or electron step algorithm is used is used and 1.E10 cm when the EXACT BCA and PRESTA-II electron step algorithm are used.

11.4 ESTEPE (ESTEPE)

`ESTEPE` is the maximum fractional energy loss per electron step. For accurate electron transport with default EGSnrc electron step algorithm, `ESTEPE` should not exceed 0.25 (the default). The value of `ESTEPE` should not be changed unless PRESTA-I is being used as the electron transport algorithm.

11.5 XImax (XIMAX)

`XIMAX` is the maximum first multiple elastic scattering moment per electron step. It is equal to roughly half the average multiple scattering angle squared. Make sure you do not set

XIMAX > 1, since this is beyond the range of available multiple scattering data. The default value of 0.5 should be sufficient for most applications.

11.6 Boundary crossing algorithm (BCA) (bca_algorithm)

This controls the algorithm used to transport electrons across region boundaries. There are two possible settings of Boundary crossing algorithm: EXACT and PRESTA-I. The default for BEAMnrc is EXACT (same as in EGSnrc itself). The EXACT boundary crossing algorithm was introduced in EGSnrc to eliminate a known fluence singularity caused by forcing a multiple scattering event at a boundary [37]. In the EXACT case, electrons are transported in single elastic scattering mode as soon as they are within a distance from the boundary given by the EGSnrc input Skin depth for BCA (see section 11.7 below). If the PRESTA-I BCA is used boundary crossing is carried out in a manner similar to EGS4/PRESTA. Specifically, the lateral correlation algorithm is turned off if the perpendicular distance from the electron to the boundary is less than Skin depth for BCA (see section 11.7 below) and then, once the electron reaches the boundary, a multiple scattering event is forced.

Until 08/06, the default BCA for BEAMnrc was PRESTA-I. However, this has been switched to the slower EXACT BCA because it was found that the use of the PRESTA-I can result in significant overestimates of dose (up to 2.5%) when the CHAMBER component module is used as a depth-dose phantom (see Section 15.3.6). The use of the EXACT BCA is not expected to significantly increase the CPU time in most accelerator simulations. See the technical note by Walters and Kawrakow[12] for more information.

11.7 Skin depth for BCA (skindepth_for_bca)

If Boundary crossing algorithm= PRESTA-I, then Skin depth for BCA is the perpendicular distance (in elastic mean free paths) from the boundary at which lateral pathlength corrections are turned off and the particle is transported in a straight line until it reaches the boundary. In this case, the default is a fixed value calculated by EGSnrc to be the same as that used in the original implementation of PRESTA in EGS4 and depends on the value of ECUT.

If Boundary crossing algorithm= EXACT, then Skin depth for BCA determines the perpendicular distance (in elastic mean free paths) to the region boundary at which electron transport will go into single elastic scattering mode. A skin depth of 3 elastic mean free paths has been found to give peak efficiency in this case and is the default.

If Boundary crossing algorithm= EXACT and Skin depth for BCA is set to a very large number (eg 1e10), then the entire simulation will be done in single scattering mode.

11.8 Electron-step algorithm (transport_algorithm)

This input determines the algorithm used to calculate lateral and longitudinal corrections to account for elastic scattering in a condensed history electron step. There are 2 possible

settings: PRESTA-II (the default) and PRESTA-I. PRESTA-II is the newer, more accurate, algorithm developed for use with EGSnrc[1]. PRESTA-I is the original PRESTA algorithm with some modifications[38]. The original PRESTA-I is known to underestimate lateral deflections, to underestimate longitudinal straggling and to produce a singularity in the distribution describing the lateral spread of electrons in a single condensed history. While PRESTA-I may be accurate enough for high energies (since elastic scattering is weak), it is not recommended for low energy applications.

11.9 Spin effects (spin_effects)

If Spin effects= On (the default), then elastic scattering cross-sections that take into account relativistic spin effects are used in electron transport. If Spin effects= off, then screened Rutherford cross-sections (similar to EGS4) are used for elastic scattering. It should be noted that using Spin effects= on does increase calculation time, however, results are more accurate and it is ABSOLUTELY necessary for good backscatter calculations.

Including spin effects has a small but distinct effect on calculated depth-dose curves. In low-Z materials such as water, the value of R_{50} for a given energy is higher than with EGS4/PRESTA. For high-Z materials it is the reverse and backscatter also increases.

11.10 Brems angular sampling (IBRDST)

This input determines the type of angular sampling that is done when a bremsstrahlung photon is created. The possible settings for Brems angular sampling are Simple and KM (the default). If Simple is used, then bremsstrahlung angles are sampled using only the leading term of modified equation 2BS of Koch and Motz[25, 39]. If Brems angular sampling= KM, then the bremsstrahlung angles are sampled using the entire modified equation. Note that Brems angular sampling= KM is similar to the bremsstrahlung angular sampling scheme used by the latest version of EGS4/BEAM, with some modifications.

Brems angular sampling= Simple is adequate at high energies, however, there is little increase in simulation time associated with using the entire modified 2BS equation, and the entire equation is recommended at low energies.

11.11 Brems cross sections (IBR_NIST)

This input determines the differential cross-section used for bremsstrahlung interactions. If Brems cross sections= BH (the default), then Bethe-Heitler cross-sections (Coulomb corrected above 50 MeV)[39] are used. These cross-sections are those used by EGS4/BEAM. If Brems cross sections= NIST, then differential cross-sections from the NIST bremsstrahlung cross-section data base[40, 41] are used. The NIST cross-sections are the basis for radiative stopping powers recommended by the ICRU[42]. The difference between BH and NIST is negligible for energies $>$ 10MeV, but becomes significant in the keV energy range where the NIST data base is preferred. In either case, the total bremsstrahlung cross sections are the same. There is also a Brems cross sections= NRC option. The NRC cross-sections are

the NIST cross-sections including corrections for electron-electron bremsstrahlung (typically only significant for low values of the atomic number Z and for k/T ¡ 0.005).

11.12 Bound Compton scattering (IBCMP)

The `Bound Compton scattering` input is used to determine whether binding effects and Doppler broadening are simulated in Compton (incoherent) scattering events. It has four possible settings: `Off`, `On`, `Simple` and `Norej` (the default). If this input is set to `Off`, then the Klein-Nishina formula[43] is used to determine differential cross-sections for Compton scattering. This is similar to the treatment of Compton scattering in EGS4/BEAM. If `Bound Compton scattering= On`, then the original Klein-Nishina formula is augmented with the impulse approximation[44] to simulate binding effects and Doppler broadening. Simulation of binding effects and Doppler broadening takes extra time and is only important below 1 MeV and/or if Rayleigh scattering is being simulated (see section 11.18). The `Simple` setting uses the impulse approximation to simulate binding effects but neglects doppler broadening. Finally, the default `Norej` option uses the total bound Compton cross sections (em i.e. no impulse approximation) resulting in no rejected Compton interactions at run time.

For MV applications, `Bound Compton scattering` can be set to `Off` with little impact on simulation accuracy but with a potentially significant decrease in CPU time. Directional bremsstrahlung splitting (see section 6.3.4), in particular, makes use of a more efficient Compton splitting routine if `Bound Compton scattering` is `Off`.

Bound Compton scattering may also be turned `On` in selected regions (`Off` everywhere else) using `Bound Compton scattering= On in regions` together with the inputs `Bound Compton start region` and `Bound Compton stop region` to define the region ranges for which bound Compton is to be turned on. Conversely, bound Compton can be turned off in selected regions (on everywhere else) by inputting `Bound Compton scattering= Off in regions` with `Bound Compton start region` and `Bound Compton stop region` used to define the region ranges where bound Compton is to be turned off. Of course, turning bound Compton on/off in regions is accomplished much more easily in the BEAMnrc GUI. Note that the `Norej` option cannot be used on a region-by-region basis.

11.13 Compton cross sections (comp_xsections)

If the `Bound Compton scattering= Simple` option is selected (see above), then the user also has the option of specifying their own Compton cross section data using the `Compton cross sections` input. Cross section data must exist in the `$HEN_HOUSE/data` directory and the file name must have the form `x_compton.data`, where `x` is a name specified by the user. All values of `x` will appear in the GUI menu where Compton cross section data can be selected. Alternatively, if editing the `.egsinp` file directly, the form of this input is:

```
Compton cross sections= x
```

Default Compton cross section data is contained in the file `compton_sigma.data` included with the EGSnrc system. Note that this default data is also used if `Bound Compton scattering` is not set to `Simple`.

11.14 Radiative Compton corrections (radc_flag)

If set to `On`, then radiative corrections for Compton scattering based on the equations of Brown and Feynman (Phys. Rev. 85, p 231–1952) are used. If set to `Off` (the default) no corrections are done. Note that if set to `On` then the variable `SOURCES` in the `sources.make` file for the accelerator (See Section 2.12.4 above) must be modified to include `$(EGS_SOURCEDIR)rad_compton1.mortran` just before `$(EGS_SOURCEDIR)get_inputs.mortran`.

11.15 Pair angular sampling (IPRDST)

This input determines the method used to sample the positron/electron emission angles (relative to the incoming photon) in a pair production event. There are three possible settings: `Off`, `Simple` (the default) and `KM`. If it is set to `Off`, then the positron and electron created by pair production have fixed polar angles, $\theta_{\pm}$, given by $\theta_{\pm} = \frac{m}{E_\gamma}$, where m is the electron rest energy and E_γ is the energy of the original photon. If `Pair angular sampling= KM`, then equation 3D-2003 in an article by Motz et al[45] is used to determine the positron/electron emission angles. This option is similar to the sampling technique used by EGS4/BEAM. Finally if `Pair angular sampling= Simple` (the default), then only the first term in the the Motz et al equation 3D-2003 is used. The `KM` option becomes less efficient with increasing accelerator energies and, moreover, involves assumptions that are questionable at low energy. For these reasons, the default setting is `Simple`.

11.16 Pair cross sections (pair_nrc)

The `Pair cross sections` input determines the cross-sections to use for pair production events. If set to `BH` (the default), then Bethe-Heitler cross sections are used. If set to `NRC`, then the NRC cross sections found in `$HEN_HOUSE/data/pair_nrc1.data` are used. The `NRC` setting is only of interest at low energies, where these cross-sections take into account assymmetry in the positron-electron energy distribution.

11.17 Photoelectron angular sampling (IPHTER)

The `Photoelectron angular sampling` input determines the sampling method used by EGSnrc to determine the angle of emission of photoelectrons. If `Photoelectron angular sampling= Off`, then photoelectrons inherit the direction of the incident photon. If `Photoelectron angular sampling= On` (the default), then Sauter's formula [46] is used to determine the angle of the photoelectron. Note that, in most applications, we have not observed any difference between the `Off` and `On` settings of this parameters. Also note that, strictly speaking, Sauter's formula is only valid for K-shell photo-absorption and is also derived from extreme relativistic approximations. Thus, if the user has a better approach, they can insert it in the `$SELECT-PHOTOELECTRON-DIRECTION;` macro in `$HEN_HOUSE/egsnrc.macros`.

Similar to bound Compton scattering, photoelectron angular sampling can be turned on

or off in selected regions (with the opposite setting everywhere else) by setting `Photoelectron angular sampling= On in regions` or `Photoelectron angular sampling= Off in regions` together with the inputs `PE sampling start region` and `PE sampling stop region` to define the region ranges for which photoelectron angular sampling is to be turned `On` or `Off`.

11.18 Rayleigh scattering (IRAYLR)

This input determines the simulation of Rayleigh (coherent) scattering. Its possible settings are `On` (the default), `Off` and `Custom`. Note that this replaces the `IDORAY` input in the BEAM main inputs (see section 10.12). If `Rayleigh scattering= On` (the default), then Rayleigh events are simulated using the total coherent cross-sections from Storm and Israel[47] and atomic form factors from Hubbell and Øverbø[48].

If `Rayleigh scattering= Off`, then Rayleigh events are not simulated. The `Custom` setting allows the user to specify custom Rayleigh form factors for specified media. To do this, the user must specify the list of media in additional input `ff media names=` and the list of files containing custom form factors for each medium in the additional input `ff file names=` . For media not included in `ff media names`, Rayleigh scattering is turned `On` and uses default atomic form factors.

Rayleigh scattering is recommended for low energy (< 1 MeV) simulations. Also, for proper simulation of Rayleigh events, bound Compton scattering (see section 11.12 above) should also be simulated.

Rayleigh scattering can be turned `Off` for MV applications with little to no effect on simulation accuracy.

Note that if Rayleigh scattering is turned `On` and PEGS4 photon cross sections are used (see section 11.19 below) then the option to include Rayleigh data must have been turned on when generating the PEGS4 data.

Rayleigh scattering can be turned `On` or `Off` in selected regions (with the opposite setting everywhere else) using `Rayleigh scattering= On in regions` or `Rayleigh scattering= Off in regions` and the inputs `Relaxations start region` and `Relaxations stop region` to define the region ranges for turning Rayleigh scattering on or off.

11.19 Photon cross sections (photon_xsections)

This input determines the photon cross section data used. Current possible settings (and the data source used) are: `xcom` (XCOM), `si` (Storm-Israel), `epdl` (Evaluated Photon Data Library), `pegs4` (PEGS4), `mcdf-xcom`, and `mcdf-epdl`. The current default is `xcom`. However, this can be changed using the `$XDATA_DEFAULT` macro in `$HEN_HOUSE/src/egsnrc.macros`. Note that if `PEGS4` cross sections are either supplied by a PEGS4 data file or are calculated on the fly in pegsless mode.

The inputs, `mcdf-xcom` and `mcdf-epdl`, instruct the code to use Sabbatucci & Salvat's renormalized cross sections when simulating photoelectric events, with either the XCOM (`mcdf-xcom`) or Evaluated Photon Data Library (`mcdf-epdl`) used for all other cross sections.

The more accurate modeling of photoelectric events allowed by the the Sabbatucci & Salvat cross sections does incur a CPU time penalty (up to 6% for a 30 kV beam) and so this option is only of interest for low energy simulations. Use of these renormalized cross sections is necessary for agreement with PENELOPE results in ICRU90[49].

If the user has their own photon cross section data, `x`, then this can be used as an input for `Photon cross sections` provided that the files: `x_photo.data` (photoelectric cross sections), `x_pair.data` (pair production event cross sections), `x_triplet.data` (triplet event cross sections) and `x_rayleigh.data` (rayleigh cross sections) exist in the `$HEN_HOUSE/data` directory. Once these files are in place, then "x" will appear in the pull-down menu in the GUI where photon cross-sections are specified. Alternatively, if you are editing the `.egsinp` file directly, you can enter the line:

```
Photon cross sections= x
```

inside the block of EGSnrc transport parameter inputs.

11.20 Photon cross-sections output (xsec_out)

The input `Photon cross-sections output` can be set to `Off` (the default) or `On`. If set to `On`, then the simulation outputs the file `$EGS_HOME/BEAM_accelname/inputfile.xsections` which contains the photon cross section data used in the simulation.

11.21 Atomic Relaxations (IEDGFL)

This input determines how the relaxation of atoms to their ground state after Compton, photoelectric and electron impact ionization events is simulated. Possible settings are `Off`, `On`, `eadl` (the default) and `simple`. The `On` option defaults to `eadl`.

If set to `Off` then relaxations are not simulated. In this case, when there is a photoelectric event, EGSnrc transfers all of the photon energy to the photoelectron. This is different from EGS4/BEAM, where the binding energy of the electron is subtracted and deposited on the spot. Both approaches are approximations, but the EGSnrc approach is more accurate.

If set to `simple` then a simplified relaxation scheme based on the transition probabilities in the Evaluated Atomic Data Library (EADL) is used. Relaxation after Compton, electron impact ionization and photoelectric events is simulated via the emission of K-, L-, M- and N-shell fluorescent photons, Auger electrons and Coster-Kronig electrons. In this option, the EADL is also used as a "one size fits all" library of binding energies, independent of the photon cross sections used (see section 11.19 above). This can result in a mismatch between the energies of the fluorescent photons emitted and the binding energies used for interactions. Note that this was the only relaxation scheme implemented until the 2018 release of EGSnrc/BEAMnrc.

If set to `eadl` then a more detailed relaxation scheme, also based on the EADL transition probabilities, is used. This differs from the `simple` relaxation scheme in that: 1) initial $\langle M \rangle$ shell vacancies are not considered; 2) for final vacancies, exact M- and N-shells replace $\langle M \rangle$ and $\langle N \rangle$. Moreover, the `eadl` relaxation algorithm uses binding energies consistent with the

photon cross sections selected (see section 11.19 above), eliminating the mismatch between the energies of fluorescent photons and these binding energies. If the user has opted to use Sabbatucci & Salvat's renormalized photoelectric cross sections (`Photon cross sections = mcdf-xcom` or `mcdf-epdl`–see section 11.19 above)–then `simple` relaxation cannot be used, and the code will automatically default to `eadl` relaxation.

The lower energy limit for relaxation processes is 1 keV. Thus, only relaxation in shells with binding energy > 1 keV is simulated.

Simulation of atomic relaxations is essential for accurate transport in low energy (kV) applications. However, it can be turned `Off` for MV applications to reduce simulation time at no significant reduction in accuracy.

Note that the `Atomic Relaxations` option supersedes the `IFLUOR` option in EGS4/BEAM (see section 10.12), which only simulates emission of K-shell fluorescent photons after photoelectric events.

Atomic relaxations can be turned `On`/`Off` in selected regions (with the opposite setting everywhere else) using `Atomic Relaxations= On in regions` or `Atomic Relaxations= Off in regions` and the inputs `Relaxations start region` and `Relaxations stop region` to define the region ranges for which relaxations are to be turned `On`/`Off`. Since the `On` setting defaults to `eadl`, the detailed `eadl` relaxation algorithm will be used in regions where atomic relaxations are set `On`.

11.22 Electron impact ionization (eii_flag)

This input determines the cross sections used to simulate electron impact ionization (EII). The possible values of `Electron impact ionization` are: `Off` (the default), `On`, `ik`, `casnati`, `gryzinski`, `kolbenstvedt`, and `penelope`.

If set to `Off` EII is not simulated. If set to `On`, then it defaults to `ik`, and cross sections from Kawrakow's electron impact ionization theory[50] are used. The selections, `casnati`, `gryzinski`, and `kolbenstvedt`, use EII cross sections derived from the theories associated with the respective names, and `penelope` uses the cross sections identical to those used to simulate EII in the Penelope code. See the EGSnrc Manual[1] for more details.

EII cross section data is stored in the files `$HEN_HOUSE/data/eii_x.data`, where `x` specifies the theory/source of the data and is identical to the `Electron impact ionization` input (Note: inputs are case sensitive). Thus, it is possible for the user to use their own EII cross section data provided it is in the supplied in a file having the same format as those already included with the distribution.

EII is only of interest for low energy X-ray applications.

11.23 Triplet production (itriplet)

This input turns on/off the simulation of triplet production. Possible values of `Triplet production` are `Off` (the default) and `On`. If set to `On` then Borsellino's first Born approximation[45]

is used to sample triplet events based on the triplet cross section data in `$HEN_HOUSE/data/triplet.data`.

11.24 Photonuclear attenuation (iphotonucr) and Photonuclear cross sections (photonuc_xsections)

First implemented in EGSnrc 2012[51], inclusion of the photonuclear effect in simulations been found to have a significant effect on transmitted photon spectra in the MV energy range. Though it is not necessary for most applications, it should be included in the high-accuracy simulations used in standards work.

The input, `Photonuclear attenuation`, determines whether the photonuclear effect is included in the simulation. Possible settings are `Off` (the default) and `On`. If set to `On` then, if present, the input `Photonuclear cross sections` specifies the cross sections used.

Cross sections must exist in the file `$HEN_HOUSE/data/x_photonuc.data`, where `x` is the name of the cross section database and corresponds to the input for `Photonuclear cross sections`. The distribution includes the default photonuclear cross sections, `iaea_photonuc.data`, which are used in the absence of any input for `Photonuclear cross sections`.

12 Custom user inputs

If custom input parameters are required (for any modifications that you have made to `beamnrc.mortran`), then it is recommended that you include them in the `.egsinp` file between the delimiters: `:start user inputs:` and `:stop user inputs:`. This section, if required, can appear either immediately before or immediately after the EGSnrc transport parameters (see Section 11 above). You will not have access to custom inputs through the GUI, but by putting them between the delimiters you ensure that they will not be "wiped out" if you modify/save the input file using the GUI at some point.

The format of custom inputs will be similar to that for the EGSnrc transport parameters:
`PARAMETER NAME= parameter value`
Where `PARAMETER NAME` can either be the actual name of the input variable or a descriptive text string. Note the space between the parameter value and the "=" sign.

The current release of BEAMnrc includes a a custom input for redefining the output directory for phase space files, `PHSP_OUTDIR` (see Section phspoutdirsect for more information about this). In general, however, it is up to the user to modify `beamnrc.mortran` to read in their custom input parameters.

13 Parallel Processing with BEAMnrc

A BEAMnrc simulation can be split into smaller jobs which can then be run on different processors in parallel to reduce the elapsed time required for a simulation. In general, parallel

processing requires that you be running on a system that supports a job queuing utility, such as `at`, or, if sending jobs to multiple cores over a network, PBS or NQS.

In previous versions of BEAMnrc, the Unix script `pprocess` was used to automatically create the individual input files for parallel jobs and submit them (also automatically setting the random number seeds to a different values in each input file and the phase space source inputs, `IPARALLEL` and `PARNUM`). This method of parallel processing had a major limitation, though, in that each job consisted of the same number of histories, making the total simulation time dependent on the slowest computing core.

In the current version of BEAMnrc, parallel processing is accomplished more efficiently using a built-in parallel processing functionality.

To submit a parallel job, use the batch submission command, `exb` (discussed in detail in section 2.7.1). The command syntax for a parallel job is:

```
exb BEAM_myaccel inputfile pegsdata [short|medium|long] [batch=batch_system] ...
      ... [start=first_job] p=N (or stop=last_job) [fresh=yes/no]
```

`first_job` is the number of the first parallel job minus 1 (`first_job` defaults to 0), `N` is the number of parallel jobs, and `last_job` is the number of the last job (only used if `N` is not input). Note that the numbering convention of `first_job` is historical and may be changed in future releases. The input `fresh` can be used to specify whether this is an independent batch of parallel jobs (`yes`–the default) or whether they are to be added to parallel jobs currently running (`no`). The other batch inputs above are covered in more detail in section 2.7.1.

Once this command is entered, the script `$HEN_HOUSE/scripts/run_user_code_batch` (to which `exb` is aliased) enters a loop which submits BEAM_myaccel to the batch queue `N` times. Each submission has the form:

```
BEAM_myaccel -i inputfile -p pegsdata -P n_parallel -j i_parallel [-f first_job+1]
```

where `n_parallel`=`N`, `i_parallel` takes on values 1,2,..., `n_parallel`, depending on which job is being submitted, and `first_job`+1 is the number of the first parallel job. The last argument is omitted if the batch submission script is run with `fresh=no`. For each parallel job, BEAMnrc will create a temporary working directory (see section 2.9 for more on temporary working directories), and the output files from the `i_parallel`th run will have the naming scheme `inputfile_w[i_parallel].egslog`, `inputfile_w[i_parallel].egslst`, `inputfile_w[i_parallel].egsphsp1`, `inputfile_w[i_parallel].egsdat`, etc. Note that `N` different input files are not created, and that all parallel runs make use of the original input file.

The first job submitted (`first_job`+1) creates a file, `inputfile.lock`, in the `BEAM_myaccel` directory. The `.lock` file, or job control file, is accessed and updated by all parallel jobs and contains:

- `n_left`: The number of histories remaining to be run after the current job is submitted.
- `n_tot`: The total number of histories already completed
- `n_job`: The number of jobs currently running

- `sum` and `sum2`: Scored quantity of interest and (quantity of interest)^2 from all previous runs (*i.e.*, runs completed before submitting the current parallel run)

- `res` and `dres`: scored quantity of interest and relative uncertainty on quantity of interest (%) after the completion of `n_tot` histories in the current parallel run plus all previous runs.

Note that the `.lock` file created by BEAMnrc during parallel runs currently does not keep track of any quantity of interest, and `res`=0.0 and `dres`=99.9%.

Before running any history, a parallel job opens the `.lock` file and reads `n_left` (the file cannot be read by more than one job at a time). If `n_left`>0, then there are still histories to run, and the job begins execution. Rather than each job running `NCASE/n_parallel` histories (as in the old parallel processing scheme), each job runs only a fraction, or chunk, of this number at a time. Thus, the maximum number of histories that a job runs at a time is `NCASE/(n_parallel*$N_CHUNK)`, where `$N_CHUNK` is defined in `$HEN_HOUSE/src/egsnrc.macros` and is set to a default value of 10. If `n_left` happens to be less than `NCASE/(n_parallel*$N_CHUNK)`, then the job will run `n_left` histories. Before beginning the run, the job updates the `.lock` file, decrementing `n_left` by the number of histories it is about to run, and incrementing `n_job` if this is the first run for this job. After the job has finished its chunk of histories, it will loop back to the beginning of the simulation, read the contents of the `.lock` file again, and determine whether another chunk of histories needs to be run.

Doing parallel simulations in chunks like this prevents jobs that are running on slower cores from tying up large portions of the simulation and, hence, determining the total elapsed time required.

If, on opening the `.lock` file, a parallel job finds that `n_left`=0 (ie no more histories to run), then it immediately exits the simulation loop, analyzes the data from its runs for output to the `inputfile_w[i_parallel].egslst` file, moves all output files out of its temporary working directory, deletes this directory, and decrements `n_job`. If, after decrementing `n_job`, `n_job`=0, then this is the last job to stop running, and it automatically calls the subroutine `egs_combine_runs` to combine the results from all parallel jobs (see section 13.3 below).

13.1 Random Number Seeds with Parallel Jobs

It is important that each parallel job start with a different state of the random number generator, otherwise you will end up combining identical results at the end of a parallel run, compromising both the results and the uncertainty estimate on them. BEAMnrc avoids this problem by automatically incrementing the second random number seed, `JXXIN` (see section 10.8) for each parallel job. So for job number `i_parallel`:

$$JXXIN = JXXIN_{input} - 1 + i\_parallel \tag{5}$$

where $JXXIN_{input}$ is the value of `JXXIN` in `inputfile.egsinp`.

13.2 Phase Space Sources with Parallel Jobs

When running parallel jobs that use a single phase space source (see section 4.13), it is important that the entire source be evenly sampled over all jobs. In order to facilitate this, BEAMnrc divides the source into `n_parallel*$N_CHUNKS` equal partitions. Each chunk of the run then uses a different partition of the source.

The number of particles in each source partition, `p_per_phsp_chunk`, is given by:

$$\texttt{p\_per\_phsp\_chunk} = \frac{\texttt{NNPHSP}}{(\texttt{n\_parallel} * \texttt{\$N\_CHUNKS})} \tag{6}$$

where `NNPHSP` is the total number of particles in the phase space source. Before a parallel job starts a chunk of histories, it calculates which chunk of the total histories it is about to run, `n_run_chunk`, using the equation:

$$\texttt{n\_run\_chunk} = \frac{(\texttt{NCASE} - \texttt{n\_left})}{\texttt{NCASE}} * (\texttt{n\_parallel} * \texttt{\$N\_CHUNKS}) \tag{7}$$

Recall that `n_left` is the total number of histories remaining AFTER this run begins. The partition of the phase space source used by this chunk of histories is then given by:

$$(\texttt{n\_run\_chunk} - 1) * \texttt{p\_per\_phsp\_chunk} + 1 \leq \texttt{INPHSP} \leq \texttt{n\_run\_chunk} * \texttt{p\_per\_phsp\_chunk} \tag{8}$$

where `INPHSP` is the particle number used.

Note that particles may be recycled within a partition, depending on the setting of `NRCYCL` (see section 4.13). Also, if all the particles in a partition are used up during a run, the partition is restarted at its first particle (ie `INPHSP = (n_run_chunk-1)*p_per_phsp_chunk+1)`.

13.3 Combining Results from Parallel Jobs

The last parallel job to finish running automatically combines the results of all parallel runs by calling the EGSnrc subroutine `egs_combine_runs`. This subroutine loops through `i_parallel`=1,...,`n_parallel` and, for each value of `i_parallel`, calls the BEAMnrc subroutine `combine_results`. `combine_results` opens the file `inputfile_w[i_parallel].egsdat`, reads the values of $\sum_{i=1}^{nhist}(\text{energy deposited})_i$, $\sum_{i=1}^{nhist}(\text{energy deposited})_i^2$, $\sum_{i=1}^{nhist}(\text{fluence})_i$ and $\sum_{i=1}^{nhist}(\text{fluence})_i^2$ (where `nhist` is the number of primary histories) in each dose or fluence scoring zone, and adds these values to their respective totals over all parallel jobs. Once the results from all parallel jobs have been summed, BEAMnrc then calculates the doses, fluence values and their uncertainties [17] for the combined run in the same way it would calculate the results of a single run.

The results of the analysis will be output to `inputfile.egslst`. It indicates which `inputfile_w[i_parallel].egsdat` have been added together followed by full output of the dose and fluence results.

Note that the `.egsdat` files are essential for combining parallel runs. Thus, if you are running in parallel, you must have the input variable `IDAT`=0 so that the `.egsdat` files are output (see section 10.5, page 120 for more information about `IDAT`).

Parallel runs can be combined manually by re-running your accelerator with the input variable IRESTART=4 (in the GUI, this is equivalent to setting the Run option to "analyze parallel jobs") after all parallel jobs have finished. Use of IRESTART=4 is generally not necessary since the last job automatically combines parallel results, however, it may be useful if, for some reason, all of the .egsdat files were not moved out of their temporary working directories or if you wish to add more .egsdat files from a separate group of parallel runs. For more information on IRESTART options, see section 10.3.

The process of combining parallel runs described above does not combine phase space files generated by the parallel jobs. These must be combined separately using the MORTRAN addphsp utility described in the next section.

13.4 Combining Phase Space Files from Parallel Runs using addphsp

Phase space files from parallel runs (with a naming scheme: inputfile_w[i_parallel].egsphsp[scoringplanenumber]) are not automatically combined at the end of the run. However, the MORTRAN utility, addphsp, can be used to combine phase space files after all parallel jobs are finished.

To use addphsp, simply go to the accelerator directory (ie where the phase space files are located) and type:

```
addphsp inputfile outputfile ipar [istart] [iscore] [i_iaea]
```

where inputfile is the name of the original input file (no .egsinp extension), outputfile is the name of the output file for the concatenated phase space data (a .egsphsp[iscore] extension is added automatically), ipar is the number of parallel jobs being added, istart is the job number at which adding begins (ie the first value of i_parallel–defaults to 1), iscore is the scoring plane number for which you are combining phase space files (*i.e.* are you adding .egsphsp1, .egsphsp2 or .egsphsp3 files?–defaults to 1), and i_iaea is set to 1 if you are adding IAEA-format phase space files (default is 0 for EGSnrc-format phase space files).

addphsp then concatenates the phase space files inputfile_w[istart].egsphsp[iscore], inputfile_w[istart+1].egsphsp[iscore], ...,inputfile_w[istart+ipar-1].egsphsp[iscore]. If you are missing some of the phase space files in the series addphsp will automatically skip over those and just concatenate those that are available.

In the case of IAEA-format phase space files, the naming scheme is different, and addphsp concatenates the files inputfile_w[istart].iscore.IAEAphsp, inputfile_w[istart+1].iscore.IAEAphsp, ...,inputfile_w[istart+ipar-1].iscore.IAEAphsp. The corresponding .IAEAheader files (em i.e. inputfile_w[istart].iscore.IAEAheader, etc) are assumed to be in the same directory. Output will be to the files outputfile.iscore.IAEAphsp and outputfile.iscore.IAEAheader. IAEA functionality in addphsp requires that EGSnrc/BEAMnrc be installed on a machine with a working C++ compiler. See Section 7.4 for more information on the IAEA phase space format.

Note that addphsp requires twice the disk space of the total size of all phase space files being combined, since files are not automatically deleted after they are added.

addphsp makes use of the record length factor ($RECL-FACTOR) for 4-byte phase space data found in $HEN_HOUSE/lib/config/machine.macros, some of the phase space reading/writing macros contained in $HEN_HOUSE/utils/phsp_macros.mortran (see section 7, page 105), some of the IAEA-format reading/writing macros in $HEN_HOUSE/utils/iaea_phsp_macros.mortran (only if installed with a working C++ compiler), and some of the macros found in $HEN_HOUSE/src/egsnrc.macros.

Normally, addphsp is compiled as part of the BEAMnrc installation. However, you can compile it separately by going into $OMEGA_HOME/progs/addphsp and typing make. The executable, addphsp*, will be put into directory $HEN_HOUSE/bin/config.

13.5 Parallel Jobs Run from the GUI

You can start parallel jobs from the BEAMnrc GUI. Select "Run" from the "Execute" menu and the running window that opens up will allow you to select "run in parallel", together with the number of parallel jobs. Note that on selecting "run in parallel", the GUI automatically switches to "batch" mode if this has not already been selected. You can then select the queue ("short", "medium" or "long") that you would like to submit the parallel jobs to. The queuing system used defaults to PBS, unless specified otherwise by the $EGS_BATCH_SYSTEM environmental variable.

13.6 Restarting Parallel Jobs

Parallel runs can be restarted (by setting IRESTART=1 in inputfile.egsinp) provided that you have the .egsdat files from all of the previous individual parallel jobs available in your $EGS_HOME/BEAM_myaccel directory. If you are using a phase space source, however, restarting presents a problem in that a particular partition of the source may get used by a different job the second time around. At the end of the run, results from the second use of this partition will be recombined with those from its first use with no attention paid to the correlation between the two results. This will result in the uncertainties being underestimated. We recommend that you do not restart a parallel run if you are using a phase space source.

14 Statistics in BEAMnrc

Starting with the February 2002 release of BEAMnrc, a history-by-history method of estimating uncertainties was implemented in BEAMnrc[17]. This method offers considerable improvements over the method of statistical batches used up until then.

The history-by-history method involves grouping scored quantities (eg fluence, energy deposited) according to primary history during a run and then determining the root mean square standard deviation on the mean of the groupings. For most sources, there is no difference between a primary history and an incident particle. However, for phase space sources, where more than one incident particle can be traced back to a single primary history,

it is important to group according to primary history in order to account for correlations between the incident particles.

For more information, see the report on history by history statistics in BEAMnrc and DOSXYZnrc [17].

15 Component Modules

15.1 Introduction

One of the design features of BEAMnrc is that each part of the accelerator or source unit is considered to be a single component module which takes up an horizontal slab portion of the accelerator. These component modules are re-usable and are all completely independent. They communicate with the rest of the system in certain well specified ways. The purpose of this section is to list all component modules currently used at NRC, describe what each component module does and how it is used in BEAMnrc.

A component module can be considered as a block which has a 'front' surface and a 'back' surface. An accelerator is built with many such blocks. Very often there is gap between two blocks. This gap is automatically filled with AIR by the BEAMnrc main routine, which is consistent with the case of a real accelerator. The air gap which is in front of a module, and after the 'back' plane of the previous module, is considered as a part of this component module. However, we still define the 'front' surface of a module as the front plane of NON-AIR medium in the module.

15.2 What Each Component Module Does

SLABS
SLABS is used for multiple slabs of arbitrary thickness and material which are perpendicular to the z axis. The outer boundary is a square.

CONS3R
CONS3R is designed to simulate cylindrical structures that can be described using a series of (Z,R) points rotated about the Z axis. Examples of such structures are rings, cylindrical slabs and cones (primary collimators). This CM has only 3 regions: the interior of the cylindrical structure, the outside of the structure, and (possibly) the air gap at the front). The current version can only allow convex shapes in the Z direction [i.e., Z(i+1) must be greater or equal to Z(i), see section 15.3.3 for more details].

CONESTAK
CONESTAK is used to simulate a stack of truncated cones. A primary collimator is a special case for CONESTAK using only one cone. Each layer is of user-defined thickness, and, within each layer, the user defines the radius of the cone at the top of the layer and at the bottom of the layer, as well as the medium inside the cone and outside the cone. In addition, the entire, multi-layered, conical structure may be surrounded by a cylindrical wall of user-defined medium. CONESTAK is rotationally symmetric about the beam axis.

FLATFILT
This is an even more general purpose CM to simulate a set of truncated stacked cones. It is necessary for some very complex flattening filter designs. Similar to CONS3R and CONESTAK, FLATFILT is rotationally symmetric about the beam axis.

CHAMBER
CHAMBER is used for parallel-plate ion chamber in the container with top and bottom planes of arbitrary thickness and material. CHAMBER can also be used to score the central axis dose in a water phantom outside an accelerator (see section 15.3.6 for more details). This is a cylindrical CM, rotationally symmetric about the beam axis.

JAWS
JAWS is used for sets of paired bars or jaws, which can be in the collimator or applicator. The user defines the angle of the inner faces of the jaws with respect to the Z (beam) axis. The jaws can open in either the X or Y direction. The bars are of arbitrary thickness and material. The outer boundary of JAWS is a square centered on the beam axis.

DYNJAWS
DYNJAWS is a version of JAWS in which the jaw settings (Z positions and opening coordinates) can be specified as changing over the course of a BEAMnrc simulation. The user can run DYNJAWS in "step-and-shoot" mode, which simulates jaw settings changing while the beam is off, and "dynamic" mode, which simulates jaw settings changing while the beam is on. When used in these modes, the jaw settings must be specified in a separate file, where each complete group of settings (ie Z positions and opening coordinates for all jaws) is defined as a "field", and the probability, or index, of each field is specified by the user. DYNJAWS is particularly useful for simulating dynamic wedges.

SYNCJAWS
SYNCJAWS is a version of DYNJAWS in which the field settings (opening dimensions and Z-positions of the jaws) can synchronized with other synchronized CMs in the accelerator (SYNCVMLC, SYNCMLC, SYNCHDMLC) and with the motion of sources 20 (phase space) and 21 (BEAM treatment head) in DOSXYZnrc[15].

APPLICAT
APPLICAT is used for a set of rectangular scrapers. Each scraper is defined by the outer region of two concentric rectangles, the inner region being air. The scrapers are of arbitrary thickness, width and position relative to the reference plane ($Z = 0$). The scrapers can be of different materials. The outer boundary of APPLICAT has square symmetry centered on the beam axis. This is an extension of the CM APPSQ described in the BEAMnrc paper[3].

CIRCAPP
CIRCAPP is similar to APPLICAT, only the openings in the scrapers are circular. Each scraper is defined by its rectangular outer boundary and circular inner boundary. The outer boundary of CIRCAPP has square symmetry centered on the beam axis.

PYRAMIDS
PYRAMIDS is used to model a stacked set of truncated pyramids, often a rectangular collimator or applicator. For each layer, the user defines the medium of the pyramid and also of the layer out of which the pyramid is "carved". Layers must have a minimum air

gap in between them but need not extend all the way to the outer boundary. Similar to APPLICAT and JAWS, the outer boundary of PYRAMIDS is a square centered on the beam axis.

BLOCK
The component module BLOCK is used to model beam treatment blocks having non-rectangular and/or multiple openings. BLOCK consists of a single layer of block material. Openings in the material are comprised of subregions specified by the user. The inner sides of the opening(s) all angle towards a single, user-specified "focus point" on the beam axis. The outer boundary of BLOCK is a square centered on the beam axis.

MLC
MLC is used to model a double focused multi-leaf collimator with a flat face. The collimator has a single layer composed of a user-specified number of leaves all opening in either the X or Y direction. The collimator opening is composed of the openings of individual leaves as specified by the user. MLC has two materials: the material in the collimator opening (usually air), and the material of the leaves and collimator body. The outer boundary of this CM is a square centered on the beam axis.

MLCQ
MLCQ is similar to MLC, however it has rounded leaf ends. The user specifies the radius of the leaf ends. In addition, the leaf ends can be angled down or up by specifying the Z position where the radius of the leaf ends originates.

VARMLC
VARMLC is similar to MLC and MLCQ (it can have either rounded or angled leaf ends). The main difference between it and these earlier component modules is that VARMLC also simulates the air gaps between leaves perpendicular to the leaf direction, the tongue-in-groove which connects adjacent leaves, and the screws at the top and bottom which are used to open and close leaves.

MLCE
MLCE was designed as a variation of VARMLC specifically for modeling Elekta MLC's. The tongue-and-groove in VARMLC has been replaced by interlocking steps. Also, unlike VARMLC, all leaves are identical and the sides of the leaves are focused (always to Z=0) by tilting each leaf about an axis that runs down the centre of its top surface. The entire leaf bank can also be tilted for off-axis focusing.

DYNVMLC
DYNVMLC, also based on VARMLC, was specifically designed to model the Varian Millenium multi-leaf collimator. The user specifies the cross-sections of the 3 different leaf types (FULL, TARGET and ISOCENTER leaves) in the collimator. These leaf cross-sections are more complex than those in VARMLC. Then, the user assigns a leaf type to each leaf in the leaf bank as well as the opening dimensions of the leaves. Instead of tongue-in-groove, the leaves fit together with interlocking steps. DYNVMLC allows the user to simulate different leaf opening coordinates during a single simulation, provided that the user supplies a file containing the leaf opening data. Leaf openings can be simulated to change while the beam is on (dynamic) or while the beam is off (step-and-shoot).

SYNCVMLC

SYNCVMLC is a version of DYNVMLC which allows fields (defined by leaf opening coordinates) to be synchronized with any other dynamic synchronized CM in the accelerator (SYNCJAWS, SYNCMLCE, SYNCHDMLC) and with the simulated motion of of either source 20 (phase space source) or source 21 (BEAM treatment head source) in DOSXYZnrc[15].

SYNCMLCE
SYNCMLCE is a dynamic, synchronized version of MLCE. Leaf opening coordinates for the ELEKTA MLC can be specified in a file and each associated with a fractional monitor unit index. Leaf motion during the simulation can be simulated using either "dynamic" (continuous leaf motion) or "step-and-shoot" (leaf motion only when the beam is off) mode. SYNCMLCE can be synchronized with any other synchronized CMs in the accelerator (SYNCJAWS, SYNCVMLC, SYNCHDMLC) and with the orientation of sources 20 (phase space) and 21 (BEAM treatment head) in DOSXYZnrc[15].

SYNCHDMLC
SYNCHDMLC is a version of SYNCVMLC optimized for modeling the High-definition Multileaf Collimator (HDMLC 120) available on TrueBeam and Novalis linacs. SYNCHDMLC features five possible leaf cross sections: FULL, HALF TARGET/HALF ISOCENTER leaves, and QUARTER TARGET/QUARTER ISOCENTER leaves. FULL leaves are similar to FULL leaves in DYNVMLC/SYNCVMLC. HALF TARGET/HALF ISOCENTER leaves are equivalent to the TARGET/ISOCENTER leaves in DYNVMLC/SYNCVMLC. QUARTER TARGET/QUARTER ISOCENTER have the same shaped cross sections as the HALF TARGET/HALF ISOCENTER leaves, but the cross section perpendicular to the opening direction is generally thinner. Similar to other synchronized CMs, leaf opening coordinates can be synchronized with opening dimensions of other synchronized CMs in the accelerator and with the motion of sources 20 (phase space) and 21 (treatment head) in DOSXYZnrc[15].

MESH
The MESH component module is used to model a single-layer wire mesh placed perpendicular to the beam direction in the path of the beam. The user specifies the X and Y dimensions of the rectangular "holes" in the mesh (all holes have the same dimensions) and the width of the wire between the holes. There is also the possibility of a region of air between the edge of the mesh and `RMAX_CM`. Air holes are placed in a regular pattern in the mesh, and the mesh has rectangular symmetry about the beam axis. The outer boundary of MESH is a square centered on the beam axis.

MIRROR
MIRROR is used for a mirror in the accelerator. It can have arbitrary angle with respect to the Z axis. The number of layers and their thicknesses, materials in the mirror can be arbitrary. The mirror is surrounded by air. The MIRROR outer boundary is a square centered on the beam axis.

XTUBE
XTUBE is used to simulate an x-ray source. It must be the first CM in a simulation. The target may consist of several flat layers of different materials backed by a target holder (backing material). The target angle is defined by the target surface and the z-axis. The incident circular or rectangular beam is from the side normal to the z-axis. XTUBE should always be used as the first CM and the second CM may be any of the available `CMs` with a central opening serving as the exit window of the x-ray tube. The square outer boundary of

XTUBE is centered on the z-axis.

SIDETUBE
SIDETUBE models concentric cylinders parallel to the X-axis. The user specifies the X positions of the cylinder ends, the number of concentric cylinders, the radii and media of the cylinders, and the medium surrounding the cylinders. This is one of the 3 CM's in which an isotropic source (`ISOURC`=3) can be placed (CONESTAK and FLATFILT are the other ones) and is excellent for modelling isotropic radiating cylinders that are perpendicular to the beam (collimated) axis. Note that all cylinders are necessarily the same length (*i.e.* have the same X limits). The outer boundary of this CM is a square centred on the Z-axis.

ARCCHM
ARCCHM is designed to model an arc-shaped array of ion chambers such as those used in the prototype tomotherapy unit at the University of Wisconsin. However, this CM can be used to model any arc-shaped structure in the path of the beam. The user specifies the Z position and radius of the front of the arc. The arc is always concave up and curves in the Y direction (*i.e.* there is no concavity in the X direction), with it's lowest point at Y=0. The angle subtended by the arc is determined by the user-specified number of chambers, widths of the chambers and of the septa separating the chambers, and the thickness of the chambers/septa. The user also specifies the extent of the arc in the X direction.

15.3 Geometry and Input Parameters of Component Modules

15.3.1 Overview

For each of the component modules, a picture is provided here (ZX or ZY projection), showing its geometry and the input parameters needed to run BEAMnrc with this module. When creating input files using the BEAMnrc GUI, you may preview each CM with the geometry parameters that you have input. You may also refer to the in-line documentation in the source code of each component module for a clear input format, particularly for those input parameters other than geometrical parameters. The in-line documentation has been included in the next pages.

Each component module has a maximum radius or half-width, `RMAX_CM` beyond which the particles will not be followed during a simulation. This parameter is read in by the MAIN routine of BEAMnrc. In the input file to run BEAMnrc, dummy lines (filled with ********** in the beginning) are used to separate the different component modules.

If there is a gap between two component modules, as in the case of a real accelerator, this gap is filled automatically with air by the BEAMnrc main routine[1]. Therefore for each component module there is a possible air gap in front of it. In general, we consider the air gap is a part of the component module. The air gap, from the back plane of previous module to the front (non-air medium) plane of this module, is shown in the picture of a component module in this documentation.

The local region numbers (`IR`) are indicated in the picture of each component module.

[1]Strictly it is with the material specified in the second record of the input file and nominally referred to as air

The code makes use of both the local region numbers and the global region numbers. The code has mappings from one to the other. For further information on these details see Appendix A.1 "Specifications for Component Modules for BEAMnrc".

The following subsections discuss the geometry, input parameters, and input format for each component module.

Except for the very first component module, usually an air gap exists in front of a module. We consider the air gap, which is in front of a module and after the previous module, is a part of this module. However we define the 'front' surface of a module is the front plane of non-air-gap medium in the module.

15.3.2 SLABS

SLABS is used for multiple planes of arbitrary thickness and material, i.e., a set of semi-infinite slabs. One single slab is a special case for SLABS. SLABS has square symmetry about the beam axis. Note that `ESAVE` can be set for each region in SLABS, unlike other CMs where `ESAVE_GLOBAL` is used. This is because SLABS is often used to model the bremsstrahlung target in photon accelerators (see section 6.1).

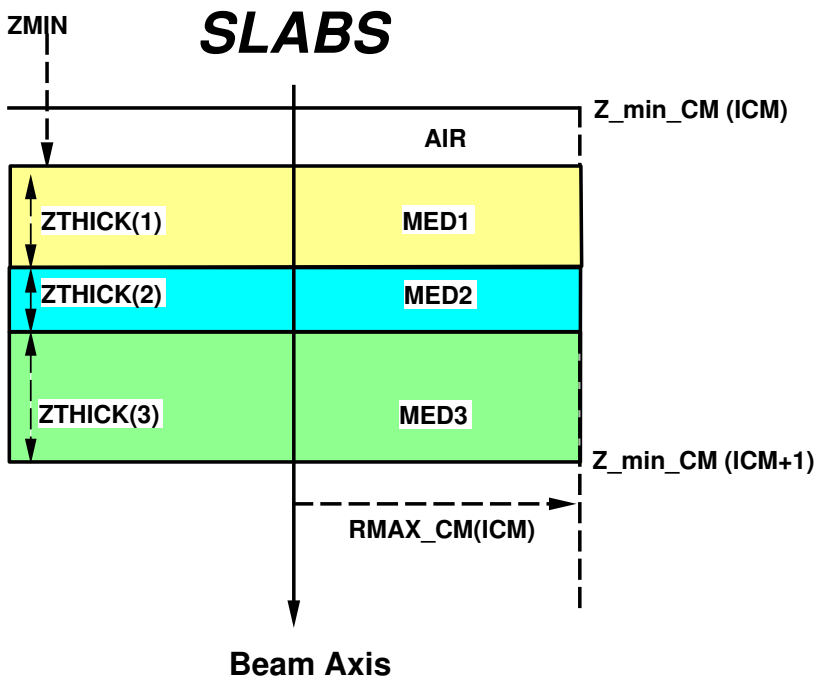


Figure 22: Schematic of CM SLABS, which in the example consists of three layers of different materials with thicknesses `ZTHICK(1)`, `ZTHICK(2)`, and `ZTHICK(3)`, respectively. The gap between the back of the previous CM and the front face of this CM is automatically filled with air. The outer boundary of this CM is a square of half-width `RMAX_CM`.

The input format for SLABS, and an example of the input file are given as follows.

```
CARDS CM_$SLABS  (SLABS: Rev 1.6)
**************

-1  dummy line (filled with ****)  read in main

 0  RMAX_CM(ICM_$SLABS)        outer boundary for CM - 1/2 side of
                               square(read in main)

 1  TITLE_$SLABS (60A1):  Title of CM.

 2  N_$SLABS (I5):  Number of planar slabs in CM = # regions in CM,
                    excludes any air gap needed.

 3  ZMIN_$SLABS (F15.0):  Distance from front of first slab to reference
```

```
                        plane  (Z=0).

 4  Parameters of each slab from front to back (increasing Z).  One pair of
    cards (4a and 4b) for each of the slabs.

    4a   ZTHICK_$SLABS, ECUT, PCUT, DOSE_ZONE, IREGION_TO_BIT, ESAVEIN
                        (3F15.0,2I5,F15.0):

            ZTHICK_$SLABS:   slab thickness.
            ECUT, PCUT:      Cutoff energies-defaults are ECUTIN,PCUTIN
            DOSE_ZONE:       Dose zone to score dose - 0 if not scored
            IREGION_TO_BIT:  map this region to this bit number in LATCH
            ESAVEIN:         Value of ESAVE for this region if range
                             rejection on. Default is ESAVE_GLOBAL.

    4b   MED_IN (24A1):  Medium of the planar slab, used to set MED_INDEX.

 Example
 *******

The following set of cards defines a 1 cm thick slab of air sandwiched
between two 0.1 cm thick slabs of tungsten.  The front slab is at
Z=7.32 cm. Electrons will be followed in the slabs down to kinetic energies
of 10 keV (total energies of 0.521 MeV) and photons will be followed down
to energies of 1 keV.  The dose deposited in the air will be scored and
added to the dose from the other regions in dose zone 1, and the
dose deposited in both tungsten slabs will be scored and added to the dose
from the other regions in dose zone 2.  Particles interacting in
the first slab will be associated with BIT 1 in LATCH.  In all slabs,
ESAVEIN=0, thus ESAVE in each slab will default to ESAVE_GLOBAL.

10.0,                          RMAX_CM
Multiple slabs: 0.1cm W-1cm air-0.1cm W, ECUT=0.521, PCUT=0.001
3,                             N_SLABS
7.32,                          ZMIN_SLABS
0.1, 0.521, 0.001, 2,1,0.0,    ZTHICK_SLABS etc
W521ICRU
1., 0.521, 0.001, 1,0,0.0
AIR521ICRU
0.1, 0.521, 0.001, 2,0,0.0
W521ICRU
```

15.3.3 CONS3R

CONS3R is a stack of truncated cones coded as three regions. It can be used for any case if there is cylindrical symmetry and if there are only two radial regions. Examples are slab, ring, stack rings, cone (primary collimator),cone stack. The current version only allows convex shapes in the Z direction, not concave shapes (*i.e.* `Z(I1)`+ is greater than `Z(i)`). CONS3R is computationally more efficient than the other conical CMs when there is more than one layer (*i.e.* more than 3 node points). This efficiency comes because there are only 2 Z boundaries for particles to cross in CONS3R, no matter how many node points there are.

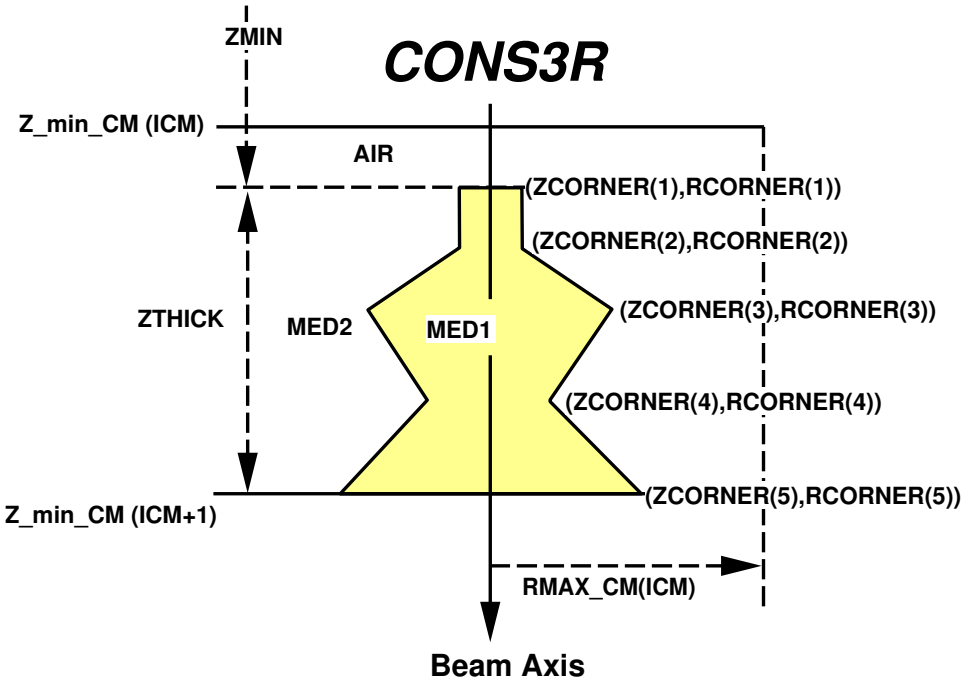


Figure 23: Component module CONS3R defined by 5 node points, where RCORNER values can be zero and `RCORNER(i+1)` can be smaller than `RCORNER(i)`, but `ZCORNER(i+1)` must be greater than or equal to `ZCORNER(i)`. Z_{MIN} is the distance from the reference plane (Z = 0) to the front plane of the CM (excluding the air gap). The beam axis is an axis of rotation and the outer boundary is a cylinder.

The input format for CONS3R, and an example of input file are given as follows.

```
CARDS CM_$CONS3R (CONS3R: Rev 1.10)
**************

 -1 Dummy line to indicate start of CM

 0  RMAX_CM(ICM_$CONS3R) (F10.0):  Outer radial boundary of CM (cm).

 1  TITLE_$CONS3R (60A1):  Title of CM.

 2  ZMIN_$CONS3R (F15.0):  Dist from front of cones to
                           reference plane (Z=0).

 3  ZTHICK_$CONS3R (F15.0): The thickness of cones (excludes front air).

 4  NUM_NODE_$CONS3R (I5):  The # of points to be used <$NPOINT_CONS3R.

 Repeat 5 for I=1,NUM_NODE_$CONS3R

 5  ZCORNER_$CONS3R(I), RCORNER_$CONS3R(I) (2F15.0):
      Positions (Z, R) for node I.  First & last must match
      ZMIN_$CONS3R and ZMIN_$CONS3R+ZTHICK_$CONS3R.  Note also the
      restriction Z(I+1) >= Z(I).

 Repeat 6, 7 for inner (ie inside cons3r), then outer region

 6  ECUT, PCUT, DOSE_ZONE, IREGION_TO_BIT, IREJCTIN  (2F15.0,3I5):

       ECUT, PCUT:  Cutoff energies for electrons and photons.
       DOSE_ZONE:   If non-zero, dose in this zone is scored in this
                    dose zone
       IREGION_TO_BIT:   region to LATCH bit correspondence for particles
                    interacting in this region
       IREJCTIN:    If IREJCT_GLOBAL is on, then by setting IREJCT=-1
                    here, range rejection is turned off in this region
                    If left blank or zero, the global value is used.

 7  MED_IN (24A1):  Medium of region
                    used to set MED_INDEX.


 Example
 *******

The following input example describes a 1cm thick flat-top cone, having
a radius at the top of 0.8cm and a radius at the bottom of 1.2cm
sitting on a 0.3cm thick cylinder of radius 1.5cm which, in turn,
is sitting atop a flat-top cone of thickness 1.0cm with top radius
```

```
0.5cm and bottom radius=0.8cm.  The two cones and cylinder are
made of H2O--note that all of these structures MUST be of the same
medium--and they are surrounded by AIR.  Dose in the surrounding
AIR is stored in dose zone 1, and dose in the cones/cylinder
structure is stored in zone 2.

5.0
example cons3r
0.0
2.3
6
0.0, 0.8
1.0, 1.2
1.0, 1.5
1.3, 1.5
1.3, 0.5
2.3, 0.8
0.521, 0.01, 2, 0, 0
H2O
0.521, 0.01, 1, 0 ,0
AIR
```

15.3.4 CONESTAK

CONESTAK consists of a stack of coaxial truncated cones surrounded by a cylindrical wall. The vertices of the cones in each layer do not have to meet but the radii must not decrease as the depth increases. This CM can model scattering foils, primary collimators and many other components. CONESTAK has cylindrical symmetry.

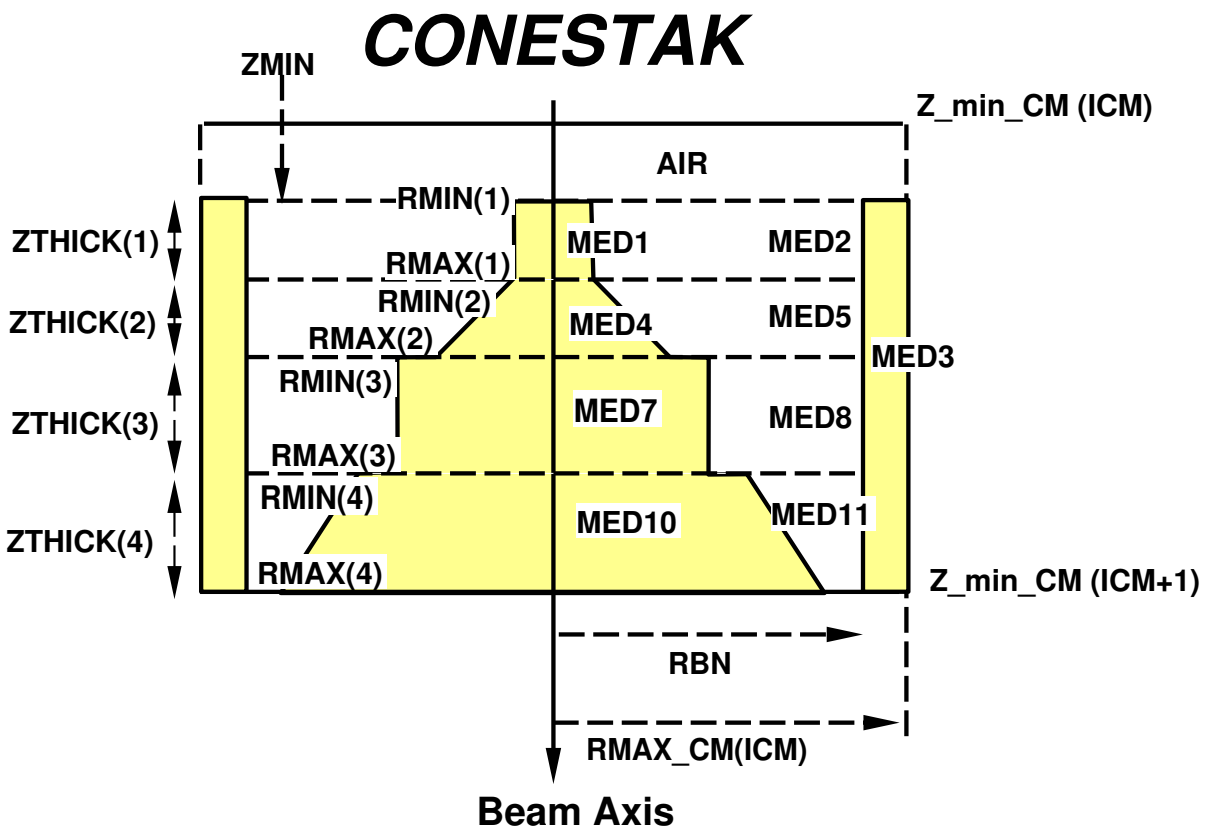


Figure 24: CONESTAK with 5 layers (ISCM_MAX=5). The outer wall has an inner radius defined by RBN and an outer radius equal to the outer boundary of the CM (RMAX_CM). Within each layer i, there is a cone with top and bottom radii defined by RMIN(i) and RMAX(i) respectively. Within each layer, there are three regions: the inner cone, the region between the cone and the outer wall (called the "outer cone"), and the outer wall. Regions 1,4,7,10,etc specify inner cones; regions 2,5,8,11,etc specify outer cones; and regions 3,6,9,etc specify the outer wall. If the user chooses to have an outer wall (by setting RBN $>$ 0) then the medium of region 3 , MED3, is input first, even before MED1 and MED2, and MED3 is automatically applied to regions 6,9,etc, so that the outer wall has the same composition in every layer. If the user chooses not to have an outer wall (by setting RBN $=$ 0), regions 3,6,9,etc shrink to zero. In specifying radii of cones, the following restrictions apply: RMAX(i) $\geq$ RMIN(i), and RMIN(i+1) $>$ RMAX(i).

The input format for CONESTAK, and an example of input file are given as follows.

```
CARDS CM_$CONESTAK (CONESTAK Rev 1.8)
*******************************
```

```
-1  dummy line to indicate start of CM

0  RMAX_CM(ICM_$CONESTAK) (F10.0): Outer radial boundary (cm).

1  TITLE_$CONESTAK (60A1):  Title of CM.

2  ZMIN_$CONESTAK, RBN_$CONESTAK (2F15.0):

      ZMIN_$CONESTAK:  Distance from front of first cone to
                       reference plane (Z=0),
      RBN_$CONESTAK:   Inner radius of outer wall
                       (Set to 0 if you do not want an outer wall)

3  ISCM_MAX_$CONESTAK (I5): Number of conical layers

Repeat 4 once for I=1,ISCM_MAX_$CONESTAK

4  ZTHICK_$CONESTAK(I), RMIN_$CONESTAK(I), RMAX_$CONESTAK(I) (3F15.0):

     ZTHICK_$CONESTAK(I):  Thickness of conical layer.
     RMIN_$CONESTAK(I):    Front radius of conical layer.
     RMAX_$CONESTAK(I):    Back radius of conical layer.
                       Note restrictions:
                       RMAX_$CONESTAK(I)>=RMIN_$CONESTAK(I)
                       RMIN_$CONESTAK(I+1)>=RMAX_$CONESTAK(I)

5 and 6 are only required if there is an outer wall (ie RBN_$CONESTAK~=0)

5  ECUT, PCUT, DOSE_ZONE, IREGION_TO_BIT in local region 3 (outer wall):
                                                 (2F15.0,2I5)

      ECUT, PCUT:  Cutoff energies for electrons and photons.
      DOSE_ZONE:   Dose scoring flag, non-zero to score dose
                   deposited in it
      IREGION_TO_BIT:   Bit setting number for the region

6  MED_IN (24A1):  Medium of local region 3
                   used to set MED_INDEX.

Repeat 7-10 for each conical layer

7  ECUT, PCUT, DOSE_ZONE, IREGION_TO_BIT in local region 1 (inside cone):
                                                 (2F15.0,2I5)

      ECUT, PCUT:  Cutoff energies for electrons and photons.
      DOSE_ZONE:   Dose scoring flag, 0 to score dose deposited in it
      IREGION_TO_BIT:   Bit setting number for the region
```

```
 8  MED_IN (24A1):  Medium of local region 1 (inside cone),
                    used to set MED_INDEX

 9  ECUT, PCUT, DOSE_ZONE, IREGION_TO_BIT in local region 2 (outside cone):
                                                  (2F15.0,2I5)

       ECUT, PCUT:  Cutoff energies for electrons and photons.
       DOSE_ZONE:   Dose scoring flag, 0 to score dose deposited in it
       IREGION_TO_BIT:   Bit setting number for the region

 10  MED_IN (24A1):  Medium of local region 2 (outside of cone),
                     used to set MED_INDEX
 Example
 *******
```

The following input example describes two conical layers. The first layer is a flat-top cone 1.0cm thick, with a radius at the top of 0.8cm and a radius at the base of 1.2cm. The second layer is a cylinder, also 1.0cm thick, of radius 1.2cm. The top cone is made of Cu and the bottom cylinder is made of Pb. The entire structure is encircled by a Pb wall with inner radius 4cm and outer radius 5cm.
In both layers, the medium between the cone and the outer wall is H2O.

Dose in the Cu cone will be scored in dose zone 1. Dose in the PB cylinder will appear in dose zone 2. The dose to the encircling PB wall will be in zone 3. And the dose to the water in both layers will be scored in zone 4. ECUT and PCUT in all cases is 0.521MeV and 0.01MeV respectively.

```
5.0                    RMAX_CM
cone and cylinder surrounded by PB wall
0.0, 4.0
2
1.0, 0.8, 1.2
1.0, 1.2, 1.2
0.521, 0.01, 3, 0
PB
0.521, 0.01, 1, 0
CU
0.521, 0.01, 4, 0
H2O
0.521, 0.01, 2, 0
PB
0.521, 0.01, 4, 0
H2O
```

15.3.5 FLATFILT

FLATFILT consists of a stacked set of truncated coaxial cones with an arbitrary number of cones on each level. The material in the cones need not be the same. Both the number of cones and the radii of cones in each layer can be specified independently. This CM can be used to model very complex beam flattening filters for photon beam simulations, including those interior to a conical collimator.

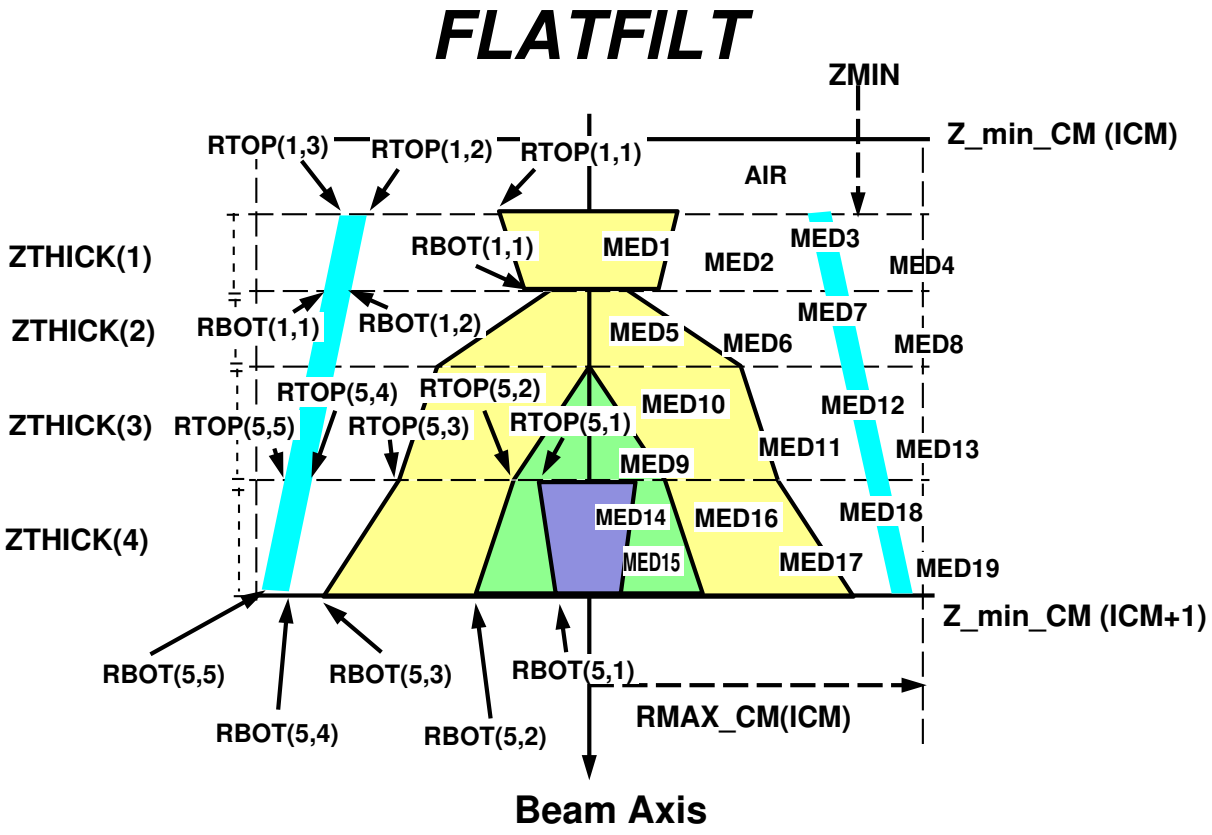


Figure 25: FLATFILT with 4 layers (ISCM_NO=4). Unlike CONESTAK, each layer can have an arbitrary number of cones. The number of cones in layer i is specified by ISSCM_NO(i). Within a layer i, each cone j is specified by its top and bottom radius, RTOP(i,j) and RBOT(i,j) respectively. Within a layer, cones cannot cross; so RTOP(i,j) < RTOP(i,j+1) and RBOT(i,j) < RBOT(i,j+1). For each layer i, ISSCM_NO(i) + 1 media must be specified. ISSCM_NO(i) media are required for the cones, and the extra medium is required for the region between the outermost cone and RMAX_CM. In the figure above, the 4 regions between outermost cones and RMAX_CM are composed of MED4, MED8, MED13 and MED19.

```
CARDS CM_$FLATFILT
******************

-1  Dummy line to indicate start of CM.

 0  RMAX_CM(ICM_$FLATFILT) (F10.0): Radius of outer boundary of CM (cm).
```

```
1  TITLE_$FLATFILT (60A1):  Title of CM.

2  ZMIN_$FLATFILT (F10.0):  Distance from front of CM (front of
    the first layer) to reference plane (Z=0),
    not including air gap.

3  ISCM_NO_$FLATFILT (I5): Number of layers.

Repeat 4-6 for I=1,ISCM_NO_$FLATFILT

4  ISSCM_NO_$FLATFILT(I), ZTHICK_$FLATFILT(I)  (I5,F15.0):

          ISSCM_NO_$FLATFILT(I): # cones in layer I(ex outer region).
          ZTHICK_$FLATFILT(I):   Thickness of layer I.

Repeat 5 for J=1,ISSCM_NO_$FLATFILT(I) all on one line in order
of increasing cone radius.

5  RTOP_$FLATFILT(I,J) (F15.0):
          Top radius of cone J in layer I.
          Note restriction: RTOP_$FLATFILT(I,J+1)>RTOP_$FLATFILT(I,J)

Repeat 6 for J=1,ISSCM_NO_$FLATFILT(I) all on one line in order
of increasing cone radius.

6  RBOT_$FLATFILT(I,J) (F15.0):
          Bottom radius of cone J in layer I.
          Note restriction: RBOT_$FLATFILT(I,J+1)>RBOT_$FLATFILT(I,J)

Repeat 7 and 8 for J=1,ISSCM_NO_$FLATFILT(I)+1 for every layer I.
When J=ISSCM_NO_$FLATFILT(I)+1, you are specifying ECUT, PCUT,
MED_IN, etc. for the region between the outermost cone and RMAX_CM
in layer I.

7  ECUT, PCUT, DOSE_ZONE, IREGION_TO_BIT  (2F15.0,2I5):

       ECUT, PCUT:  Cutoff energies for electrons and photons
                    in cone J.
       DOSE_ZONE:   Dose scoring flag for cone J.
    IREGION_TO_BIT: Bit setting for region defined by cone J.

8  MED_IN (24A1):  Medium of cone J
                   used to set MED_INDEX.

Example
*******

  The following example describes a FLATFILT with 2 layers.  The
```

```
RMAX of this FLATFILT is 2cm.  There is no air gap between the
first layer and the top of the CM.

The first layer is 0.3 cm thick and comprises a convex cone made
of H2O within a concave cone made of PB.  The convex H2O cone has
a top radius of 0.0cm and a bottom radius of 1.0cm.  The concave PB
cone has a top radius of 1.5cm and a bottom radius of 1.1cm.
The region between the outer, concave cone and RMAX_CM is AIR.
The second layer is also 0.3 cm thick and comprises a single
cylinder of H2O having radius 1cm (ie top radius=bottom
radius=1cm).  The region between the outer boundary of this
cylinder and RMAX_CM (an annular region) is AIR.

The dose to the AIR regions is scored in dose zone 1.  The dose
to H2O regions (the convex cone in the first layer and the cylinder
in the second layer) is scored in zone 2.  And the dose to PB
(the concave cone in the first layer) is scored in zone 3.
ECUT and PCUT for all regions are 0.521 MeV and 0.01 MeV
respectively.

2.00000,              RMAX_CM
flatfilt
0.0,                  ZMIN of first layer
2,                    no. of layers
2,  0.30,             no. of cones in first layer and thickness
0.0,1.5,              top radii of cones
1.0,1.1,              bottom radii of cones
1, 0.3                no. of cones in second layer and thickness
1.0                   top radius of cone
1.0                   bottom radius of cone
0.521, 0.01, 2, 0,    ECUT, PCUT, DOSE_ZONE, IREGION_TO_BIT
H2O                   MEDIUM -- for layer 1, cone 1
0.521, 0.01, 3, 0
PB                           -- for layer 1, cone 2
0.521, 0.01, 1, 0
AIR                          -- for region between cone 2 and RMAX
0.521, 0.01, 2, 0
H2O                          -- for layer 2, cone 1
0.521, 0.01, 1, 0
AIR                          -- for region between cone 1 and RMAX
```

15.3.6 CHAMBER

CHAMBER models a parallel-plate ion chamber in a container with top and bottom planes of arbitrary thickness and material. CHAMBER also models a combination of scattering foils, cylindrical collimators and an ion chamber, *etc.*. This CM is also useful for central-axis depth-dose calculations and for analysis of dose components due to particles coming from different parts of an accelerator. CHAMBER has cylindrical symmetry.

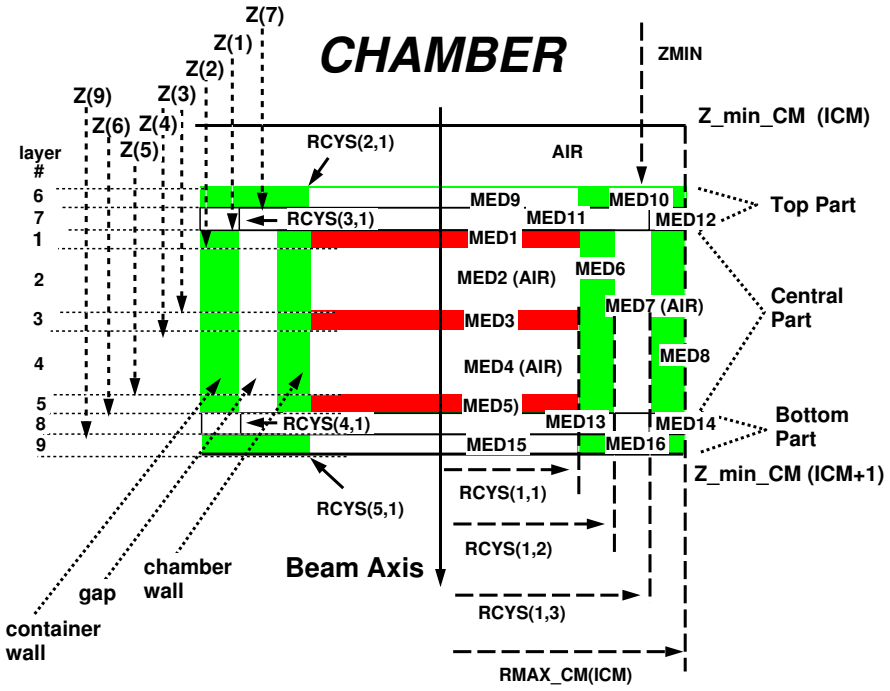


Figure 26: CHAMBER component module for an ionization chamber or any symmetric cylindrical-planar geometries. The CM shown consists of three parts: top, central(chamber), and bottom. The top and bottom parts each have 2 layers (`N_TOP = N_BOT`=2), and the chamber part has 5 layers (`N_CHM`=5). In general, the central part is required while the top and bottom parts are not. Each layer in both top and bottom parts is divided into an inner cylinder and an outer annulus. For a layer i in the top part, the outer radius of the cylinder (inner radius of the annulus) is given by `RCYS(i+1,1)`, and the outer radius of the annulus is `RMAX_CM`. For a layer j in the bottom part, the outer radius of the cylinder is given by `RCYS(j+N_TOP+1,1)`, and outer radius of the annulus is `RMAX_CM`. Within the top or bottom part, layers can have different `RCYS` and media; however, input for the top or bottom part is simplified if all layers within that part have the same `RCYS`, the same medium in the inner cylinders, and the same medium in the outer annuli. In the central part, all layers have the same outer radius, `RCYS(1,1)`, which also defines the inner radius of the chamber wall. The outer radius of the chamber wall is given by `RCYS(1,2)`, which, in turn, is the inner radius of the gap between the chamber wall and the container wall. The gap has outer radius `RCYS(1,3)`, which is also the inner radius of the container wall. The container wall has outer radius `RMAX_CM`.

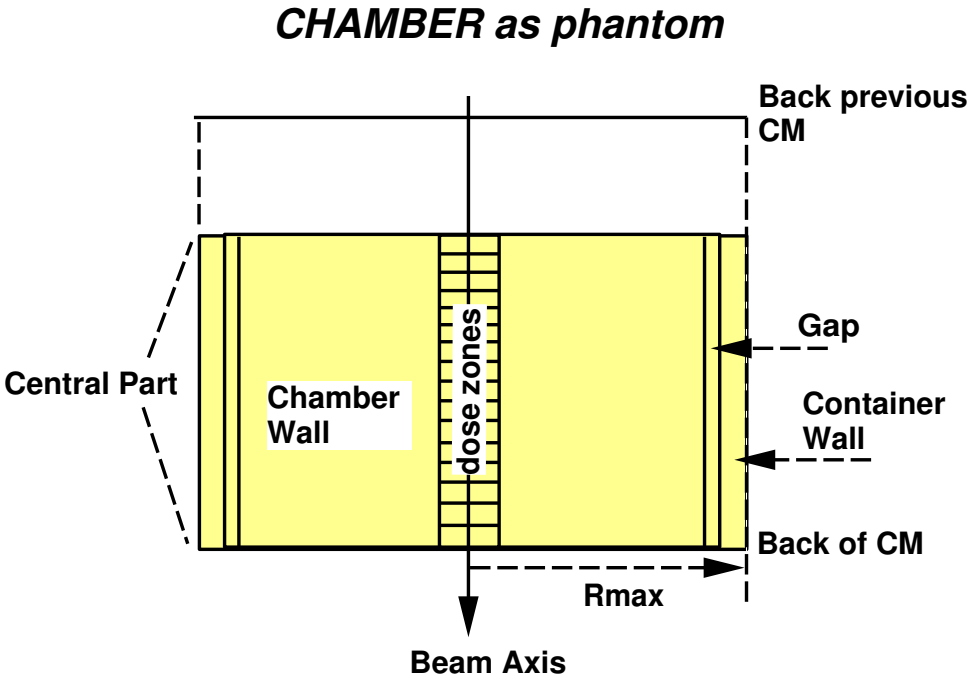


Figure 27: CHAMBER as a depth-dose phantom. Only the central (chamber) part of the CM is used. The layers within this part become the depth-dose zones, and, hence, all layers are composed of the same material (the phantom material). Depth resolution is determined by the number of layers. If all layers have the same thickness, then the input can be made more efficient by inputting the number of layers on the same line as the layer thickness, `ZTHICK`. Then you only have to input `ECUT`, `PCUT`, `DOSE_ZONE` and `MED` once for all layers. If `DOSE_ZONE` >0, the input value will be used for the first layer and then `DOSE_ZONE` will be incremented by 1 automatically for each subsequent layer. Alternatively, if there are N separate groups of layers, where the layers in each group have equal thickness, you can set `N_CHM_$CHAMBER` = -N. For each of the N groups of layers, you must specify the layer thickness and the number of layers in the group. Following this, you only have to specify `ECUT`, `PCUT`, `DOSE_ZONE` and `MED` once for all layers, similar to the case where ALL layers are of equal thickness. Note that all depth-dose zones have the same radius, determined by `RCYS`(1,1). A small `RCYS`(1,1) determines the depth-dose on the beam axis, while a larger `RCYS`(1,1) includes dose deposited further away from the beam axis. If the phantom extends beyond the radius of the dose zones, then the chamber wall, gap, and container wall can be composed of the same phantom material. This set-up can be very efficient for central axis depth-dose calculations when used in conjunction with range rejection because the large volume in the chamber wall leads to effective range rejection. However, it has been observed for electron beams that the lack of z boundaries in the wall region can lead to small step size effects. This effect can be minimised by decreasing `ESTEPE` (see section 11.4).

```
CARDS CM_$CHAMBER
****************

-1  Dummy line to indicate start of CM.

 0  RMAX_CM(ICM_$CHAMBER):( F10.0):   Maximum radius of component module

 1  TITLE_$CHAMBER (60A1):  Title of CM.

 2  ZMIN_$CHAMBER (F15.0): Distance from front surface of 1st cylinder to
                           reference plane (Z=0). Excludes any air gap.

 3  N_TOP_$CHAMBER, N_CHM_$CHAMBER, N_BOT_$CHAMBER (3I5)

      N_TOP_$CHAMBER:   Number of layers in top part (>= 0).
      N_CHM_$CHAMBER:   Number of layers in chamber itself (> 0 to input
                        chamber layers individually or if ALL layers have
                        the same thickness and medium; < 0 to
                        input -N_CHM_$CHAMBER groups of layers where
                        layers in each group have the same thickness and
                        ALL layers have the same MED).
      N_BOT_$CHAMBER:   Number of layers in bottom part (>= 0).
 ============================================================================
 4  Inputs for the top part (If N_TOP_$CHAMBER >0):
 ============================================================================

  If all layers in this part are identical, then in line (a) include
  NFLAG=N_TOP_$CHAMBER, otherwise repeat (a) to (e) for each of the
  layers.

   a) ZTHICK, RCYS_$CHAMBER , NFLAG (2F15.0,I5)
         ZTHICK (F15.0):        Thickness of each layer in top part
         RCYS_$CHAMBER (F15.0): Radius of inner cylinders in each layer
         N_TOP_$CHAMBER (I5):   Number of layers in top part

   b) ECUT,PCUT,DOSE_ZONE,IREGION_TO_BIT for inner cylinders
                                        (2F15.0,2I5,1-line):
       ECUT, PCUT: Cutoff energies for electrons and photons.
       DOSE_ZONE:  Dose scoring region for this region, 0=>no dose scored.
       IREGION_TO_BIT :  Bit # in LATCH designated to this region

   c) MED_IN (24A1):  Medium of inner cylinder (used for MED_INDEX)

   d) ECUT,PCUT,DOSE_ZONE,IREGION_TO_BIT for outer annuli
                                        (2F15.0,2I5,1-line):

   e) MED_IN (24A1):  Medium for outer annuli (used for MED_INDEX)

 ============================================================================
```

```
5  Inputs for the chamber/phantom part:
=========================================================================

 The chamber/phantom part has a central part of potentially many layers
 which may have different media and dimensions.  Outside this there
 are 3 cylindrical shells, called the chamber wall, gap, and container
 wall.  Each is a single material running the entire Z-span of the
 central part.

5.1) RCYS_$CHAMBER(1,1), RCYS_$CHAMBER(1,2), RCYS_$CHAMBER(1,3) (3F15.0)
       RCYS_$CHAMBER (1,1): Inner r of chamber wall=outer r central region
       RCYS_$CHAMBER (1,2): Outer r of chamber wall=inner r of gap
       RCYS_$CHAMBER (1,3): Inner r of container wall=outer r of gap


5.2) If N_CHM_$CHAMBER>0: If all layers in this part are identical, then
        in line (a) include NFLAG=N_CHM_$CHAMBER and input (b) once for
        all layers, otherwise repeat (a) to (c) for each of the layers.
     If N_CHM_$CHAMBER<0: Repeat (a) once for each of the
        -N_CHM_$CHAMBER groups of layers of equal thickness. In this case,
        NFLAG is the number of layers in the group.   Then input (b) once
        for all layers.

  a) ZTHICK, NFLAG  (F15.0,I5)
       ZTHICK:    Thickness of each layer in chamber part
                  (N_CHM_$CHAMBER>0) or of each layer in this particular
                  group of layers (N_CHM_$CHAMBER<0)
       NFLAG:     Number of layers in chamber IF all same
                  (N_CHM_$CHAMBER>0) or number of layers in the group
                  (N_CHM_$CHAMBER<0)

  b) ECUT,PCUT,DOSE_ZONE,IREGION_TO_BIT for chamber layers
                                            (2F15.0,2I5,one line):
       ECUT, PCUT: Cutoff energies for electrons and photons.
       DOSE_ZONE:  if all layers are of equal thickness or there are
                   groups of layers of equal thickness (ie
                   NFLAG=N_CHM_$CHAMBER or N_CHM_$CHAMBER<0) then, if
                   DOSE_ZONE>0 the dose is scored in regions
                   DOSE_ZONE, DOSE_ZONE+1,.., DOSE_ZONE+N_CHM_$CHAMBER-1
                   ie, a sequence of dose scoring zones are set up
                   automatically for all layers.
                   For single region at a time
                   Dose scoring region for this region,0=>no dose scored.
       IREGION_TO_BIT :  Bit # in LATCH designated to this region

  c) MED_IN (24A1):  Medium of chamber layers (used to set MED_INDEX)


  5.3) Inputs for the chamber wall:
```

```
  --------------------------------

  a) ECUT, PCUT, DOSE_ZONE, IREGION_TO_BIT, (2F15.0,2I5):
        ECUT, PCUT: Cutoff energies for electrons and photons.
        DOSE_ZONE:  Dose scoring region for this region,0=>no dose scored.
        IREGION_TO_BIT:  Bit # in LATCH designated to this region

  b) MED_IN (24A1):  Medium of chamber wall (used to set MED_INDEX)

  5.4) Inputs for the gap between chamber wall and container wall:
  ---------------------------------------------------------------

  a) ECUT, PCUT, DOSE_ZONE, IREGION_TO_BIT, (2F15.0,2I5):

  b) MED_IN (24A1):  Medium of gap (used to set MED_INDEX)


  5.5) Inputs for the container wall:
  ----------------------------------------

  a) ECUT, PCUT, DOSE_ZONE, IREGION_TO_BIT, (2F15.0,2I5):

  b) MED_IN (24A1):  Medium of container wall (used to set MED_INDEX)
=================================================================
6  Inputs for the bottom part (If N_BOT_$CHAMBER >0):
=================================================================

 5.6) If all layers in this part are identical, then in line (a) include
      NFLAG=N_BOT_$CHAMBER, otherwise repeat (a) to (e) for each of the
      layers.

  a) ZTHICK, RCYS_$CHAMBER , NFLAG (2F15.0,I5)
       ZTHICK:           Thickness of each layer in bottom part
       RCYS_$CHAMBER:    Radius of inner cylinders in bottom part

  b) ECUT,PCUT,DOSE_ZONE,IREGION_TO_BIT for inner cylinders
                                        (2F15.0,2I5,1-line):
       ECUT, PCUT:  Cutoff energies for electrons and photons.
       DOSE_ZONE:   Dose scoring flag, 0=>do not score dose.
       IREGION_TO_BIT :  Bit # in LATCH designated to this region

  c) MED_IN (24A1):  Medium of inner cylinders (used for MED_INDEX)

  d) ECUT,PCUT,DOSE_ZONE,IREGION_TO_BIT for outer annuli
                                       (2F15.0,2I5,1-line):
  e) MED_IN (24A1):  Medium of outer annuli (used for MED_INDEX)

=================================================================
7  Inputs for range rejection options:
```

```
=====================================================================

    MRNGE (I5)         0 or 1

       MRNGE     : = 1 to estimate thickness of the CHAMBER for
                     ECUTRR calculations in automated range rejection
                     (IREJCT_GLOBAL=1) (crude approx for 5 layers)
                   = 0 no ECUTRR calculation--range rejection will
                     still be done on a region-by-region basis

         Note that MRNGE only has an effect if automated range
         rejection is on (IREJCT_GLOBAL=1).

 Example
 *******

The following set of cards defines a chamber with 2 top layers, 3 chamber
layers, and 2 bottom layers.
The chamber wall is AL & the chamber container is CU. The detecting
material is air.
The air cavities are assigned as dose region 1 and the rest as region 2.

10.5; radius of CM
Chamber with 2 top layers, 3 chamber layers, 2 bottom layers
10.0; distance from front surface of the CM to the reference plane (z=0)
2,3,2;                    2 top layers, 3 chamber layers, 2 bottom layers
0.1,5.0,0;           first layer in the top part, 0.1cm thick, IR=5cm
0.521,0.010,2,2;  dose region # = 2
CU              ;  medium
0.521,0.010,2,2;
CU
0.2,5.0,0;           second layer is 0.2 cm thick, radius = 5.0 cm
0.521,0.010,2,2;  for inner cylinder (dose region # = 2)
AL
0.521,0.010,2,2;  for outer annulus
AL
5.0,5.2,10.0;     IR & OR of chamber wall, IR of container
0.2;              thickness of the first layer (air) in chamber part
0.521,0.010,1,2;  dose region # = 1
AIR
0.1;              thickness of the second layer (AL) in chamber part
0.521,0.010,2,2;  dose region # = 2
AL
0.2;              thickness of the third layer (air) in chamber part
0.521,0.010,1,2;  dose region # = 1
AIR
0.521,0.010,2,2;  chamber wall (dose region # = 2)
AL
0.521,0.010,2,2;  air gap betweem chamber wall and container wall
```

```
AIR
0.521,0.010,2,2;  chamber container
CU
0.1,5.0,2;        2 layers in bot. part have = thickness, IR
0.521,0.010,2,2;  for inner cylinders
AL
0.521,0.010,2,2;  for outer annuli
AL
0;                do not calculate ECUTRR
```

15.3.7 JAWS

JAWS is used for a set of paired bars, which can be bars in the collimator or applicator. The bars are of arbitrary thickness and material, and X or Y orientation. The outer boundary of JAWS has square symmetry but the jaws themselves can be very asymmetric. The input for JAWS can be tedious but the GUI has a feature which will automatically set the X and Y values to give an arbitrary on-axis rectangular photon beam field size at a given SSD based on tracing back to an arbitrary focus located on the central axis.

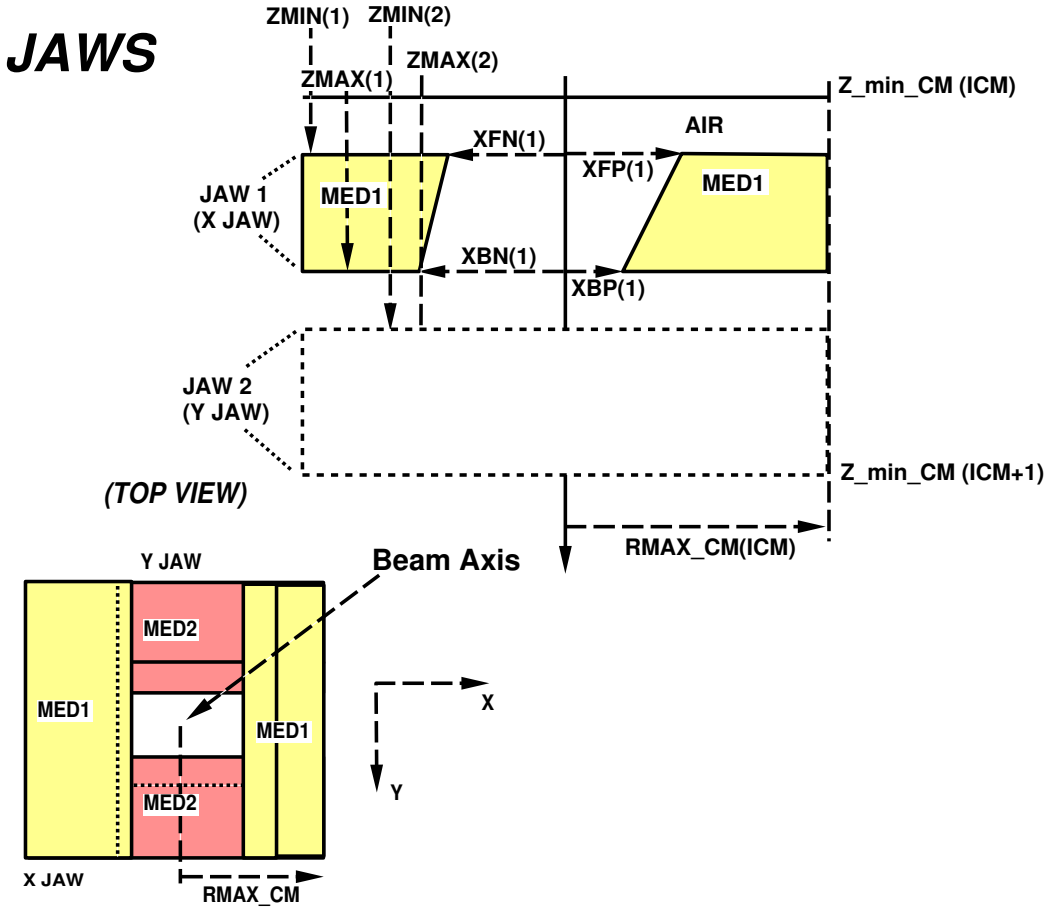


Figure 28: JAWS component module with paired X and Y jaws (`ISCM_MAX=2`). The cutaway view cuts right through the opening in the Y jaws, hence the representation of the Y jaws as a dashed rectangle. For a set of jaws i, the front opening of the jaws is specified by `XFP(i)` and `XFN(i)`. The back opening is specified by `XBP(i)` and `XBN(i)`. These coordinates are not shown for the Y jaws, however they are evident in the top view. The inside jaw surfaces are 2 planes, one connecting `XFP(i)` to `XBP(i)` and the other connecting `XFN(i)` to `XBN(i)` X jaws and openings between X jaws extend to $\pm$ `RMAX_CM` in the Y-direction; Y jaws and openings extend to $\pm$ `RMAX_CM` in the X direction. There must be a gap of at least 0.01 cm between the first set of jaws and the top of the CM (*i.e.* `ZMIN(1)-ZMIN_CM` $\geq$ 0.01) and between sets of jaws (*i.e.* `ZMIN(i)-ZMAX(i-1)` $\geq$ 0.01). Materials in each layer can be different but opposite jaws must be the same material.

The input format for JAWS, and an example of input file are given as follows.

```
CARDS CM_$JAWS(JAWS: Rev 1.8)
**************

-1  dummy line read in main used to separate input for CMs

 0  RMAX_CM(ICM_JAWS) (F10.0):
                      Perpendicular distance from Z-axis to boundary
                      surrounding component module.  This component
                      module has a square boundary.

 1  TITLE_$JAWS (60A1):  Title of CM.

 2  ISCM_MAX_$JAWS (I5):  Number of paired bars or jaws in CM.

 Repeat 3 and 4 for I=1,ISCM_MAX_$JAWS

 3  XY_CHOICE (A1):  indicate orientation of the paired bars/jaws
                      X means bars/jaws perpendicular to x axis
                     i.e. separation and movement is along x-axis

 4  ZMIN_$JAWS(I), ZMAX_$JAWS(I), XFP_$JAWS(I), XBP_$JAWS(I),
    XFN_$JAWS(I), XBN_$JAWS(I) (6F15.0)

      ZMIN_$JAWS(I): Distance front of bars/jaws to reference plane.
      ZMAX_$JAWS(I): Distance back of bars/jaws to reference plane.
      XFP_$JAWS(I):  positive bar/jaw x or y coodinate at front.
      XBP_$JAWS(I):  positive bar/jaw x or y coodinate at back.
      XFN_$JAWS(I):  negative bar/jaw x or y coodinate at front.
      XBN_$JAWS(I):  negative bar/jaw x or y coodinate at back.

 5   ECUT, PCUT, DOSE_ZONE, IREGION_to_BIT (2F15.0,2I5): for interior
                                                    (assumed to be AIR)

        ECUT, PCUT:     Cutoff energies for electrons and photons.
        DOSE_ZONE:      Dose scoring zone of air surrounding bars.
        IREGION_TO_BIT: This region associated with this bit in LATCH

 Repeat 6 and 7 for I=1,ISCM_MAX_$JAWS

 6   ECUT, PCUT, DOSE_ZONE, IREGION_TO_BIT (2F15.0,2I5)

        ECUT, PCUT:     Cutoff energies for electrons and photons in
                        jaw I.
        DOSE_ZONE:      Dose scoring zone for jaw I.
        IREGION_TO_BIT: Both bars of jaw I associated with this bit.

 7   MED_IN (24A1):  Medium of jaw I, used to set MED_INDEX.
```

Example
****

```
The following set of cards defines a pair of 5 cm-thick Al jaws.
The first set of bars open along the X axis.  The inside faces of
this jaw are vertical at |X|=3cm.  The second set of bars open along
the Y axis.  The inside
faces of this jaw are angled out slightly, beginning at |Y|=3cm at the top
of the jaw and ending at |Y|=3.05cm at the bottom of the jaw.
The first jaw starts at Z=30.0 cm.  Note the 0.01 cm airgap between
the two jaws.

Electrons will be followed in the CM down to kinetic energies of
10 keV (total energies of 0.521 MeV) and photons will be followed down
to energies of 1 keV.  The dose deposited in the air will be scored and
added to the dose deposited in the bars in dose zone 1.

15.0    RMAX_CM
JAWS: 2 Al jaws, 5cm thick
2
X
30.0, 35.0, 3.0, 3.0, -3.0, -3.0
Y
35.01, 40.01, 3.0, 3.05, -3.0, -3.05
0.0, 0.0, 1, 0
0.0, 0.0, 1, 0
AL
0.0, 0.0, 1, 0
AL
```

15.3.8 DYNJAWS

DYNJAWS is a version of JAWS (see Section 15.3.7 above) which allows the user to specify changing jaw settings (Z positions and opening coordinates) over the course of a simulation. There are two dynamic modes: "step-and-shoot" (MODE_$DYNJAWS=2) and "dynamic" (MODE_$DYNJAWS=1). If being run in one of these modes, then the user must specify the jaw settings in a separate file. The format of this file is:

```
TITLE
NFIELD
FOR I=1,NFIELD[
INDEX(I)
FOR J=1,ISCM_MAX[
ZMIN(J,I),ZMAX(J,I),XFP(J,I),XBP(J,I),XFN(J,I),XBN(J,I)
]
]
```

where:

TITLE is a title line for the file.

NFIELD = the number of "sets" of jaw settings, or fields.

INDEX(I) = the index of field I, where 0≤INDEX(I)≤1 and INDEX(I)>INDEX(I-1).

ISCM_MAX = the number of jaws.

ZMIN(J,I) = the Z position of the front of jaw J for field I.

ZMAX(J,I) = the Z position of the back of jaw J for field I.

XFP(J,I) = the front X or Y coordinate (depending on jaw orientation) of the positive bar of jaw J for field I.

XBP(J,I) = the back X or Y coordinate of the positive bar of jaw J for field I.

XFN(J,I) = the front X or Y coordinate of the negative bar of jaw J for field I.

XBN(J,I) = the back X or Y coordinate of the negative bar of jaw J for field I.

The parameter, INDEX(I), is used to determine which field is used for a given primary history simulated. At the beginning of each primary history a random number, RND, on [0,1] is chosen and compared to the INDEX(I). The field no. used is the lowest value of I for which RND≤INDEX(I). If the user has selected "step-and-shoot" mode, then the jaw settings for field I are simply used. If the user has selected "dynamic" mode, then the jaw settings are a linear interpolation between field I-1 and I based on the value of RND. For example, the Z position of the front of jaw J, ZMIN(J), would be given by:

$$\mathtt{ZMIN}(\mathtt{J}) = \mathtt{ZMIN}(\mathtt{J},\mathtt{I}-1) + \frac{[\mathtt{ZMIN}(\mathtt{J},\mathtt{I}) - \mathtt{ZMIN}(\mathtt{J},\mathtt{I}-1)]}{[\mathtt{INDEX}(\mathtt{I}) - \mathtt{INDEX}(\mathtt{I}-1)]} \times [\mathtt{RND} - \mathtt{INDEX}(\mathtt{I}-1)] \quad (9)$$

In this way, the "dynamic" mode simulates jaw motion while the beam is on.

Note that the user can also select "static" mode, in which the inputs are identical to JAWS and jaw settings are fixed.

An example file containing the jaw settings for 3 fields for a single jaw are contained in the file `$OMEGA_HOME/beamnrc/CMs/sample_dynjaws.sequence`.

The input format for DYNJAWS, and an example of an input are given as follows.

```
CARDS CM_$DYNJAWS(JAWS: Rev 1.8)
**************

-1  dummy line read in main used to separate input for CMs

 0  RMAX_CM(ICM_JAWS) (F10.0):
                      Perpendicular distance from Z-axis to boundary
                      surrounding component module.  This component
                      module has a square boundary.

 1  TITLE_$DYNJAWS (60A1):  Title of CM.

 2  ISCM_MAX_$DYNJAWS, MODE_$DYNJAWS (2I5)

        ISCM_MAX_$DYNJAWS = Number of paired bars or jaws in CM.
            MODE_$DYNJAWS = 0 for static settings of jaw openings
                            1 for dynamic settings with simulated
                              jaw movement while beam is on
                            2 for step-and-shoot jaw movement--beam
                              off while jaw settings are changed

Repeat 3 (if MODE_$DYNJAWS=1,2) or 3 and 4 (if MODE_$DYNJAWS=0)
           for I=1,ISCM_MAX_$DYNJAWS

3  XY_CHOICE (A1):  indicate orientation of the paired bars/jaws
                     X means bars/jaws perpendicular to x axis
                    i.e. separation and movement is along x-axis

Next input is only required if MODE_$DYNJAWS=0 (static)

4  ZMIN_$DYNJAWS(I), ZMAX_$DYNJAWS(I), (XFP_$DYNJAWS(I), XBP_$DYNJAWS(I),
   XFN_$DYNJAWS(I), XBN_$DYNJAWS(I)) (6F15.0)

     ZMIN_$DYNJAWS(I): Distance front of bars/jaws to reference plane.
     ZMAX_$DYNJAWS(I): Distance back of bars/jaws to reference plane.
     XFP_$DYNJAWS(I):  positive bar/jaw x or y coodinate at front.
     XBP_$DYNJAWS(I):  positive bar/jaw x or y coodinate at back.
     XFN_$DYNJAWS(I):  negative bar/jaw x or y coodinate at front.
     XBN_$DYNJAWS(I):  negative bar/jaw x or y coodinate at back.
```

```
Next input is only required if MODE_$DYNJAWS=1 or 2

4a jaws_file (A80)

     jaws_file: The full name of a file containing jaw opening data
                in the following format:

   NFIELDS_$DYNJAWS (I10)
   FOR J=1,NFIELDS_$DYNJAWS[
    INDEX_$DYNJAWS(J) (F15.0)
    (ZMIN_$DYNJAWS(I),ZMAX_$DYNJAWS(I),XFP_$DYNJAWS(I),XBP_$DYNJAWS(I),
     XFN_$DYNJAWS(I),XBN_$DYNJAWS(I), I=1,ISCM_MAX_$DYNJAWS)
   ]

   where:

   NFIELDS_$DYNJAWS:  Total number of jaw settings.
   INDEX_$DYNJAWS(J):  Index of setting J. 0 <= INDEX_$DYNVMLC(J) <= 1
                      and INDEX_$DYNVMLC(J) >= INDEX_$DYNVMLC(J-1).  This
                      number is compared to a random number on [0,1] at
                      the start of each history; if the random number is
                      <= INDEX_$DYNVMLC(J), then, if MODE_$DYNJAWS=2,
                      settings J are used.  If MODE_$DYNJAWS=1, then
                      the settings used are a linear interpolation
                      between fields J-1 and J based on the random
                      number selected.  See manual for more details.
   ZMIN_$DYNJAWS(I),ZMAX_$DYNJAWS(I),XFP_$DYNJAWS(I),XBP_$DYNJAWS(I),
   XFN_$DYNJAWS(I),XBN_$DYNJAWS(I):
                      See description of input 4.  These are defined
                      for each field, J.

5   ECUT, PCUT, DOSE_ZONE, IREGION_to_BIT (2F15.0,2I5): for interior
                                                  (assumed to be AIR)

       ECUT, PCUT:     Cutoff energies for electrons and photons.
       DOSE_ZONE:      Dose scoring zone of air surrounding bars.
       IREGION_TO_BIT: This region associated with this bit in LATCH

Repeat 6 and 7 for I=1,ISCM_MAX_$DYNJAWS

6   ECUT, PCUT, DOSE_ZONE, IREGION_TO_BIT (2F15.0,2I5)

       ECUT, PCUT:     Cutoff energies for electrons and photons in
                       jaw I.
       DOSE_ZONE:      Dose scoring zone for jaw I.
       IREGION_TO_BIT: Both bars of jaw I associated with this bit.

7   MED_IN (24A1):  Medium of jaw I, used to set MED_INDEX.
```

```
 Example
 *******

The following input defines a single tungsten jaw oriented in the Y
direction operating in step-and-shoot (MODE_$DYNJAWS=2) mode.
The jaw settings (Z position, opening coordinates) are defined
in the file $OMEGA_HOME/beamnrc/CMs/dynjaw.opening.file
(included with the distribution).

40.0, RMAX
CM #7: jaws set for a 10x10cm field at SSD=100 cm
1,2, # PAIRED BARS OR JAWS, MODE
Y
$OMEGA_HOME/beamnrc/CMs/dynjaw.opening.file
0.7, 0.01, 0, 21,
0.7, 0.01, 0, 13,
W700ICRU
```

15.3.9 SYNCJAWS

SYNCJAWS is a version of DYNJAWS (See Section 15.3.8 above) in which the "dynamic" or "step-and-shoot" jaw settings (opening coordinates and Z-positions) specified in a file are synchronized with the settings of any other synchronized CMs (SYNCVMLC, SYNCMLCE, SYNCHDDMLC) in the accelerator and with the motion of source 20 (synchronized phase space) or source 21 (synchronized BEAM treatment head simulation) in DOSXYZnrc[15]. SYNCJAWS was contributed by Julio Lobo and Tony Popescu[24].

Inputs to SYNCJAWS are identical to those for DYNJAWS. The DYNJAWS input, `INDEX(I)`, used in dynamic and step-and-shoot modes to determine the cumulative probability of field `I` being used for an incident history, is now interpreted as `muIndex(I)`, the fraction of total monitor units delivered up to and including field `I`.

Similar to DYNJAWS, at the beginning of each primary history, a random number, `MU_RND`$\in$ [0,1], is compared to the `muIndex(I)` and used to determine the field number and jaw settings used. In dynamic mode, the equation for calculating jaw settings is the same as Equation (6) above, with `muIndex` replacing `INDEX` and `MU_RND` replacing `RND`.

The only difference between DYNJAWS and SYNCJAWS is that the value of `MU_RND` used by SYNCJAWS for each primary history is the same as that used by any other synchronized CMs in the accelerator. `MU_RND` is generated by the first (most upstream) synchronized CM (which could be SYNCJAWS) and is then passed on to any downstream synchronized CMs. Thus, the dynamic motion of the jaws can be coordinated (synchronized) with the dynamic motion of all other synchronized CMs in the the accelerator. Moreover, if an accelerator with synchronized CMs has been compiled as a shared library for use in DOSXYZnrc source 20 (synchronized phase space source) or source 21 (synchronized BEAM treatment head simulation), then `MU_RND` is passed from the accelerator simulation to DOSXYZnrc. This allows the dynamic motion of the source plane in DOSXYZnrc sources 20 and 21 to be synchronized with the settings of the CMs in the accelerator. Note that it is up to the user to set up the synchronization between CMs within the accelerator, using the files defining jaw settings and, for MLC's, leaf opening coordinates for each field, and between the CM parameters and the motion of DOSXYZnrc source 20 or 21. See the DOSXYZnrc Users Manual[15] for more information on sources 20 and 21.

An example of a file specifying dynamic jaw settings for SYNCJAWS is included with the distribution. This file is `$OMEGA_HOME/beamnrc/CMs/sample_syncjaws.sequence`.

15.3.10 APPLICAT

APPLICAT is used for a set of rectangular scrapers. Each scraper is defined by the outer region of two concentric rectangles, the inner region being air. The scrapers are of arbitrary thickness, width in both X and Y directions, and position relative to the reference plane ($Z = 0$). The scrapers can be of different materials. This CM may be used for modelling the square applicator found in electron beams although it does not allow for a bevelled edge. The outer boundary of APPLICAT is a square centered on the beam axis.

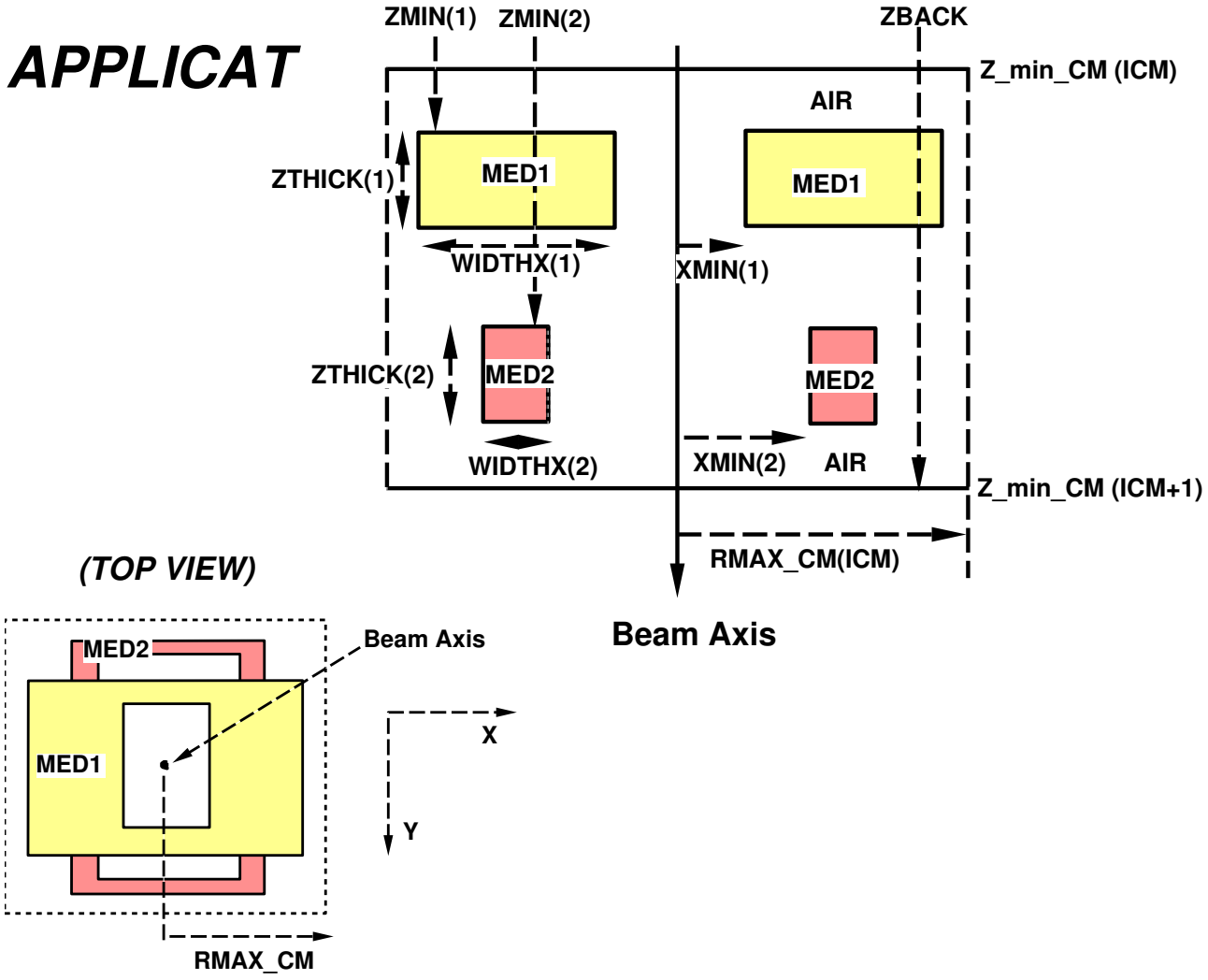


Figure 29: An APPLICAT with 2 scrapers (N=2). Each scraper i has a rectangular opening in the centre with half-width in the X-direction XMIN(i) and half-width in the Y-direction YMIN(i) . The width of scraper i in the X-direction is given by WIDTHX(i) and in the Y-direction by WIDTHY(i). The cross section does not show the Y dimensions of the scrapers, however the Y dimensions are evident in the top view. The minimum space between two scraper is 0.01 cm of air. There is also a minimum 0.01 cm air gap at the front (ie between Z_min_CM(1) and ZMIN(1)) and back (*i.e.* between ZBACK and the back of the last scraper) of the CM.

The input format for APPLICAT, and an example of input file are given as follows.

```
CARDS CM_APPLICAT
*****************

-1  Dummy line to indicate start of CM.

 0  RMAX_CM(ICM_$APPLICAT) (F10.0): Half-width of outer boundary of CM (cm).

 1  TITLE_$APPLICAT (60A1):  Title of CM.

 2  ZBACK_$APPLICAT (F15.0): Z of back face of the CM
                         (air will be added if necessary below the
                         last applicator)

     Note that there is always an air gap (thickness =
     AIRGAPMIN) in the front and the back of this CM.
     Therefore ZBACK_$APPLICAT should be >= Z of the back face of the
     last scraper + AIRGAPMIN.

 3  N_$APPLICAT, ISHAPE (2I5):
      N_$APPLICAT:  Number of scrapers in the CM.
      ISHAPE:       Index of applicators' shape, default to square, 1 for
                    rectangle.

 Repeat 4 for I=1,N_$APPLICAT.

 4  ZMIN_$APPLICAT(I), ZTHICK_$APPLICAT(I), XMIN_$APPLICAT(I),
    YMIN_$APPLICAT(I),
    WIDTHX_$APPLICAT(I), WIDTHY_$APPLICAT(I),
    DOSE_ZONE,IREGION_TO_BIT (6F15.0,2I4):

      ZMIN_$APPLICAT(I):   Z of front face of scraper I.
                           Note that ZMIN_$APPLICAT(1)-
                           Z_min_CM must be >= AIRGAPMIN.
      ZTHICK_$APPLICAT(I): Thickness of scraper I.  Note that
                           ZMIN_$APPLICAT(I+1)-(ZMIN_$APPLICAT(I)+
                           ZTHICK_$APPLICAT(I))
                           must be >= AIRGAPMIN.
      XMIN_$APPLICAT(I):   (ISHAPE=1)  Half-width of inner opening in
                                       x in scraper I.
                           (ISHAPE~=1) Half-width of inner opening in x
                                       and y in scraper I.
      YMIN_$APPLICAT(I):   (ISHAPE=1)  Half-width of inner opening in y in
                                       scraper I.
                           (ISHAPE~=1) Not required.
      WIDTHX_$APPLICAT(I): (ISHAPE=1)  Width of bar in x (ie material
                                       surrounding inner opening) of
                                       scraper I.
```

```
                          (ISHAPE~=1) Width of bar in x and y for
                                      scraper I.
     WIDTHY_$APPLICAT(I): (ISHAPE=1)  Width of bar in y (ie material
                                      surrounding inner opening) of
                                      scraper I.
                          (ISHAPE~=1) Not required.
     DOSE_ZONE:           Dose scoring zone for the scraper bar.
     IREGION_TO_BIT:      Bit setting number for the scraper bar.

     Note restrictions to allow air gaps between scrapers and
     before the first scraper:
      ZMIN_$APPLICAT(1)-Z_min_CM >= AIRGAPMIN
      ZMIN_$APPLICAT(I+1)-(ZMIN_$APPLICAT(I)+ZTHICK_$APPLICAT(I)) >=
      AIRGAPMIN

 6   ECUT, PCUT, DOSE_ZONE_AIR,IREGION_TO_BIT_AIR (2F15.0,2I5):

       ECUT, PCUT:     Cutoff energies for electrons and photons for
                       both the bars and the surrounding (air) region
       DOSE_ZONE_AIR:  Dose scoring zone for the surrounding region
  IREGION_TO_BIT_AIR:  Bit set number for the surrounding (air) region

 Repeat 7 for I=1,N_$APPLICAT.

 7   MED_IN (24A1):  Medium of the bar of scraper I, used to
                     set MED_INDEX.

 Example
 *******

The following set of cards defines an applicator consisting of 2
0.2cm-thick Al scrapers.  The scrapers are separated by 8cm of air.
Scrapers can be thought of as made of 4 bars arranged in a rectangle
orthogonal to the Z axis.  For both scrapers in this example, the half-
width of the openings created by the bars is 2cm in x, 4cm in y, and the
width of the bars themselves is 1cm in x, 1.5cm in y.  The front scraper
starts at Z=60.5 cm.

Electrons will be followed in the CM down to kinetic energies of
10 keV (total energies of 0.521 MeV) and photons will be followed down
to energies of 1 keV.  The dose deposited in the air will be scored and
added to the dose from the other regions in dose scoring zone 1, and the
dose deposited in both scrapers will be scored and added to the dose
from the other regions in dose scoring zone 2.  There is a minimum 0.1 cm
air gap at the front and back of the scrapers CM so that the applicator
bars are completely surrounded by air.

10.0,    RMAX_CM
Applicators: 0.2cm Al at 60.5cm and 68.7cm, ECUT=0.521, PCUT=0.01
```

```
100.0,   extended air to Z=100 cm
2, 1,    two rectangular applicators
60.5, 0.2, 2.0, 4.0, 1.0, 1.5, 2,3
68.7, 0.2, 2.0, 4.0, 1.0, 1.5, 2,2
0.521, 0.01, 1, 0
AL521ICRU
AL521ICRU
```

15.3.11 CIRCAPP

CIRCAPP is similar to APPLICAT, however, it is used to model scrapers that have circular openings. The scrapers retain their rectangular outer edges, but the opening is now defined as a circle concentric with the rectangle defining the outer edges. Similar to APPLICAT, CIRCAPP does not allow bevelled edges. The outer boundary of CIRCAPP is a square centered on the beam axis.

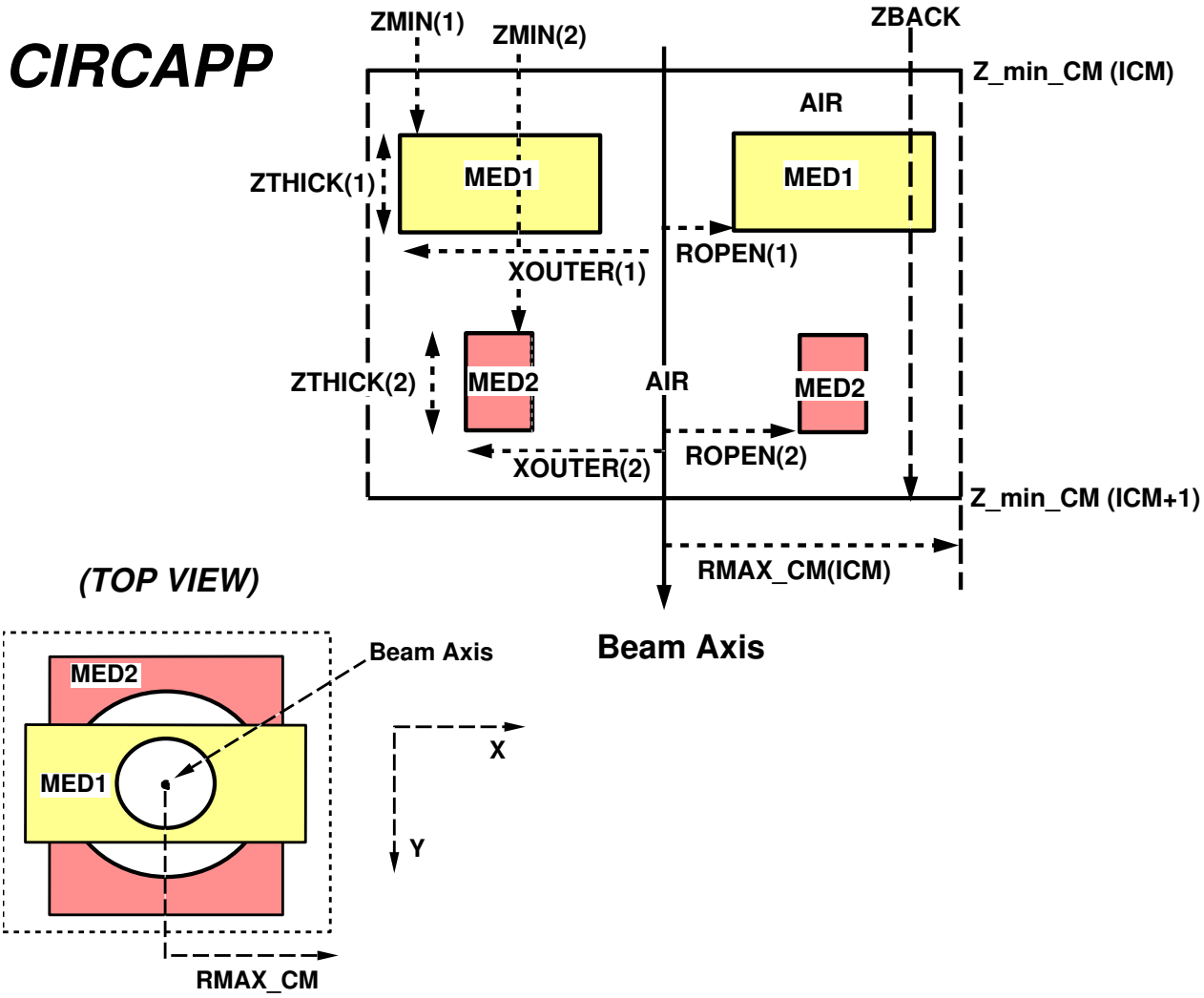


Figure 30: A CIRCAPP with 2 scrapers (N=2). Each scraper i has half-widths in the X- and Y-directions given by XOUTER(i) and YOUTER(i) and a circular opening in the centre with radius ROPEN(i). The Y dimensions are not shown in the cross section, however, they are apparent in the top view. As in APPLICAT, there are minimum air gaps of 0.01 cm between scrapers, between Z_min_CM(1) and the first scraper and between ZBACK and the last scraper.

The input format for CIRCAPP, and an example of input file are given as follows.

```
CARDS CM_CIRCAPP
*****************

-1  Dummy line to indicate start of CM.

 0  RMAX_CM(ICM_$CIRCAPP) (F10.0): Half-width of outer boundary of CM (cm).

 1  TITLE_$CIRCAPP (60A1):  Title of CM.

 2  ZBACK_$CIRCAPP (F15.0): Z of back face of the CM
                          (air will be added if necessary below the
                          last scraper)

     Note that there is always an air gap (thickness =
     AIRGAPMIN) in the front and the back of this CM.
     Therefore ZBACK_$CIRCAPP should be >= Z of the back face of the
     last scraper + AIRGAPMIN.

 3  N_$CIRCAPP:
      N_$CIRCAPP:  Number of scrapers in the CM.

 Repeat 4 for I=1,N_$CIRCAPP.

 4  ZMIN_$CIRCAPP(I), ZTHICK_$CIRCAPP(I), ROPEN_$CIRCAPP(I),
    XOUTER_$CIRCAPP(I), YOUTER_$CIRCAPP(I),
    DOSE_ZONE,IREGION_TO_BIT (6F15.0,2I4):

      ZMIN_$CIRCAPP(I):   Z of front face of scraper I. Note that
                          ZMIN_$CIRCAPP(1)-Z_min_CM must be >= AIRGAPMIN.
      ZTHICK_$CIRCAPP(I): Thickness of scraper I.  Note that
                           ZMIN_$CIRCAPP(I+1)-(ZMIN_$CIRCAPP(I)+
                           ZTHICK_$CIRCAPP(I)) must be >= AIRGAPMIN.
      ROPEN_$CIRCAPP(I):   Radius of inner opening in scraper I.
      XOUTER_$CIRCAPP(I):  X half-width of outer edge of scraper I.
      YOUTER_$CIRCAPP(I):  Y half-width of outer edge of scraper I.
      DOSE_ZONE:           Dose scoring zone for the scraper bar.
      IREGION_TO_BIT:      Bit setting number for the scraper bar.

      Note restrictions to allow air gaps between scrapers and
      before the first scraper:
       ZMIN_$CIRCAPP(1)-Z_min_CM >= AIRGAPMIN
       ZMIN_$CIRCAPP(I+1)-(ZMIN_$CIRCAPP(I)+ZTHICK_$CIRCAPP(I)) >=
       AIRGAPMIN

 6   ECUT, PCUT, DOSE_ZONE_AIR,IREGION_TO_BIT_AIR (2F15.0,2I5):

        ECUT, PCUT:     Cutoff energies for electrons and photons for
```

```
                        both the bars and the surrounding (air) region
        DOSE_ZONE_AIR:  Dose scoring zone for the surrounding region
   IREGION_TO_BIT_AIR:  Bit set number for the surrounding (air) region

 Repeat 7 for I=1,N_$CIRCAPP.

 7   MED_IN (24A1):  Medium of the bar of scraper I, used to
                     set MED_INDEX.

 Example
 *******

The input defines a circular applicator with 2 scrapers, one of Pb with
radius of opening = 0.6cm, X and Y half-widths of 1.0cm, and thickness
0.5cm. The other scraper, consisting of Al, has an opening with radius
2.4cm, X half-width = 4.6cm, Y half-width = 3.2cm, and thickness 1.0cm.
The scrapers are separated by 0.1cm of air. The front scraper starts at
Z=1.1cm and the second at 1.6 cm.

Electrons will be followed down to kinetic energies of 10 keV (total
energies of 0.521 MeV) and photons will be followed down to 1 keV.  The
dose deposited in the air will be scored in dose zone 1, and the dose in
the 2 scrapers will be scored in dose zones 2 and 3. There is a minimum
0.1 cm air gap at the front and back of the CIRCAPP CM.

14.0,    RMAX_CM
circular applicator
44.0,     extended air to Z=44 cm
2,      two scrapers
1.10,  0.50,  0.600, 1.0, 1.0, 2, 6,
1.60,  1.00,  2.40,  4.6, 3.2, 3, 7,
0.521, 0.01, 1, 0,
PB521ICRU
AL521ICRU
```

15.3.12 PYRAMIDS

The PYRAMIDS CM is used to model pyramid-shaped structures comprising one or more layers in the path of the beam. Each layer has three distinct regions: the central region (the pyramid), the surrounding region and the outer region (beyond the outer edges of the layer). The central and outer regions default to air but can also be filled with a user-specified medium (assumed the same for the central and outer regions within a layer). PYRAMIDS is useful for modelling rectangular collimators and beam blocks. This CM has a square outer boundary centered on the beam axis.

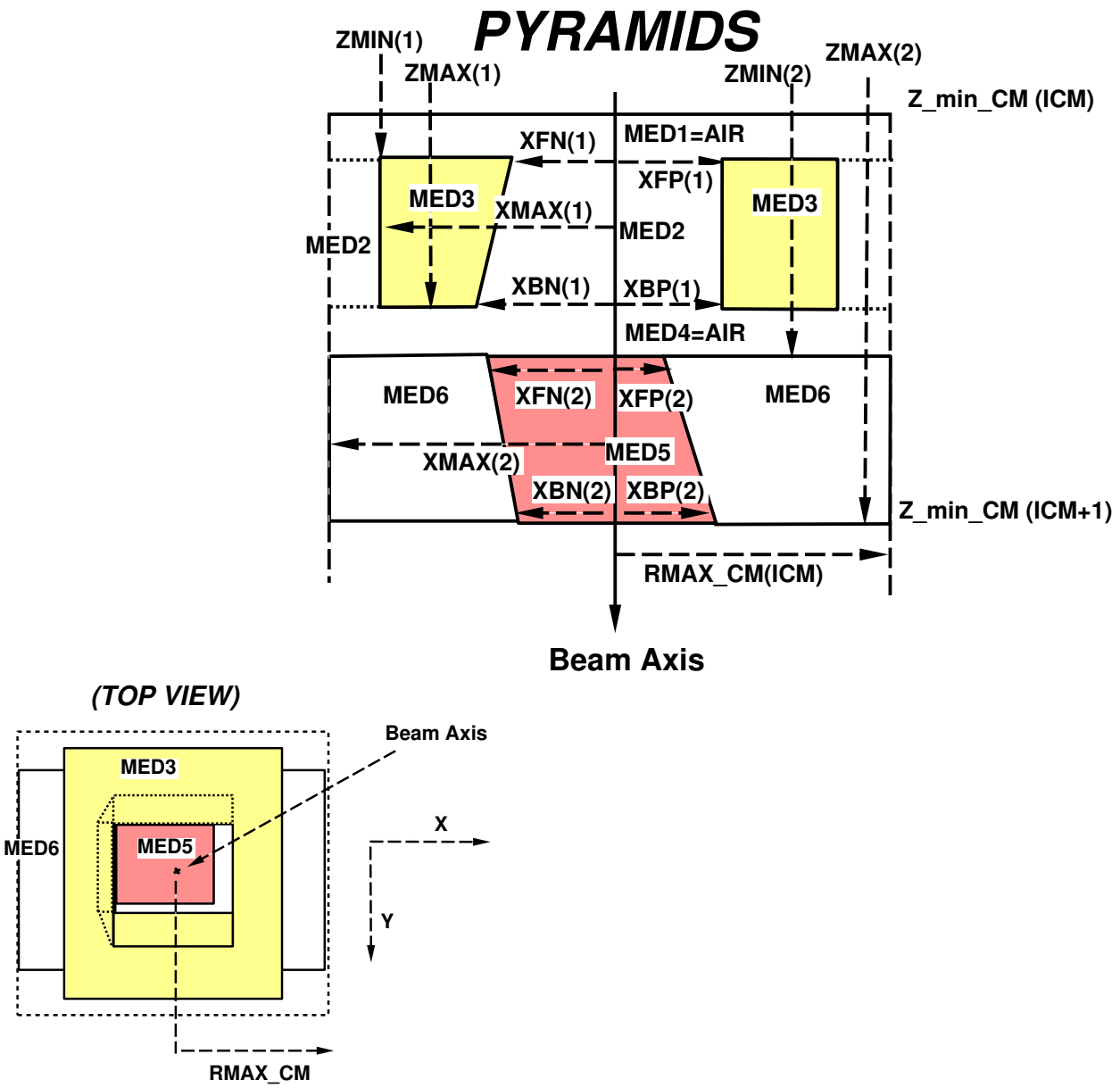


Figure 31: A PYRAMIDS component module with 2 layers (`ISCM_MAX`=2). Within a layer i, the front of the central region is specified by `XFP(i)`, `XFN(i)`, `YFP(i)`, `YFN(i)`, and the back is defined by `XBP(i)`, `XBN(i)`, `YBP(i)`, `YBN(i)`. The central region's inner faces are the planes connecting rectangle defining the front of the region with that defining the back of the region. The X and Y outer edges of layer i are defined by `XMAX(i)` and `YMAX(i)` respectively. The figure does not show the Y dimensions (*i.e.* `YFN(i)`, `YFP(i)`, `YMAX(i)`, *etc.*.), however the Y dimensions are evident in the top view. Layers must be separated by at least 0.01cm (`ZMIN(i+1)-ZMAX(i)` $\geq$ 0.01cm). By setting `IFILL=1`, the user can set the medium filling the central and outer regions in each layer (`MED2` and `MED5`). The gaps between layers are assumed to be filled with air.

The input format for PYRAMIDS and an example input are given below.

```
CARDS CM_PYRAMIDS(Rev 1.5)
*****************

-1  dummy line filled with ***  read in main

 0  RMAX_CM(ICM_$PYRAMIDS):  Perpendicular distance from Z-axis
                             to boundary surrounding component
                             This component module has a square
                             boundary.

 1  TITLE_$PYRAMIDS (60A1):  Title of CM.

 2  ISCM_MAX_$PYRAMIDS, IFILL_$PYRAMIDS (2I5):

      ISCM_MAX_$PYRAMIDS: Number of layers in CM.
         IFILL_$PYRAMIDS: Set to 0 (default) if all central
                          and outer regions contain air, or 1
                          if central and outer regions contain
                          user-specified media.

 Repeat 3 for I=1,ISCM_MAX_$PYRAMIDS

 3  ZMIN_$PYRAMIDS(I), ZMAX_$PYRAMIDS(I), XFP_$PYRAMIDS(I),
    XBP_$PYRAMIDS(I), XFN_$PYRAMIDS(I), XBN_$PYRAMIDS(I),
    YFP_$PYRAMIDS(I), YBP_$PYRAMIDS(I), YFN_$PYRAMIDS(I),
    YBN_$PYRAMIDS(I),XMAX_$PYRAMIDS(I), YMAX_$PYRAMIDS(I) (12F15.0):

      ZMIN_$PYRAMIDS(I): Distance from front of layer I to reference plane.
      ZMAX_$PYRAMIDS(I): Distance from back of layer I to reference plane.

      XFP_$PYRAMIDS(I):  positive x dimension of central region at front
      XBP_$PYRAMIDS(I):  positive x dimension of central region at back
      XFN_$PYRAMIDS(I):  negative x dimension of central region at front
      XBN_$PYRAMIDS(I):  negative x dimension of central region at back

      YFP_$PYRAMIDS(I):  positive y dimension of central region at front
      YBP_$PYRAMIDS(I):  positive y dimension of central region at back
      YFN_$PYRAMIDS(I):  negative y dimension of central region at front
      YBN_$PYRAMIDS(I):  negative y dimension of central region at back

      XMAX_$PYRAMIDS(I): outer x edge of layer (absolute value)
      YMAX_$PYRAMIDS(I): outer y edge of layer (absolute value)

      Note restriction to leave airgap between layers:
           ZMIN_$PYRAMIDS(I+1)-ZMAX_$PYRAMIDS(I) >= AIRGAPMIN

 4   ECUT, PCUT, DOSE_ZONE, IREGION_TO_BIT (2F15.0,2I5): for AIR
```

```
                                    between layers, and,
                                    if IFILL_$PYRAMIDS=0,
                                    in all central and outer regions.
      ECUT, PCUT:   Cutoff energies for electrons and photons.
       DOSE_ZONE:   Dose scoring zone of air.
  IREGION_TO_BIT:   mapping of region to bit for LATCH

Repeat 7-8 (IFILL_$PYRAMIDS=0) or 5-8 (IFILL_$PYRAMIDS=1)
for I=1,ISCM_MAX_$PYRAMIDS

5 and 6 are required only if IFILL_$PYRAMIDS=1

5  ECUT, PCUT, DOSE_ZONE, IREGION_TO_BIT (2F15.0,2I5):
      ECUT, PCUT:   Cutoff energies for electrons and photons
                    in central region and outer region, layer I
       DOSE_ZONE:   Dose scoring zone for central region and outer region
  IREGION_TO_BIT:   mapping of central region and outer region to bit
                    for LATCH

6  MED_IN (24A1):   Medium of central region and outer region in layer I,
                    used to set MED_INDEX.

7   ECUT, PCUT, DOSE_ZONE, IREGION_TO_BIT (2F15.0,2I5):
      ECUT, PCUT:   Cutoff energies for electrons and photons
                    in surrounding region, layer I
       DOSE_ZONE:   Dose scoring zone of surrounding region.
  IREGION_TO_BIT:   mapping of surrounding region to bit for LATCH

8  MED_IN (24A1):   Medium of surrounding region in layer I,
                    used to set MED_INDEX.

 Example
 *******
The following input file describes two pyramidal openings,
both 0.3cm thick, and both filled with air (ie IFILL_$PYRAMIDS=0).
The first opening begins at Z=0.0 cm and is cut out of Pb.  Its positive
X face begins at XFP=0.8cm and angles back to XBP=1.2cm.  The negative X
face is a mirror image, beginning at XFN=-0.8cm and going to XBN=-1.2cm.
The outer x edge is at XMAX=5.0cm. The Y faces of the first opening are
perpendicular to the beam, with YFP=YBP=0.8cm and YFN=YBN=-0.8cm. The
outer y edge is at YMAX=4.6cm.
The second opening begins at Z=0.31 and is cut out of aluminum.  Its
positive and negative X faces describe a parallelepiped, with XFP=1.2cm,
XBP=0.8cm, XFN=-0.8cm, XBN=-1.2cm.
The positive Y face angles from YFP=2.0cm to
YBP=0.5cm and the negative Y face is a mirror image of this, going from
XFN=-2.0cm to XBN=-0.5cm. The outer x edge and y edge are at XMAX=5.1cm,
YMAX=5.2cm respectively. Note that an air gap >= AIRGAPMIN (=0.01cm) must
be left between pyramids and the top of the CM and the first pyramid.
```

```
In this particular input file, there is no gap at the top of the
CM, so Z_min_CM will be automatically reset to -0.01cm to
provide the required gap.

Dose in the air gaps and openings will be scored in dose zone 1.
Dose in the Pb will be scored in zone 2, and dose in the Al will be
scored in zone 3.  ECUT and PCUT in all regions are set to 0.521MeV and
0.01 MeV respectively.

10.0000,
PYR
2,0
0.0, 0.3, 0.8, 1.2, -0.8, -1.2, 0.8, 0.8, -0.8, -0.8, 5.0, 4.6
0.31, 0.61, 1.2, 0.8, -0.8, -1.2, 2.0, 0.5, -2.0, -0.5, 5.1, 5.2
0.521, 0.01, 1, 0
0.521, 0.01, 2, 0
PB521ICRU
0.521, 0.01, 3, 0
AL521ICRU
```

15.3.13 BLOCK

The BLOCK CM is used to model a treatment block having non-rectangular and/or multiple openings. The user specifies openings in the block material using up to 20 "subregions". For each subregion the user specifies the X-Y coordinates of its vertices at the top surface of the block material (either clockwise or counter-clockwise around the perimeter). The inner planes of all subregions are angled with respect to the beam (Z) axis towards a single user-specified "focus point" on the beam axis. Note that no subregion can have an inner angle > 180 degrees. Openings can consist of a single subregion or may require several adjoining subregions in order to avoid the restriction on the inner angles. The user also specifies the X and Y coordinates of the 4 outer edges of the block material, so the material need not extend to the square outer boundary of this CM. Due to its generality, BLOCK may require up to 2 times the CPU time of PYRAMIDS to simulate simple rectangular geometries; thus, PYRAMIDS is recommended when there is a single rectangular opening.

The current version of BLOCK has been significantly changed from BLOCK in BEAM99 (see `CHANGES_from_BEAM99.for.BEAM00`). This is because John Antolak of the M.D. Anderson Cancer Center found and corrected some serious errors in this CM.

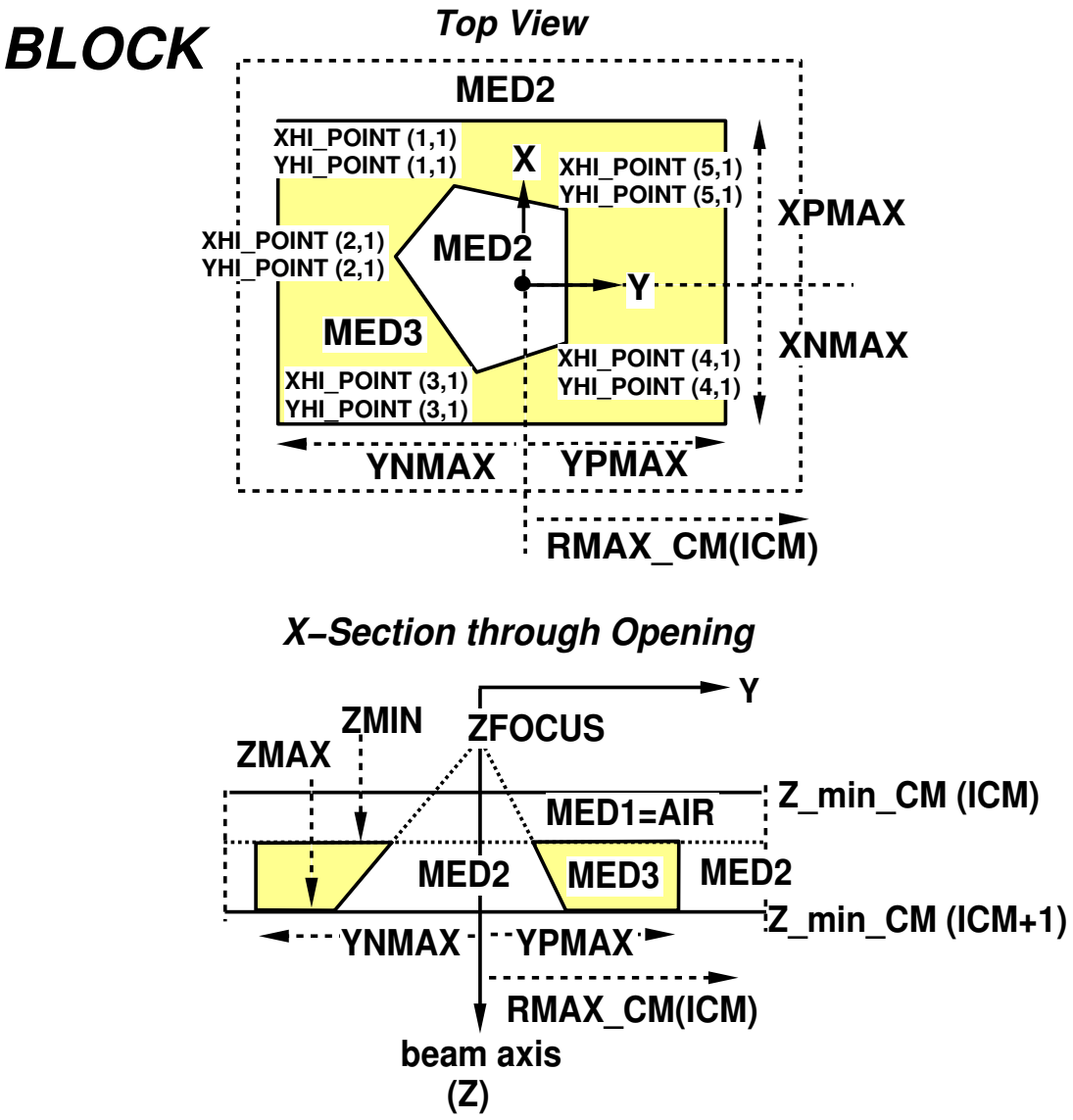


Figure 32: A BLOCK component module with a single opening composed of 1 subregion (ISUB_MAX=1). As shown in the top view, the subregion has 5 vertices (NSUB(1)=5). The X-Y coordinates of the vertices at the top surface of the block material are specified counter-clockwise around the perimeter of the subregion, beginning with (XHI_POINT(1,1),YHI_POINT(1,1)). Also shown in the top view are the coordinates defining the outer boundary of the block, XPMAX, YPMAX, XNMAX and YNMAX. The cross-section through the opening shows that the planes defining the inner surfaces of the subregion/opening are all angled with respect to the Z-axis towards the single focus point on the Z-axis. Note that there must be an air gap between ZMIN and Z_min_CM(ICM) of ≥ 0.01 cm. Note that the user specifies MED2, the material filling the opening(s) and the region beyond the block edges. The top gap is assumed to be air. More complicated opening shapes may require several adjoining subregions.

The input format for BLOCK and an example input are given below.

```
CARDS CM_BLOCK(Rev 1.4)
*****************

-1  dummy line filled with ***  read in main

 0  RMAX_CM(ICM_$BLOCK):  Perpendicular distance from Z-axis
                          to boundary surrounding component
                          This component module has a square
                          boundary.

 1  TITLE_$BLOCK (60A1):  Title of CM.

 2  ZMIN_$BLOCK, ZMAX_$BLOCK, ZFOCUS_$BLOCK (3F15.0):

      ZMIN_$BLOCK:  Z of front of CM (not including airgap) (cm).
      ZMAX_$BLOCK:  Z of back of CM (cm).
    ZFOCUS_$BLOCK:  Z at which the inner sides of the opening(s)
                    in the block are focused (cm).

       Note restrictions: ZMAX < ZFOCUS or ZFOCUS < ZMIN, ie not in between
                          ZMIN - ZFOCUS >= 0.01 if ZFOCUS < ZMIN

 3  ISUB_MAX_$BLOCK (I5): Number of subregions.  Each opening is made
                          up of one or more subregions.

 Repeat 4 - 4a for J = 1, ISUB_MAX_$BLOCK

 4  NSUB_$BLOCK(J)   (I5)

      NSUB_$BLOCK(J):  number of points defining subregion J

 Repeat 4a for I = 1, NSUB_$BLOCK(J)

4a  XHI_POINT_$BLOCK(I,J),YHI_POINT_$BLOCK(I,J) (2F15.0):

    XHI_POINT_$BLOCK(I,J): X coordinate of point I at upper surface (cm)
    YHI_POINT_$BLOCK(I,J): Y coordinate of point I at upper surface (cm)

    NOTE:  Input the points clockwise or counter-clockwise around
           the perimeter of each subregion.  A subregion may not
           have an interior angle > 180 degrees.

 5  XPMAX_$BLOCK,YPMAX_$BLOCK,XNMAX_$BLOCK,YNMAX_$BLOCK (4F15.0):

      XPMAX_$BLOCK: X coordinate of block edge in +X direction (cm)
      YPMAX_$BLOCK: Y coordinate of block edge in +Y direction (cm)
      XNMAX_$BLOCK: X coordinate of block edge in -X direction (cm)
```

```
      YNMAX_$BLOCK: Y coordinate of block edge in -Y direction (cm)

 6   ECUT, PCUT, DOSE_ZONE, IREGION_TO_BIT (2F15.0,2I5) for air
                        in gap at top.

       ECUT, PCUT:   Cutoff energies for electrons and photons.
        DOSE_ZONE:   Dose scoring zone of air.
   IREGION_TO_BIT:   mapping of region to bit for LATCH

 7   ECUT, PCUT, DOSE_ZONE, IREGION_TO_BIT (2F15.0,2I5) in openings
                             and beyond edges of the block

       ECUT, PCUT:   Cutoff energies for electrons and photons in
                     openings and beyond edges of block
        DOSE_ZONE:   Dose scoring zone of openings and region beyond
                     edges of block
   IREGION_TO_BIT:   mapping of region comprising openings and region
                     beyond block edges to bit for LATCH

 8   MED_IN (24A1):  Medium in openings and beyond block edges,
                     used to set MED_INDEX.

 9   ECUT, PCUT, DOSE_ZONE, IREGION_TO_BIT (2F15.0,2I5) in block material

       ECUT, PCUT:   Cutoff energies for electrons and photons
                     in material surrounding openings
        DOSE_ZONE:   Dose scoring zone of material surrounding
                     openings.
   IREGION_TO_BIT:   mapping of region surrounding openings
                     to bit for LATCH

 10  MED_IN (24A1):  Medium of block,
                     used to set MED_INDEX.
 Example
 *******
The following input file describes a BLOCK of 4cm thick.
The block begins at Z=0.0 cm and is made of MILDSTEEL. The air-filled
opening(s) focus at Z=-10cm. Its positive X boundary begins at XPMAX=4.2cm,
positive Y boundary at YPMAX=4.8cm, and its negative X boundary at
XNMAX=-5.0cm, negative Y boundary at YNMAX=-3.0cm.
There are 2 sub-regions describing one opening shaped like an arrow.
In the first sub-region, there are 5 defining points; in the second one,
there are 4 defining points. The defining points should be input
clockwisely or counter-clockwisely. Each point is input as x, y in
a line.
In this particular input file, there is no gap at the top of the
CM, so Z_min_CM will be automatically reset to -0.01cm to
provide the required gap.
```

```
Dose in the air regions will be scored in dose zone 1.
Dose in the block material will be scored in zone 2.
ECUT and PCUT in all regions are set to 0.521MeV and
0.01 MeV respectively.

*********************************************
10.0,           RMAX
arrow shaped cutoff
0.0, 4.0, -10.0,     ZMIN, ZMAX, ZFOCUS
2,             2 sub-regions
5,       5 defining points in sub 1
0., 3.,        x,y of point 1 in sub 1
-2., 1.,       x,y of point 2 in sub 1
-1., 0.,       x,y of point 3 in sub 1
1., 0.,        x,y of point 4 in sub 1
2., 1.,        x,y of point 5 in sub 1      end of sub 1
4,       4 defining points in sub 2
-1., 0.,       x,y of point 1 in sub 2
1., 0.,        x,y of point 2 in sub 2
1., -2.,       x,y of point 3 in sub 2
-1., -2.,      x,y of point 4 in sub 2.    end of sub 2.
4.2, 4.8, -5.0, -3.0,      xpmax,ypmax,xnmax,ynmax
0.0, 0.0, 1, 0,     ecut, pcut, dose-zone, ir-to-bit for air
0.0, 0.0, 1, 0,     ecut, pcut, dose-zone, ir-to-bit for openings
AIR521ICRU
0.0, 0.0, 2, 0,     ecut, pcut, dose-zone, ir-to-bit for materail
MILDSTEEL521
*********************************************
```

15.3.14 MLC

The MLC CM is used to model a double-focusing multi-leaf collimator with flat faces. The collimator has a single layer with a user-specified number of leaves all opening in either the X or Y direction. The collimator opening is specified by the coordinates of the individual leaf openings at the top of the collimator, the thickness of the leaves in the Z direction, and two Z "foci" that determine the angles of the leaf side and end surfaces. The outer boundary of the MLC CM is a square centred on the beam axis. Currently, the collimator body extends to this outer boundary.

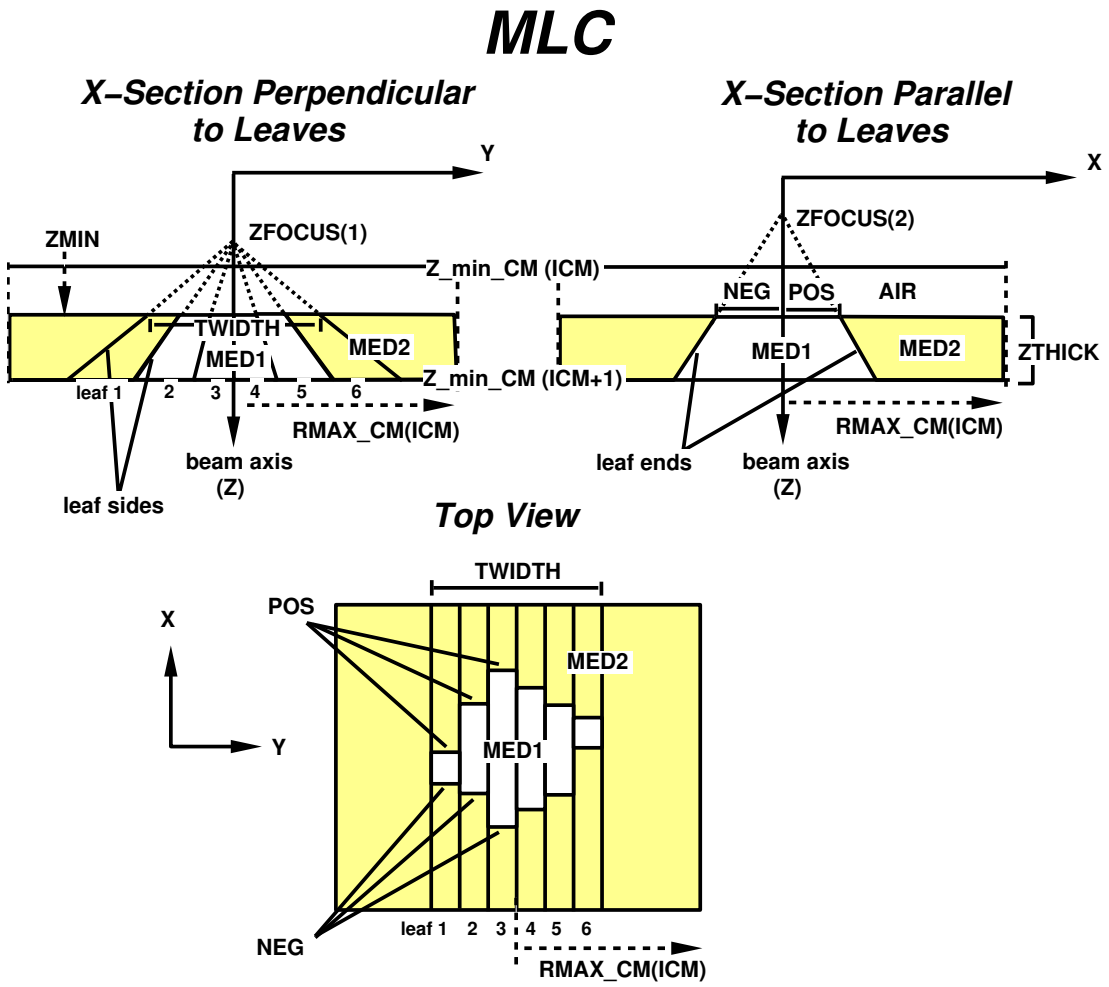


Figure 33: An MLC component module with 6 leaves (NUM_LEAF=6) opening in the X direction (IDMLFC=1). The cross-section perpendicular to the leaves (in the Y direction) shows the total width of the leaves at the top of the collimator, TWIDTH. The width of each leaf at the top is TWIDTH/NUM_LEAF. Note that NUM_LEAF must be an even number so that there is an equal number of leaves on either side of the X axis. The Y coordinates of the leaves at the bottom of the collimator are calculated automatically so that the tangents to the side surfaces of the leaves all cross the Z-axis at Z=ZFOCUS(1). The cross-section parallel to the leaves (in the X direction) shows the negative, NEG, and positive, POS, coordinates of the opening in one leaf. NEG and POS coordinates are input for every leaf. The X coordinates of the leaf openings at the bottom of the collimator are calculated automatically so that the tangents to the leaf ends all cross the Z-axis at Z=ZFOCUS(2). The top view of the collimator shows how the collimator opening is composed of the individual leaf openings. MED1 defines the material in the collimator opening and MED2 defines the material of the collimator leaves and body. This example would be applicable to leaves opening in the Y direction (IDMLFC=0) by reversing the roles of X and Y. Although this particular example shows ZFOCUS points that are < ZMIN, the ZFOCUS points can also be > ZMIN+ZTHICK, in which case leaf surfaces angle in towards the Z-axis with increasing Z.

The input format for MLC and an example input are given below.

```
CARDS CM_$MLC
**************
-1  Dummy line to indicate start of CM

 0  RMAX_CM(ICM_$MLC) (F10.0):  Half-width of CM boundary (cm).

 1  TITLE_$MLC (60A1):  Title of CM.

 2  IDMLFC_$MLC (I5) = 0 for leaves parallel to Y direction
                     = 1 for leaves parallel to X direction

 3  ZMIN_$MLC (F15.0): Z of top of collimator (excluding airgap)

 4  ZTHICK_$MLC (F15.0): Thickness of the leaves (cm)

 5  NUM_LEAF_$MLC, TWIDTH_$MLC (I5,F15.0)

         NUM_LEAF_$MLC: Number of leaves
         TWIDTH_$MLC:   Total width of leaves in X (IDMLFC_$MLC=0)
                        or Y (IDMLFC_$MLC=1) direction (cm)

      Note: width of each leaf = TWIDTH_$MLC/NUM_LEAF_$MLC

 6  ZFOCUS_$MLC(1) (F15.0): Focal point on Z-axis of leaf sides (ie.
                            imaginary lines drawn extending the slopes of
                            the leaf sides will all intersect the Z-axis
                            at this point)

      Note restriction: ZFOCUS_$MLC(1) < ZMIN_$MLC or
                                       > ZMIN_$MLC + ZTHICK_$MLC

 7  ZFOCUS_$MLC(2) (F15.0): Focal point on Z-axis of leaf ends (ie.
                            imaginary lines drawn extending the slopes of
                            the leaf ends will all intersect the Z-axis
                            at this point)

      Note restriction: ZFOCUS_$MLC(1) < ZMIN_$MLC or
                                       > ZMIN_$MLC + ZTHICK_$MLC

    Repeat 8 until coordinates of all leaves are defined once.  Leaves
    are numbered 1,2,...NUM_LEAF_$MLC, where numbering goes from left
    to right in the X-Y plane if IDMLFC_$MLC=0 and from top to bottom
    in the X-Y plane if IDMLFC_$MLC=1.

 8  NEG_$MLC, POS_$MLC, NUM_$MLC (2F15.0,I5)

      NEG_$MLC:    Min. Y (IDMLFC_$MLC=0) or X (IDMLFC_$MLC=1)
```

```
                of front opening in leaf I (ie the opening at
                ZMIN_$MLC)
    POS_$MLC:   Max. Y (IDMLFC_$MLC=0) or X (IDMLFC_$MLC=1)
                of front opening in leaf I
    NUM_$MLC:   Apply NEG_$MLC(I) and POS_$MLC(I) to leaves
                I,...,I+NUM_$MLC-1.  Defaults to 1 if set <= 0.
                Defaults to NUM_LEAF_$MLC-I+1 if set >
                NUM_LEAF_$MLC-I+1

 9  ECUT, PCUT, DOSE_ZONE, IREGION_TO_BIT in local region 1
                                      (inside collimator) (2F15.0,I5):

     ECUT, PCUT:  Cutoff energies for electrons and photons.
     DOSE_ZONE:   Dose scoring flag, 0 to not score dose
 IREGION_TO_BIT:  Bit number associated with this region

10  MED_IN (24A1):  Medium of in local region 1 (inside collimator)
                    used to set MED_INDEX.

11  ECUT, PCUT, DOSE_ZONE, IREGION_TO_BIT in local region 2
                                      (collimator leaves) (2F15.0,I5):

     ECUT, PCUT:  Cutoff energies for electrons and photons.
     DOSE_ZONE:   Dose scoring flag, 0 to note score dose
 IREGION_TO_BIT:  Bit number associated with this region

12  MED_IN (24A1):  Medium of local region 2 (collimator leaves),
                    used to set MED_INDEX.

Example
*******

The following example defines a multi-leaf collimator design based
loosely on that used with the MM50 Racetrack Microtron accelerator.
The collimator starts at Z=65 cm and has 64 tungsten leaves opening
in the X direction.  The leaves are each 0.8125cm wide and 7.5cm
thick.  The Z focus of the leaf sides is at Z=-1000 cm, resulting
in sides that are essentially straight up and down.  The Z focus
of the leaf ends is at Z=0, which is the position of the beam source.
In this example, leaf opening coordinates are chosen to create a
irregular off-center collimator opening.

Electrons and photons in both the collimator and the opening regions
will be followed down to kinetic energies of 10 keV (ECUT=0.521,
PCUT=0.01).  Dose deposited in the tungsten leaves will be stored
in dose zone 2, and dose deposited in the opening will be stored
in dose zone 1.

26.0,           RMAX_CM
```

```
Collimator based on MLC for MM50 accelerator
1,              Leaves open in X direction
65.0,           ZMIN
7.5,            ZTHICK
64, 52.0,       NUM_LEAF, TWIDTH
-1000.0,        ZFOCUS(1)
0.0             ZFOCUS(2)
0.0,0.0,15,     15 closed leaves
0.0,2.0,5,      5 leaves with opening 0.0 - 2.0
0.5,3.0,2       2 leaves with 0.5 - 3.0
1.0,4.0,3       3 leaves with 1.0 - 4.0
2.0,7.0,10,     10 leaves with opening 2.0 - 7.0
1.5,6.0
1.0,6.0
0.0,5.0,3,      3 leaves with 0.0 - 5.0
-1.0,4.0,5,     5 leaves with -1.0 - 4.0
-2.0,4.0,3,     3 leaves with -2.0 - 4.0
-4.0,4.0,5,     5 leaves with -4.0 - 4.0
-5.0,3.0
-6.0,1.0
-8.0,0.0
-10.0,-2.0,3,   3 leaves with -10.0 - -2.0
-12.0,-2.0,2,   2 leaves with -12.0 - -2.0
-15.0,-3.0,2,   2 leaves with -15.0 - -3.0
-15.0,-15.0
0.5210,  0.010,  1,  0
AIR700ICRU
0.5210,  0.010,  2,  0
W700ICRU
```

15.3.15 MLCQ

The MLCQ CM is used to model a focusing multi-leaf collimator with rounded leaf ends. The collimator is similar to MLC with the exception that, rather than specifying a Z focus for the leaf ends, the user specifies a radius for the leaf ends and the Z position of the origin of this radius (*i.e.* so the leaf ends can be angled up or down). The collimator opening is defined by specifying the X or Y (depending on leaf orientation) origin of the radius for the positive and negative portions of each leaf.

The first version of this CM, which was based on MLC, was coded by Hugo Palmans and Kristiaan De Vlamynck of the University of Gent, Belgium.

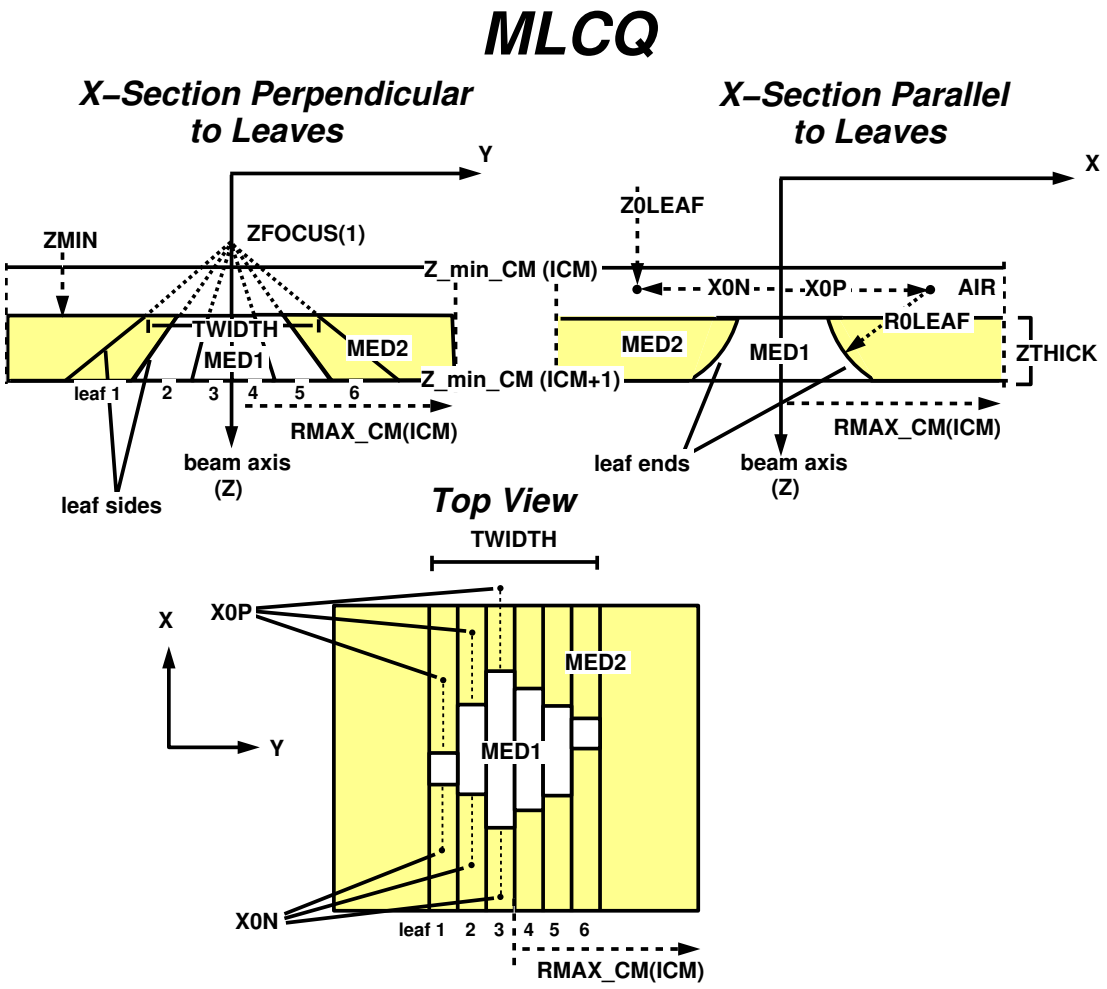


Figure 34: An MLCQ component module with 6 leaves (NUM_LEAF=6) opening in the X direction (IDMLFC=1). Parameters are similar to MLC with the exception of those for the leaf ends. The cross-section parallel to the leaves shows how the rounded leaf ends are defined. Basically, for each leaf, the user defines a circle with radius R0LEAF (the radius of the leaf ends), centred at (X0N,Z0LEAF) for the negative portion of the leaf and (X0P,Z0LEAF) for the positive portion of the leaf. Note that X0N,X0P define X coordinates if the leaves are parallel to the X direction and Y coordinates if the leaves are parallel to the Y direction. Also note that Z0LEAF applies to all leaves. The rounded leaf end is then the arc subtended by the intersection of this circle with ZMIN and ZMIN + ZTHICK. The user controls the angle of the leaf ends with respect to the Z axis using Z0LEAF and the dimensions of the leaf opening using X0N and X0P. The top view shows that ABS(X0N) and ABS(X0P) can be > RMAX_CM, however the points of intersection of the circle defining a leaf end with ZMIN and ZMIN + ZTHICK must be within the boundaries of the CM or else the volume/mass of the collimator will be determined incorrectly for dose calculations.

The input format for MLCQ and an example input are given below.

```
CARDS CM_$MLCQ
**************
-1  Dummy line to indicate start of CM

 0  RMAX_CM(ICM_$MLCQ) (F10.0):  Half-width of CM boundary (cm).

 1  TITLE_$MLCQ (60A1):  Title of CM.

 2  IDMLFC_$MLCQ (I5) = 0 for leaves parallel to Y direction
                      = 1 for leaves parallel to X direction

 3  ZMIN_$MLCQ (F15.0): Z of top of collimator (excluding airgap)

 4  ZTHICK_$MLCQ (F15.0): Thickness of the leaves (cm)

 5  NUM_LEAF_$MLCQ, TWIDTH_$MLCQ (I5,F15.0)

         NUM_LEAF_$MLCQ: Number of leaves
         TWIDTH_$MLCQ:   Total width of leaves in X (IDMLFC_$MLCQ=0)
                       or Y (IDMLFC_$MLCQ=1) direction (cm)

      Note: width of each leaf = TWIDTH_$MLCQ/NUM_LEAF_$MLCQ

 6  ZFOCUS_$MLCQ(1) (F15.0): Focal point on Z-axis of leaf sides (ie.
                             imaginary lines drawn extending the slopes of
                             the leaf sides will all intersect the Z-axis
                             at this point)

      Note restriction: ZFOCUS_$MLCQ(1) < ZMIN_$MLCQ or
                                        > ZMIN_$MLCQ + ZTHICK_$MLCQ

 7  ROLEAF_$MLCQ,ZOLEAF_$MLCQ (2F15.0)

         ROLEAF_$MLCQ: Radius of leaf ends in cm.
         ZOLEAF_$MLCQ: Z where radius of leaf ends originates in cm.

    Note restrictions: 1. ZMIN_$MLCQ < ZOLEAF_$MLCQ <
                                        ZMIN_$MLCQ + ZTHICK_$MLCQ
                       2. ROLEAF_$MLCQ >
                            MAX(ZMIN_$MLCQ+ZTHICK_$MLCQ-ZOLEAF_$MLCQ,
                                ZOLEAF_$MLCQ-ZMIN_$MLCQ)
    Repeat 8 until coordinates of all leaves are defined once.  Leaves
    are numbered 1,2,...NUM_LEAF_$MLCQ, where numbering goes from left
    to right in the X-Y plane if IDMLFC_$MLCQ=0 and from top to bottom
    in the X-Y plane if IDMLFC_$MLCQ=1.

 8  XON_$MLCQ, XOP_$MLCQ, NUM_$MLCQ (2F15.0,I5)
```

```
     XON_$MLCQ:   Y (IDMLFC_$MLCQ=0) or X (IDMLFC_$MLCQ=1)
                  of origin of radius of negative part of leaf I
     XOP_$MLCQ:   Y (IDMLFC_$MLCQ=0) or X (IDMLFC_$MLCQ=1)
                  of origin of radius of positive part of leaf I
     NUM_$MLCQ:   Apply XON_$MLCQ and XOP_$MLCQ to leaves
                  I,...,I+NUM_$MLCQ-1.  Defaults to 1 if set <= 0.
                  Defaults to NUM_LEAF_$MLCQ-I+1 if set >
                  NUM_LEAF_$MLCQ-I+1

 9  ECUT, PCUT, DOSE_ZONE, IREGION_TO_BIT in local region 1
                                       (inside collimator) (2F15.0,I5):

      ECUT, PCUT:  Cutoff energies for electrons and photons.
      DOSE_ZONE:   Dose scoring flag, 0 to not score dose
 IREGION_TO_BIT:   Bit number associated with this region

10  MED_IN (24A1):  Medium of in local region 1 (inside collimator)
                    used to set MED_INDEX.

11  ECUT, PCUT, DOSE_ZONE, IREGION_TO_BIT in local region 2
                                       (collimator leaves) (2F15.0,I5):

      ECUT, PCUT:  Cutoff energies for electrons and photons.
      DOSE_ZONE:   Dose scoring flag, 0 to note score dose
 IREGION_TO_BIT:   Bit number associated with this region

12  MED_IN (24A1):  Medium of local region 2 (collimator leaves),
                    used to set MED_INDEX.

Example
*******

The following example defines a multi-leaf collimator
that starts at Z=65 cm and has 64 tungsten leaves opening
in the X direction.  The leaves are each 0.8125cm wide and 7.5cm
thick.  The Z focus of the leaf sides is at Z=-1000 cm, resulting
in sides that are essentially straight up and down.  The radius
of the leaf ends is 10cm and has a Z origin of 65cm (ie ZMIN),
The unique position of the Z origin of the radius means that the X
dimensions of the opening in each leaf at the top of the collimator
(ZMIN) will be given by XON+10 - XOP-10 cm.
In this example, the X origins of the radii of the individual leaves
are chosen to create an irregular off-center collimator opening.

Electrons and photons in both the collimator and the opening regions
will be followed down to kinetic energies of 10 keV (ECUT=0.521,
PCUT=0.01).  Dose deposited in the tungsten leaves will be stored
in dose zone 2, and dose deposited in the opening will be stored
```

```
in dose zone 1.

26.0,           RMAX_CM
Collimator based on MLC for MM50 accelerator
1,              Leaves open in X direction
65.0,           ZMIN
7.5,            ZTHICK
64, 52.0,       NUM_LEAF, TWIDTH
-1000.0,        ZFOCUS(1)
10.0,65.0       ROLEAF,ZOLEAF
-10.0,10.0,15,  15 leaves closed at top
-10.0,12.0,5,   5 leaves with opening 0.0 - 2.0 at top
-9.5,13.0,2     2 leaves with 0.5 - 3.0 at top
-9.0,14.0,3     3 leaves with 1.0 - 4.0 at top
-8.0,17.0,10,   10 leaves with opening 2.0 - 7.0 at top
-8.5,16.0
-9.0,16.0
-10.0,15.0,3,   3 leaves with 0.0 - 5.0 at top
-11.0,14.0,5,   5 leaves with -1.0 - 4.0 at top
-12.0,14.0,3,   3 leaves with -2.0 - 4.0 at top
-14.0,14.0,5,   5 leaves with -4.0 - 4.0 at top
-15.0,13.0
-16.0,11.0
-18.0,10.0
-20.0,8.0,3,    3 leaves with -10.0 - -2.0 at top
-22.0,8.0,2,    2 leaves with -12.0 - -2.0 at top
-25.0,7.0,2,    2 leaves with -15.0 - -3.0 at top
-25.0,-5.0
0.5210,  0.010,  1,  0
AIR700ICRU
0.5210,  0.010,  2,  0
W700ICRU
```

15.3.16 VARMLC

The VARMLC CM is used to model a focusing multi-leaf collimator with either rounded leaf ends or straight leaf ends with a Z focus. The major difference between VARMLC and MLC or MLCQ is that VARMLC simulates the air gaps between leaves, the tongue-in-groove mechanism by which adjacent leaves slide against each other, and the driving screws at the top and bottom of each leaf used to open and close the leaves. VARMLC can also simulate leaves of different widths within the same collimator. In VARMLC the medium beyond the leaves in the direction perpendicular to the leaves defaults to be the same as the medium in the leaf opening(s) (ie AIR). This is different from MLC and MLCQ in which the medium in this region defaults to the leaf medium (ie solid). This default can be changed within the `VARMLC_cm.mortran` and `VARMLC_macros.mortran` codes.

The first version of this CM was coded by Ajay Kapur with Charlie Ma at Stanford University. The current version has significant modifications.

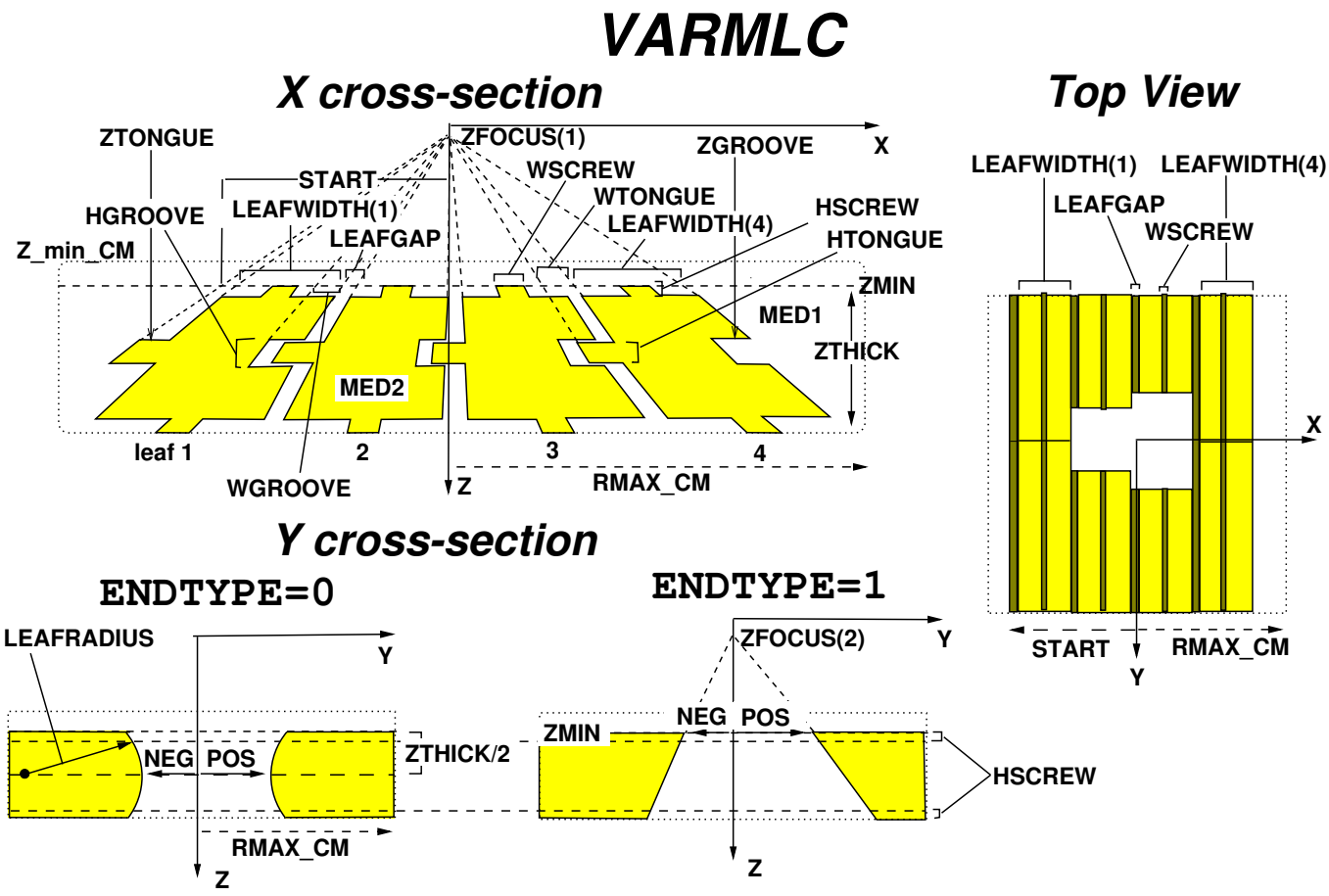


Figure 35: A VARMLC component module with 4 leaves (NUM_LEAF=4) opening in the Y direction (ORIENT=0). The X cross-section (perpendicular to the leaf direction) shows the widths of the leaves (LEAFWIDTH(1)...LEAFWIDTH(4)) and the dimensions of the leaf gap (LEAFGAP), tongue-in-groove mechanism (WGROOVE, HGROOVE, WTONGUE, HTONGUE) and the driving screws (WSCREW, HSCREW) as input by the user. The user also sets the position of the top of the tongue, ZTONGUE, and the top of the groove, ZGROOVE, with the restriction that the tongue and groove cannot overlap. If ZTONGUE=ZGROOVE=0 then the tongue and groove are centred at Z = ZMIN + ZTHICK/2. For restrictions on leaf gap and tongue-in-groove inputs, see the detailed description of input parameters below. Note that all leaf side surfaces are focused at ZFOCUS(1) and that horizontal dimensions (WGROOVE, WTONGUE, WSCREW, START) are all given at ZMIN. Due to LEAFGAP and varying LEAFWIDTH the leaves may not be symmetric about the Z axis, although START can be set to give approximate symmetry. Also note that the medium in regions beyond the leaves perpendicular to the leaf direction defaults to MED1, the medium of the openings and air gaps. Y cross-sections are shown for leaves with rounded ends (ENDTYPE=0) and straight ends (ENDTYPE=1). For rounded ends, the user inputs the radius of the ends (LEAFRADIUS) and, for each leaf, the minimum and maximum coordinates of the leaf opening at Z = ZMIN + ZTHICK/2 (NEG, POS). The end radius is always assumed to originate at Z = ZMIN + ZTHICK/2. For straight leaf ends, the user inputs the Z focus of the ends (ZFOCUS(2)) and, for each leaf, the minimum and maximum coordinates of the opening at Z = ZMIN (NEG, POS).

The input format for VARMLC and an example input are given below.

```
CARDS CM_$VARMLC
**************
-1  Dummy line to indicate start of CM

 0  RMAX_CM(ICM_$VARMLC) (F10.0):  Half-width of CM boundary (cm).

 1  TITLE_$VARMLC (60A1):  Title of CM.

 2  ORIENT_$VARMLC, NGROUP_$VARMLC (2I5)

          ORIENT_$VARMLC= 0 for leaves parallel to Y direction
                        = 1 for leaves parallel to X direction
          NGROUP_$VARMLC= number of groups of adjacent leaves where
                          all leaves in a group have the same width
                          (defaults to 1 if set <= 0)

 3  ZMIN_$VARMLC (F15.0): Z of top of MLC (excluding airgap)

 4  ZTHICK_$VARMLC (F15.0): Thickness of the leaves ( z-axis (cm))

 Repeat 5 NGROUP_$VARMLC times

 5  NUM_LEAF_$VARMLC(I), LEAFWIDTH_$VARMLC(I)  (I5,F15.0)

         NUM_LEAF_$VARMLC(I):  Number of adjacent leaves in group I
         LEAFWIDTH_$VARMLC(I): Width of each leaf in group I
                        as projected at ZMIN_$VARMLC - the width does
                        not include the tongue (see figure).

       Note: total number of leaves is stored in TOT_LEAF_$VARMLC

 6  START_$VARMLC (F15.0) : the start position (cm) wrt the CAX of
                     leaf 1 tongue as projected to ZMIN_$VARMLC.

 7  WSCREW_$VARMLC, HSCREW_$VARMLC (2F15.0) : The width and height
          of the screw on the carriage railing. The width is
          as projected at ZMIN_$VARMLC and the height projected at
          the z-axis.

 8  WTONGUE_$VARMLC, HTONGUE_$VARMLC, ZTONGUE_$VARMLC (3F15.0) :
          The width and height of the tongue projected to ZMIN_$VARMLC
          and z-axis respectively and the Z starting position of the
          tongue.  ZTONGUE_$VARMLC=0 assumes that the tongue is centred
          at ZMIN_$VARMLC+ZTHICK_$VARMLC/2 (ie centre of leaf body).

 9  WGROOVE_$VARMLC, HGROOVE_$VARMLC, ZGROOVE_$VARMLC (3F15.0) : The
          width and height of the groove projected to ZMIN_$VARMLC
```

```
          and z-axis respectively and the Z starting position of the
          groove.  ZGROOVE_$VARMLC=0 assumes that the groove is centred
          at ZMIN_$VARMLC+ZTHICK_$VARMLC/2 (ie centre of leaf body).

          Note restriction: ZTONGUE >= ZGROOVE
                            ZTONGUE+HTONGUE <= ZGROOVE+HGROOVE
                            WTONGUE <= WGROOVE

10  LEAFGAP_$VARMLC (F15.0) : The width of the interleaf air gap
           at ZMIN_$VARMLC.

          Note restriction: LEAFGAP_$VARMLC <=WTONGUE_$VARMLC

11  ENDTYPE_$VARMLC (I5) : The  type of leaf end :
           0 -- rounded leaf end and
           1 -- focused divergent leaf end.

12  ZFOCUS_$VARMLC (F15.0) : Focal point on Z-axis of leaf ends
                  (i.e. imaginary lines drawn extending the slopes
                  of leaf ends will all intersect the Z-axis
                  at this point) - chosen if ENDTYPE_$VARMLC = 1.

      Note restriction: ZFOCUS_$VARMLC(1) < ZMIN_$VARMLC or
                                    > ZMIN_$VARMLC + ZTHICK_$VARMLC

    LEAFRADIUS_$VARMLC (F15.0) : Radius of the leaf end if
                         ENDTYPE_$VARMLC = 0. This must be greater
                         than or equal to half the leaf thickness.

13  ZFOCUS_$VARMLC(1) (F15.0): Focal point on Z-axis of leaf sides
                     imaginary lines drawn extending the slopes of
                     the leaf sides will all intersect the Z-axis
                     at this point)

      Note restriction: ZFOCUS_$VARMLC(1) < ZMIN_$VARMLC or
                             > ZMIN_$VARMLC + ZTHICK_$VARMLC

                   For focused ends the leaf position is defined
                   at ZMIN_$VARMLC; for rounded at ZMIN_$VARMLC +
                  0.5*ZTHICK_$VARMLC (ie center of the leaf in z)

 Repeat 14 until coordinates of all leaves are defined once.  Leaves
 are numbered 1,2,...TOT_LEAF_$VARMLC, where numbering goes from leaf
 1 to leaf TOT_LEAF_$VARMLC. Convention is lower to upper or
 left to right depending on ORIENT_$VARMLC i.e from negative to
 positive.

14  NEG_$VARMLC, POS_$VARMLC, NUM_$VARMLC (2F15.0,I5)
```

```
 NEG_$VARMLC:   Min. Y (ORIENT_$VARMLC=0) or X (ORIENT_$VARMLC=1)
                of front opening in leaf I (ie the opening at
                ZMIN_$VARMLC) if ENDTYPE=1, or of rounded end
                of leaf I if ENDTYPE=0.
 POS_$VARMLC:   Max. Y (ORIENT_$VARMLC=0) or X (ORIENT_$VARMLC=1)
                of front opening in leaf I if ENDTYPE=1, or of
                rounded end of leaf I if ENDTYPE=0.
 NUM_$VARMLC:   Apply NEG_$VARMLC and POS_$VARMLC to leaves
                I,...,I+NUM_$VARMLC-1.  Defaults to 1 if set <=0.
                Defaults to TOT_LEAF_$VARMLC-I+1 if set >
                TOT_LEAF_$VARMLC-I+1.

15   ECUT, PCUT, DOSE_ZONE, IREGION_TO_BIT in opening(s) and
                                          air gaps (2F15.0,2I5)

      ECUT, PCUT:  Cutoff energies for electrons and photons.
      DOSE_ZONE:   Dose scoring flag, 0 to not score dose
 IREGION_TO_BIT:   Bit number associated with this region

16  MED_IN (24A1):  Medium in opening(s) and air gaps
                    used to set MED_INDEX.

17  ECUT, PCUT, DOSE_ZONE, IREGION_TO_BIT in leaves, IGNOREGAPS_$VARMLC
                                                      (2F15.0,3I5):

      ECUT, PCUT:  Cutoff energies for electrons and photons.
      DOSE_ZONE:   Dose scoring flag, 0 to note score dose
 IREGION_TO_BIT:   Bit number associated with this region
     IGNOREGAPS:   If set to 1, ignore all air gaps when doing range
                   rejection in leaf material when the particle X
                   position
                   is < min X of all leaf openings (not including leaf
                   ends) or > max X of leaf openings (not including ends)
                   (ORIENT_$VARMLC=1) or if the particle Y position
                   is < min Y of all leaf openings (not including leaf
                   ends) or > max Y of leaf openings (not including ends)
                   (ORIENT_$VARMLC=0). This approximation is designed
                   to make range rejection more efficient deep in the
                   leaves, while still preserving accurate transport
                   in the leaf ends.  Note that if you have significant
                   air gaps between leaves, it is recommended that
                   you not use this option (ie run with the default
                   setting of 0).

18  MED_IN (24A1):  Medium of leaves,
                    used to set MED_INDEX.

Example
*******
```

```
The following example defines a multi-leaf collimator design based
loosely on that used with the Varian Clinac 2100C 26 leaf pair.
Actual parameters are DIFFERENT - this serves just as a template.
Do not attempt to use these parameters for a simulation of the real
machine.

The collimator starts at Z=50 cm and has 26 tungsten leaves opening
in the X direction.  The leaves are each 0.5cm wide and 6.0cm
thick.  The Z focus of the leaf sides is at Z=-1000 cm, resulting
in sides that are essentially straight up and down.  The Z focus
of the leaf ends is at Z=0, which is the position of the beam source.
In this example, leaf opening coordinates are chosen to create a
pattern of alternating open and closed leaves.

Electrons and photons in both the collimator and the opening regions
will be followed down to kinetic energies of 10 keV (ECUT=0.521,
PCUT=0.01).  Dose deposited in the tungsten leaves will be stored
in dose zone 2, and dose deposited in the opening will be stored
in dose zone 1.  Finally, the option to ignore air gaps when doing
range rejection deep in the leaves is off.

26.0,           RMAX_CM
MLC based on mock 26 leaf pair Varian 2100C type of accelerator
1,1,            Leaves open in X direction, all have same thickness
50.0,           ZMIN
6.0,            ZTHICK
26, 0.50,       NUM_LEAF, LEAFWIDTH
-6.5,           START POSITION
0,2, 0,4,       WIDTH AND HEIGHT OF SCREW
0.2, 2,0,       WIDTH, HEIGHT, Z OF TONGUE, tongue centred in leaf
0.21,2.6,0,     WIDTH, HEIGHT, Z OF GROOVE, groove centred in leaf
0.005,          LEAFGAP BETWEEN ADJACENT LEAVES
1,              ENDTYPE IS FOCUSED AND DIVERGENT
0.0,            ZFOCUS(2)
-1000.0         ZFOCUS(1)
0.0,0.0
-5.0, 5.0
0.0,0.0
-5.0, 5.0
0.0,0.0
-5.0, 5.0
0.0,0.0
-5.0, 5.0
0.0,0.0
-5.0, 5.0
0.0,0.0
-5.0, 5.0
0.0,0.0
```

```
-5.0, 5.0
0.0,0.0
-5.0, 5.0
0.0,0.0
-5.0, 5.0
0.0,0.0
-5.0, 5.0
0.0,0.0
-5.0, 5.0
0.0,0.0
-5.0, 5.0
0.0,0.0
-5.0, 5.0
0.5210,  0.010,  1,  0
AIR700ICRU
0.5210,  0.010,  2,  0, 0
W700ICRU
```

Note that VARMLC has an additional input, IGNOREGAPS, appearing on the same line as ECUT, PCUT etc for the leaves. This input increases the efficiency of range rejection for particles in the leaves. If IGNOREGAPS is set to 1, then all air gaps within the leaves (ie gaps between leaves and the Z-directional gaps caused by the presence of carriage screws) are ignored when doing charged particle range rejection provided that the particle is in the leaves and satisfies:

particle X $<$ minimum X of all leaf openings (excluding rounded or focused leaf end) or
particle X $>$ maximum X of all leaf openings (excluding leaf end)

if the leaves are parallel to X (ORIENT=1) or:

particle Y $<$ minimum Y of all leaf openings (excluding leaf end) or
particle Y $>$ maximum Y of all leaf openings (excluding leaf end)

if the leaves are parallel to Y (ORIENT=0). Exact transport is preserved in the shaped leaf ends (rounded or focused) by excluding them from the volume in which air gaps are ignored. This is important, since the leaf ends define the field shape and particles interacting in the ends are more likely to reach the field at the SSD.

Use of the IGNOREGAPS option can reduce the CPU time spent in VARMLC by a factor of 2. However, if the multi-leaf collimator you are modeling has significant air gaps between leaves, we recommend that you run with this option turned off (ie IGNOREGAPS set to 0; the default) so that there is exact transport everywhere.

15.3.17 MLCE

The MLCE CM is used to model multi-leaf collimators specific for Elekta machines. Instead of specifying a tongue-and-groove, the user specifies the dimensions of interlocking steps typical of this class of MLC. All leaves have identical cross-sections, and leaf sides are focused (always to Z=0) by tilting each leaf about an axis that runs parallel to the leaf opening direction and along the centre of its top surface. The entire leaf bank can also be rotated in a plane perpendicular to the leaf opening direction by a user-specified angle. As in VARMLC, the medium beyond the leaves in the direction perpendicular to the leaves defaults to be the same as the medium in the leaf opening(s) (ie AIR).

This CM was mostly coded by Nick Reynaert at the University of Ghent. There have been some modifications of inputs and some small bugs were fixed.

Details of MLCE are shown in Figure 36. All leaves have identical cross-sections, which are defined using an imaginary "central leaf". For this leaf, the user specifies the 4 points defining the cross-section (which is symmetric about X=0 if the steps are ignored), `X3`, `ZMIN`, `X4`, `ZMAX`, the Z positions of the left and right steps, `ZSTEPL`,`ZSTEPR`, and the step width, `TGW`. Note that `ZSTEPL` must be >`ZSTEPR` for the steps to fit into one another once the leaves are generated. The central leaf is then duplicated `NUM_LEAF` times, and each leaf is rotated/translated in the X (`ORIENT`=0) or Y (`ORIENT`=1) direction. Rotation angle and translation distance are determined by user inputs which define the spacing between the centres of the leaf cross-sections, `SPACE`, as projected down to `SSD` and the requirement that the leaf sides focus to Z=0. For leaf I=1,2,...,`NUM_LEAF`, the cross-section is first rotated about X=0 (`ORIENT`=0) or Y=0 (`ORIENT`=1), Z=`ZMIN` using:

$$rotation = ATAN\left[(2I - NUM\_LEAF - 1)\left(\frac{SPACE}{2}\right)\left(\frac{1}{SSD}\right)\right] \tag{10}$$

and then translated in the X (`ORIENT`=0) or Y (`ORIENT`=1) direction using:

$$translation = (2I - NUM\_LEAF - 1)\left(\frac{SPACE}{2}\right)\left(\frac{ZMIN}{SSD}\right) \tag{11}$$

where `NUM_LEAF` must be an even number. Note that the result of applying Equations 10 and 11 is that cross-sections for leaves 1 - `NUM_LEAF`/2 are rotated/translated in the negative sense with respect to the Z-axis while cross-sections for leaves `NUM_LEAF`/2+1 - `NUM_LEAF` are rotated/translated in the positive sense and the cross-section of the leaf bank is symmetric about the Z-axis.

The user also has the option to rotate the entire leaf bank about the axis X=0 (`ORIENT`=0) or Y=0 (`ORIENT`=1), Z=`ZMIN` by angle `LBROT` (which must be specified in radians). This rotates the central axis of the leaves (ie the axis along which they are focused) from Z to Z', as shown in Figure 36. Rotation of the leaves, both individually and of the entire leaf bank, requires automatic resetting of `ZMIN` and `ZMAX`, shown as `(adjusted)` values in the figure.

Figure 36 also shows the two types of leaf ends possible with MLCE. Cylindrical leaf ends (`ENDTYPE`=0) are defined by the radius, `LEAFRADIUS`, and Z position, `ZCIL`, of the cylinder describing the ends. Leaf openings are specified by the X (`ORIENT`=1) or Y (`ORIENT`=0) coordinates of the cylinder origins for the positive (`POS`) and negative (`NEG`) portions of the

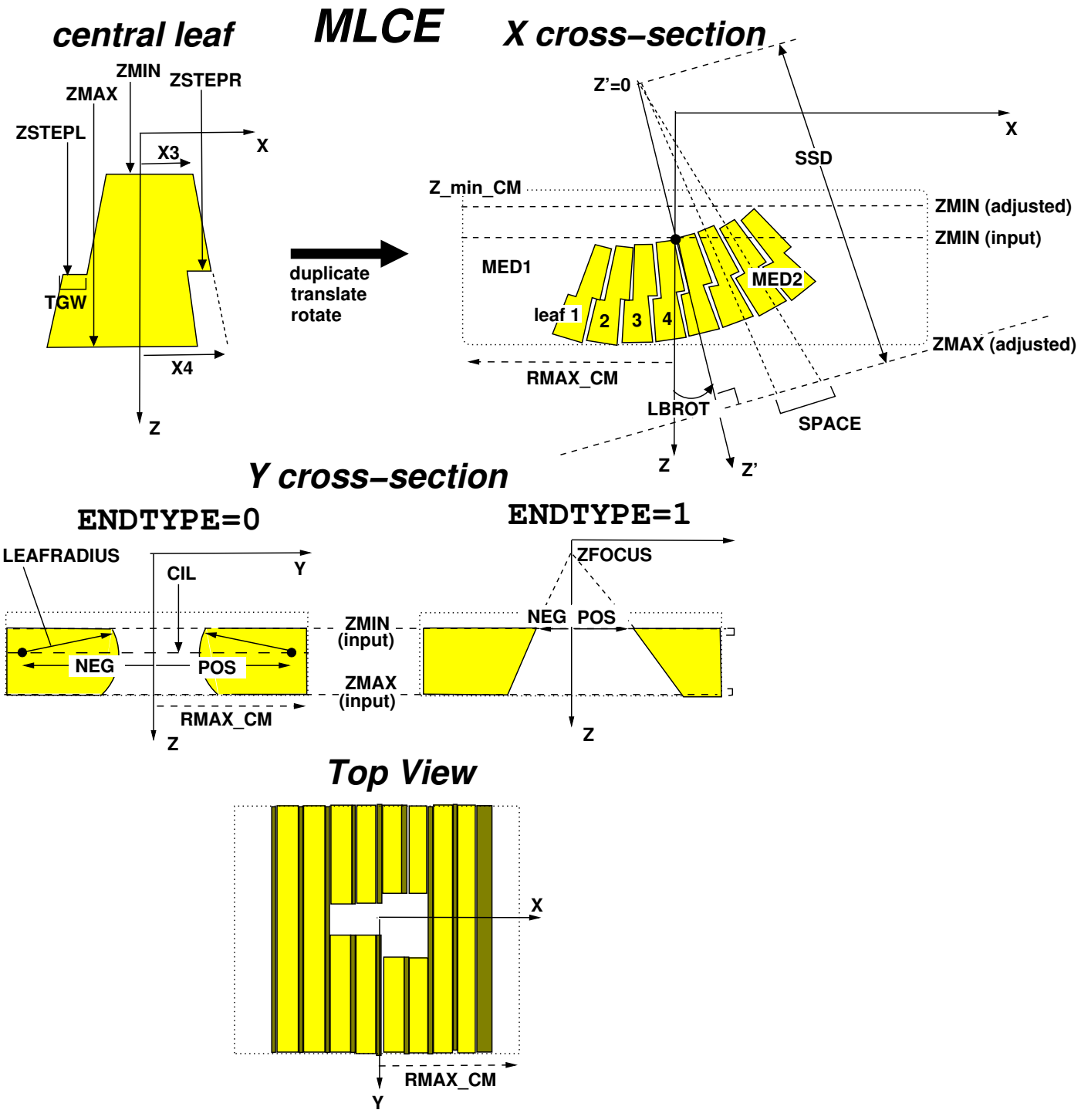


Figure 36: Example MLCE with 8 leaves (NUM_LEAF=8) opening in the Y direction (ORIENT=0) and with a leaf bank rotation angle of LBROT. The figure also shows the cross-section of the imaginary "central leaf", used to create the leaves, and the two possible leaf end types: cylindrical (ENDTYPE=0) and straight (ENDTYPE=1). See text for more details.

leaf. Straight leaf ends (ENDTYPE=1) are angled according to the user-input ZFOCUS, and leaf openings are specified by the X ORIENT=1) or Y (ORIENT=0) coordinates of the ends of the positive (POS) and negative (NEG) portions of the leaf at Z=ZMIN.

Note that MLCE input allows you to apply values of POS and NEG to groups of adjacent leaves, which can save tedious input in the case where many adjacent leaves have the same opening coordinates. See the input description below for more details.

The input format for MLCE and an example input are given below.

```
CARDS CM_$MLCE
**************
-1  Dummy line to indicate start of CM

 0  RMAX_CM(ICM_$MLCE) (F10.0):  Half-width of CM boundary (cm).

 1  TITLE_$MLCE (60A1):  Title of CM.

 2  ORIENT_$MLCE (I5) = 0 for leaves parallel to Y direction
                      = 1 for leaves parallel to X direction

 3  NUM_LEAF_$MLCE: Number of leaves.  Note: this must be even.

 4  ZMIN_$MLCE,ZMAX_$MLCE (2F15.0): upper and lower z coordinates
      of leafbank (before tilt, see below)

 5  ZSTEPL_$MLCE, ZSTEPR_$MLCE: Z-coordinates of left and right step
      in central leaf (an imaginary, unrotated leaf on the Z axis).

 6  TGW_$MLCE (F15.0): X (ORIENT_$MLCE=0) or Y (ORIENT_$MLCE=1)
      width of steps in central leaf (cm).

 7  X3_$MLCE, X4_$MLCE (2F15.0): X (ORIENT_$MLCE=0) or Y
       (ORIENT_$MLCE=1) coordinates of the upper right and
       lower right corners of central leaf, ignoring steps defined above.

 8  SPACE_$MLCE, SSD_$MLCE (2F15.0)

       SPACE_$MLCE: distance between centres of adjacent leaves in X
                    (ORIENT_$MLCE=0) or Y (ORIENT_$MLCE=1) direction
                    as projected to SSD_$MLCE (cm).
         SSD_$MLCE: distance from Z=0 at which SPACE_$MLCE is
                    defined (cm).

      Leaf numbers I= 1-NUM_LEAF_$MLCE/2 are created by rotating a
      duplicate of the central leaf about the axis X=0 (if
      ORIENT_$MLCE=0) or Y=0 (if ORIENT_$MLCE=1),
      Z=ZMIN_$MLCE by an angle:
      ARCTAN(-(2I-1)*SPACE_$MLCE/2.*ZMIN_$MLCE/SSD_$MLCE)
      and then translating it in the X (if ORIENT_$MLCE=0) or
```

```
      or Y (ORIENT_$MLCE=1) direction by a distance
      -(2I-1)*SPACE_$MLCE/2.*ZMIN_$MLCE/SSD_$MLCE

  Leaf numbers I= NUM_LEAF_$MLCE/2+1 NUM_LEAF_$MLCE are created by
  rotating a duplicate of the central leaf about the axis X=0
  (if ORIENT_$MLCE=0) or Y=0 (if ORIENT_$MLCE=1), Z=ZMIN_$MLCE by:
  ARCTAN((2I-1)*SPACE_$MLCE/2.*ZMIN_$MLCE/SSD_$MLCE)
  and then translating it in the X (if ORIENT_$MLCE=0) or
  or Y (ORIENT_$MLCE=1) direction by a distance
  (2I-1)*SPACE_$MLCE/2.*ZMIN_$MLCE/SSD_$MLCE

 9  LBROT_$MLCE (F15.0): Leaf bank rototian angle (tilt) about
     X=0 (ORIENT_$MLCE=0) or Y=0 (ORIENT_$MLCE=1) and Z=ZMIN_$MLCE
     (radians).  This is applied to the leaves after they have been
     translated/rotated according to SPACE_$MLCE, SSD_$MLCE above.

10  ENDTYPE_$MLCE (I5) : The  type of leaf end :
           0 -- rounded (cylindrical) leaf end and
           1 -- focused divergent leaf end.

IF ENDTYPE_$MLCE=0
11  LEAFRADIUS_$MLCE,CIL_$MLCE (2F15.0)

        LEAFRADIUS_$MLCE:  Radius curvature leaf ends
               CIL_$MLCE:  Z position from which LEAFRADIUS_$MLCE is
                           defined

IF ENDTYPE_$MLCE=1
11  ZFOCUS_$MLCE (F15.0):  Z position of focal point of leaf ends

 Repeat 12 until coordinates of all leaves are defined once.  Leaves
 are numbered 1,2,...NUM_LEAF_$MLCE, where numbering goes from leaf
 1 to leaf NUM_LEAF_$MLCE. Convention is lower to upper or
 left to right depending on ORIENT_$MLCE i.e from negative to
 positive.

12  NEG_$MLCE, POS_$MLCE, NUM_$MLCE (2F15.0,I5)

  NEG_$MLCE:   Min. Y (ORIENT_$MLCE=0) or X (ORIENT_$MLCE=1)
                 of a) opening in leaf I at ZMIN_$MLCE (ENDTYPE=1)
                 or b) of origin of cylindrical leaf end (ENDTYPE=0)
  POS_$MLCE:   Max. Y (ORIENT_$MLCE=0) or X (ORIENT_$MLCE=1)
                 of a) opening in leaf I at ZMIN_$MLCE (ENDTYPE=1)
                 or b) of origin of cylindrical leaf end (ENDTYPE=0)
  NUM_$MLCE:   Apply NEG_$MLCE and POS_$MLCE to leaves
                 I,...,I+NUM_$MLCE-1.  Defaults to 1 if set <=0.
                 Defaults to NUM_LEAF_$MLCE-I+1 if set >
                 NUM_LEAF_$MLCE-I+1.
```

```
13   ECUT, PCUT, DOSE_ZONE, IREGION_TO_BIT in opening(s) and
                                          air gaps (2F15.0,I5)

       ECUT, PCUT:  Cutoff energies for electrons and photons.
       DOSE_ZONE:   Dose scoring flag, 0 to not score dose
  IREGION_TO_BIT:   Bit number associated with this region

14  MED_IN (24A1):  Medium in opening(s) and air gaps
                    used to set MED_INDEX.

15  ECUT, PCUT, DOSE_ZONE, IREGION_TO_BIT in leaves (2F15.0,I5):

       ECUT, PCUT:  Cutoff energies for electrons and photons.
       DOSE_ZONE:   Dose scoring flag, 0 to note score dose
  IREGION_TO_BIT:   Bit number associated with this region

16  MED_IN (24A1):  Medium of leaves,
                    used to set MED_INDEX.

 Example
 *******

 The following example defines a multi-leaf collimator design based
 loosely on that used with the Elekta SLiplus 40 leaf pair.
 Actual parameters are DIFFERENT - this serves just as a template.
 Do not attempt to use these parameters for a simulation of the real
 machine.

 The collimator starts at Z=30 cm and has 40 tungsten leaves opening
 in the X direction.  The leaves are each ~0.4cm wide and 7.0 cm
 thick.  In this example, the leaf openings will form a barbel shape
 with its long axis parallel to Y.  It will be slightly off-centre
 due to the leaf bank rotation of -0.01 rads.

 Electrons and photons in both the collimator and the opening regions
 will be followed down to kinetic energies of 10 keV (ECUT=0.521,
 PCUT=0.01).  Dose deposited in the tungsten leaves will be stored
 in dose zone 2, and dose deposited in the opening will be stored
 in dose zone 1.


 In this example the numbers are only approximate, more details should
 be obtained from the vendor
 26.0,           RMAX_CM
 MLC based on mock 40 leaf pair Elekta SLiplus type of accelerator
 1,              Leaves open in X direction
 40,             40 leaf paires
 30.0,37.0,      ZMIN,ZMAX
 34.0, 33.5,     ZSTEPL,ZSTEPR
```

```
 0.04,           step width
 0.17,0.2,       X3, X4 of central leaf
 1.2, 100.0,     leaf centres spaced 1.2 cm apart projected to SSD=100cm
 -0.01,          leaf bank tilt angle (radians)
 0,              ENDTYPE IS CURVED
 15.0,33.5,      curvature radius, zposition cylinder axis curvature
-15.0, 15.0, 16
-17.0, 17.0, 2
-16.0, 16.0, 4
-17.0, 17.0, 2
-15.0, 15.0, 16
0.5210,  0.010,  1,  0
AIR700ICRU
0.5210,  0.010,  2,  0
W700ICRU
```

15.3.18 SYNCMLCE

SYNCMLCE is a synchronized version of MLCE (See Section 15.3.17 above) that can be used to simulate changing leaf opening coordinates during an accelerator simulation. Similar to DYNVMLC (See Section 15.3.19), motion of opening coordinates can be simulated in "dynamic" or "step-and-shoot" mode. In dynamic mode, the leaves move while the beam is on, while in step-and-shoot mode, motion of the leaves occurs while the beam is off. When used in either mode, leaf opening coordinates in SYNCMLCE can be synchronized with the settings of any other synchronized CMs (SYNCJAWS, SYNCVMLC and SYNCHDMLC) in the accelerator and with the motion of source 20 (synchronized phase space) or source 21 (synchronized BEAM treatment head) in DOSXYZnrc[15]. SYNCMLCE was contributed by Lobo & Popescu[24].

Inputs to SYNCMLCE are similar to that for MLCE with the exception that, on the same line as `ORIENT`, defining the leaf orientation, there is an input, `MODE`, defining the dynamic mode of SYNCMLCE. If `MODE`=0, then SYNCMLCE is used in static mode, and the inputs are identical to MLCE. If `MODE`=1 ("dynamic") or 2 ("step-and-shoot"), however, instead of reading the leaf opening coordinates from the BEAMnrc input file, SYNCMLCE reads the opening coordinates for `NFIELD` multiple fields from a separate file.

The format of the file of leaf opening coordinates is identical to that used for DYNVMLC, SYNCVMLC and SYNCHDMLC and is given in more detail in Section 15.3.19 below. In the case of SYNCMLCE, the input parameter, `INDEX(I)`, defining the cumulative probability of all fields up to and including field `I`, is interpreted as `muIndex(I)`, the fraction of monitor units delivered up to and including field `I`.

For each primary history, a random number, `MU_RND`$\in$ [0,1], is compared to the `muIndex(I)` and is used to determine which field to use and, in "dynamic" mode, to calculate the leaf opening coordinates. A more detailed description of how this is done is given in the documentation for DYNVMLC (Section 15.3.19). The equation used by SYNCMLCE to calculate leaf opening coordinates in "dynamic" mode is the same as Equation (9), with `INDEX` replaced by `muIndex` and `RNDM1` replaced by `MU_RND`.

If SYNCMLCE is the only synchronized CM in the accelerator, then `MU_RND` is chosen by SYNCMLCE and used only by SYNCMLCE. However, if there are other synchronized CMs in the accelerator, then `MU_RND` is generated by the first (most upstream) synchronized CM and then passed down to all downstream synchronized CMs. Thus, the motion of SYNCMLCE leaves can be synchronized with the coordinates of all other synchronized CMs in the accelerator.

Furthermore, if the accelerator is compiled as a shared library, then the value of `MU_RND` used in SYNCMLCE is also passed on to DOSXYZnrc simulations when using DOSXYZnrc sources 20 (synchronized phase space) or 21 (synchronized BEAM treatment head simulation). This allows the motion of these dynamic sources to be synchronized with that of the leaf openings. See the DOSXYZnrc Users Manual[15] for more details.

An example of a file specifying dynamic leaf settings for SYNCMLCE is included with the distribution. This file is `$OMEGA_HOME/beamnrc/CMs/sample_syncmlce.sequence`.

The input format for SYNCMLCE and an example input are shown below:

```
CARDS CM_$SYNCMLCE
**************
-1  Dummy line to indicate start of CM

 0  RMAX_CM(ICM_$SYNCMLCE) (F10.0):  Half-width of CM boundary (cm).

 1  TITLE_$SYNCMLCE (60A1):  Title of CM.

 2  ORIENT_$SYNCMLCE, MODE_$SYNCMLCE (2I5)

      ORIENT_$SYNCMLCE = 0 for leaves parallel to Y direction
                       = 1 for leaves parallel to X direction
      MODE_$SYNCMLCE = 0 for static mode (same as MLCE)
                     = 1 for dynamic mode
                     = 2 for step-and-shoot mode

 3  NUM_LEAF_$SYNCMLCE: Number of leaves.  Note: this must be even.

 4  ZMIN_$SYNCMLCE,ZMAX_$SYNCMLCE (2F15.0): upper and lower z coordinates
      of leafbank (before tilt, see below)

 5  ZSTEPL_$SYNCMLCE, ZSTEPR_$SYNCMLCE: Z-coordinates of left and right step
      in central leaf (an imaginary, unrotated leaf on the Z axis).

 6  TGW_$SYNCMLCE (F15.0): X (ORIENT_$SYNCMLCE=0) or Y (ORIENT_$SYNCMLCE=1)
      width of steps in central leaf (cm).

 7  X3_$SYNCMLCE, X4_$SYNCMLCE (2F15.0): X (ORIENT_$SYNCMLCE=0) or Y
       (ORIENT_$SYNCMLCE=1) coordinates of the upper right and
       lower right corners of central leaf, ignoring steps defined above.

 8  SPACE_$SYNCMLCE, SSD_$SYNCMLCE (2F15.0)

       SPACE_$SYNCMLCE: distance between centres of adjacent leaves in X
                    (ORIENT_$SYNCMLCE=0) or Y (ORIENT_$SYNCMLCE=1) direction
                    as projected to SSD_$SYNCMLCE (cm).
         SSD_$SYNCMLCE: distance from Z=0 at which SPACE_$SYNCMLCE is
                    defined (cm).

      Leaf numbers I= 1-NUM_LEAF_$SYNCMLCE/2 are created by rotating a
      duplicate of the central leaf about the axis X=0 (if
      ORIENT_$SYNCMLCE=0) or Y=0 (if ORIENT_$SYNCMLCE=1),
      Z=ZMIN_$SYNCMLCE by an angle:
      ARCTAN(-(2I-1)*SPACE_$SYNCMLCE/2.*ZMIN_$SYNCMLCE/SSD_$SYNCMLCE)
      and then translating it in the X (if ORIENT_$SYNCMLCE=0) or
      or Y (ORIENT_$SYNCMLCE=1) direction by a distance
      -(2I-1)*SPACE_$SYNCMLCE/2.*ZMIN_$SYNCMLCE/SSD_$SYNCMLCE

    Leaf numbers I= NUM_LEAF_$SYNCMLCE/2+1 NUM_LEAF_$SYNCMLCE are created by
```

```
  rotating a duplicate of the central leaf about the axis X=0
  (if ORIENT_$SYNCMLCE=0) or Y=0 (if ORIENT_$SYNCMLCE=1), Z=ZMIN_$SYNCMLCE by:
  ARCTAN((2I-1)*SPACE_$SYNCMLCE/2.*ZMIN_$SYNCMLCE/SSD_$SYNCMLCE)
  and then translating it in the X (if ORIENT_$SYNCMLCE=0) or
  or Y (ORIENT_$SYNCMLCE=1) direction by a distance
  (2I-1)*SPACE_$SYNCMLCE/2.*ZMIN_$SYNCMLCE/SSD_$SYNCMLCE

 9  LBROT_$SYNCMLCE (F15.0): Leaf bank rototian angle (tilt) about
  X=0 (ORIENT_$SYNCMLCE=0) or Y=0 (ORIENT_$SYNCMLCE=1) and Z=ZMIN_$SYNCMLCE
  (radians).  This is applied to the leaves after they have been
  translated/rotated according to SPACE_$SYNCMLCE, SSD_$SYNCMLCE above.

10  ENDTYPE_$SYNCMLCE (I5) : The  type of leaf end :
           0 -- rounded (cylindrical) leaf end and
           1 -- focused divergent leaf end.

IF ENDTYPE_$SYNCMLCE=0
11  LEAFRADIUS_$SYNCMLCE,CIL_$SYNCMLCE (2F15.0)

        LEAFRADIUS_$SYNCMLCE:  Radius curvature leaf ends
               CIL_$SYNCMLCE:  Z position from which LEAFRADIUS_$SYNCMLCE
                               is defined

IF ENDTYPE_$SYNCMLCE=1
11  ZFOCUS_$SYNCMLCE (F15.0):  Z position of focal point of leaf ends

 If MODE_$SYNCMLCE=0 (static field)

 Repeat 12a until coordinates of all leaves are defined once.  Leaves
 are numbered 1,2,...NUM_LEAF_$SYNCMLCE, where numbering goes from leaf
 1 to leaf NUM_LEAF_$SYNCMLCE. Convention is lower to upper or
 left to right depending on ORIENT_$SYNCMLCE i.e from negative to
 positive.

12a  NEG_$SYNCMLCE, POS_$SYNCMLCE, NUM_$SYNCMLCE (2F15.0,I5)

  NEG_$SYNCMLCE:   Min. Y (ORIENT_$SYNCMLCE=0) or X (ORIENT_$SYNCMLCE=1)
                 of a) opening in leaf I at ZMIN_$SYNCMLCE (ENDTYPE=1)
                 or b) of origin of cylindrical leaf end (ENDTYPE=0)
  POS_$SYNCMLCE:   Max. Y (ORIENT_$SYNCMLCE=0) or X (ORIENT_$SYNCMLCE=1)
                 of a) opening in leaf I at ZMIN_$SYNCMLCE (ENDTYPE=1)
                 or b) of origin of cylindrical leaf end (ENDTYPE=0)
  NUM_$SYNCMLCE:   Apply NEG_$SYNCMLCE and POS_$SYNCMLCE to leaves
                 I,...,I+NUM_$SYNCMLCE-1.  Defaults to 1 if set <=0.
                 Defaults to NUM_LEAF_$SYNCMLCE-I+1 if set >
                 NUM_LEAF_$SYNCMLCE-I+1.

 If MODE_$SYNCMLCE=1 (dynamic delivery) or 2 (step-and-shoot delivery)
```

```
12b  mlc_file (A80): The full name of the file containing leaf opening
                     data.  The format of the file contents is as follows:

    MLC_TITLE (A80)
    NFIELDS_$SYNCMLCE (I10)
    FOR I=1,NFIELDS_$SYNCMLCE[
     MUINDEX_$SYNCMLCE(I) (F15.0)
     NEG_$SYNCMLCE, POS_$SYNCMLCE, NUM_$SYNCMLCE (2F15.0,I5) -- repeat this
                                                               line until
                                                               coordinates
                                                               for all leaves
                                                               have been
                                                               defined for
                                                               field I.
    ]

    where:

            MLC_TITLE:  A title line
   NFIELDS_$SYNCMLCE:  Total number of fields
MUINDEX_$SYNCMLCE(I):  Fractional monitor unit index up to and including
                       field I. 0 <= MUINDEX_$SYNCMLCE(I) <= 1 and
                       MUINDEX_$SYNCMLCE(I) >= MUINDEX_$SYNCMLCE(I-1).
                       This number is compared to a random number on
                       [0,1] at the start of each history; if the random
                       number is <= MUINDEX_$SYNCMLCE(I), then field I is
                       used.
        NEG_$SYNCMLCE:  Min. Y (ORIENT_$SYNCMLCE=0) or X (ORIENT_$SYNCMLCE=
                       1) of front opening in leaf (ie the opening at
                       ZMIN_$SYNCMLCE) if ENDTYPE=1, or of origin of
                       cylinder defining rounded end of leaf if ENDTYPE=0
                       for leaf J in field I.
        POS_$SYNCMLCE:  Max. Y (ORIENT_$SYNCMLCE=0) or X (ORIENT_$SYNCMLCE=
                       1) of front opening in leaf if ENDTYPE=1, or of 9
                       origin of cylinder defining rounded end of leaf if
                       ENDTYPE=0 for leaf J in field I.
        NUM_$SYNCMLCE:  Apply NEG_$SYNCMLCE and POS_$SYNCMLCE to leaves
                       J,...,J+NUM_$SYNCMLCE-1.  Defaults to 1 if set <=0.
                       Defaults to TOT_LEAF_$SYNCMLCE-J+1 if set >
                       TOT_LEAF_$SYNCMLCE-J+1.

  Note that the inputs NEG_$SYNCMLCE, POS_$SYNCMLCE and NUM_$SYNCMLCE have
  the same meanings as in 12a (static field inputs) but that they must
  now be repeated for every field I.


13   ECUT, PCUT, DOSE_ZONE, IREGION_TO_BIT in opening(s) and
                                           air gaps (2F15.0,I5)
```

```
       ECUT, PCUT:  Cutoff energies for electrons and photons.
       DOSE_ZONE:   Dose scoring flag, 0 to not score dose
  IREGION_TO_BIT:   Bit number associated with this region

14  MED_IN (24A1):  Medium in opening(s) and air gaps
                    used to set MED_INDEX.

15  ECUT, PCUT, DOSE_ZONE, IREGION_TO_BIT in leaves (2F15.0,I5):

       ECUT, PCUT:  Cutoff energies for electrons and photons.
       DOSE_ZONE:   Dose scoring flag, 0 to note score dose
  IREGION_TO_BIT:   Bit number associated with this region

16  MED_IN (24A1):  Medium of leaves,
                    used to set MED_INDEX.

 Example
 *******

 The following example defines a multi-leaf collimator design based
 loosely on that used with the Elekta SLiplus 40 leaf pair.
 Actual parameters are DIFFERENT - this serves just as a template.
 Do not attempt to use these parameters for a simulation of the real
 machine.

 The collimator starts at Z=30 cm and has 40 tungsten leaves opening
 in the X direction.  The leaves are each ~0.4cm wide and 7.0 cm
 thick.  In this example, the leaf openings will form a barbel shape
 with its long axis parallel to Y.  It will be slightly off-centre
 due to the leaf bank rotation of -0.01 rads.

 Electrons and photons in both the collimator and the opening regions
 will be followed down to kinetic energies of 10 keV (ECUT=0.521,
 PCUT=0.01).  Dose deposited in the tungsten leaves will be stored
 in dose zone 2, and dose deposited in the opening will be stored
 in dose zone 1.

 In this example, the MLC is being used in step-and-shoot mode, with
 the leaf opening coordinates specified in a file.

 26.0,           RMAX_CM
 MLC based on mock 40 leaf pair Elekta SLiplus type of accelerator
 1,2             Leaves open in X direction, step-and-shoot mode
 40,             40 leaf paires
 30.0,37.0,      ZMIN,ZMAX
 34.0, 33.5,     ZSTEPL,ZSTEPR
 0.04,           step width
 0.17,0.2,       X3, X4 of central leaf
 1.2, 100.0,     leaf centres spaced 1.2 cm apart projected to SSD=100cm
```

```
 -0.01,          leaf bank tilt angle (radians)
 0,              ENDTYPE IS CURVED
 15.0,33.5,      curvature radius, zposition cylinder axis curvature
 /home/bwalters/HEN_HOUSE/omega/beamnrc/CMs/sample_syncmlce.sequence
0.5210,  0.010,  1,  0
AIR700ICRU
0.5210,  0.010,  2,  0
W700ICRU
```

15.3.19 DYNVMLC

DYNVMLC is a CM specifically designed to model the Varian Millenium multi-leaf collimator. The code is based on VARMLC. The user specifies cross-sections perpendicular to the leaf opening direction for the 3 leaf types (FULL, TARGET and ISOCENTER leaves) found in the mlc. Each leaf in the leaf bank (usually comprising 120 leaves in all) is assigned a type, with TARGET/ISOCENTER leaves always occuring in pairs. The user also specifies a Z focal point for the leaf sides and either the radius (for cylindrical ends) or Z focal point (for straight ends) for the leaf ends. Note that DYNVMLC leaf cross sections are more complex than those in the VARMLC CM.

DYNVMLC allows the simulation of multiple fields (where a field is defined by a complete set of leaf openings) during a single run. Leaf opening coordinates can be simulated changing while the beam is on (dynamic, or `MODE=1`, simulation) or while the beam is off (step-and-shoot, or `MODE=2`, simulation). For these simulations, the user must supply a file containing the leaf opening data for each field and the relative intensity of each field. Alternatively, DYNVMLC can be used to simulate a fixed set of leaf opening coordinates for the entire run (static, or `MODE=0`, simulation).

DYNVMLC was originally coded by Emily Heath at McGill University. There have been some modifications of inputs and some bugs were fixed.

Figure 37 shows an example DYNVMLC with 8 leaves (`NUM_LEAF`=8) opening in the Y direction (`ORIENT`=0). The X cross-sections of the 3 leaf types, along with the dimensions that you must specify, are shown in detail at the top. The meanings of the labels are shown in Table 3.

FULL and ISOCENTER leaves extend from `ztip` to `ZMIN+ZTHICK`, while TARGET leaves extend from `ZMIN` to `ztip`. For each leaf, you must also specify the Z dimensions of a driving screw hole (filled with `MED3`) that spans the entire width of the leaf cross-section. When specifying the cross-sections of each leaf type, the grid lines shown in Figure 37 must not change order, however they can overlap. This means, for example, that when specifying the FULL leaf dimensions $\texttt{zt} \leq \texttt{zg}$, $\texttt{wt} \leq \texttt{wtip} \leq \texttt{wt + wts - wbs}$, etc. Also, Z positions of tongues and grooves must allow leaves to fit together in the following combinations: FULL/FULL (ie $\texttt{zt} \leq \texttt{zg}$), FULL/TARGET (ie $\texttt{zg}_{FULL} \geq \texttt{zt}_{TARGET}$), TARGET/ISOCENTER (ie $\texttt{zg}_{TARGET} \leq \texttt{zt}_{ISOCENTER}$), ISOCENTER/TARGET (ie $\texttt{zg}_{ISOCENTER} \geq \texttt{zt}_{TARGET}$) and ISOCENTER/FULL (ie $\texttt{zg}_{ISOCENTER} \geq \texttt{zt}_{FULL}$). All of these requirements put fairly tight restrictions on cross-section dimensions.

Not shown in the leaf cross-sections is the distance, `HOLEPOS`, between the leaf end and the end of the driving screw hole in the leaf opening direction. This must also be specified for each leaf type.

Once you have specified the cross-sections for the 3 leaf types, then you must assign a type to each leaf in the leaf bank. Since TARGET/ISOCENTER leaves must always occur in pairs (with the TARGET leaf on the negative X side if `ORIENT`=0 or on the negative Y side if `ORIENT`=1), you really only have a choice of two leaf types: 1. FULL leaf or 2. TARGET/ISOCENTER pair. You can also specify the number of adjacent leaves that have the same type so that you do not have to input the type for each leaf or leaf pair separately. In the example shown in Figure 37, leaves 1,2,7,8 are FULL (type 1) and leaves 3-6 are

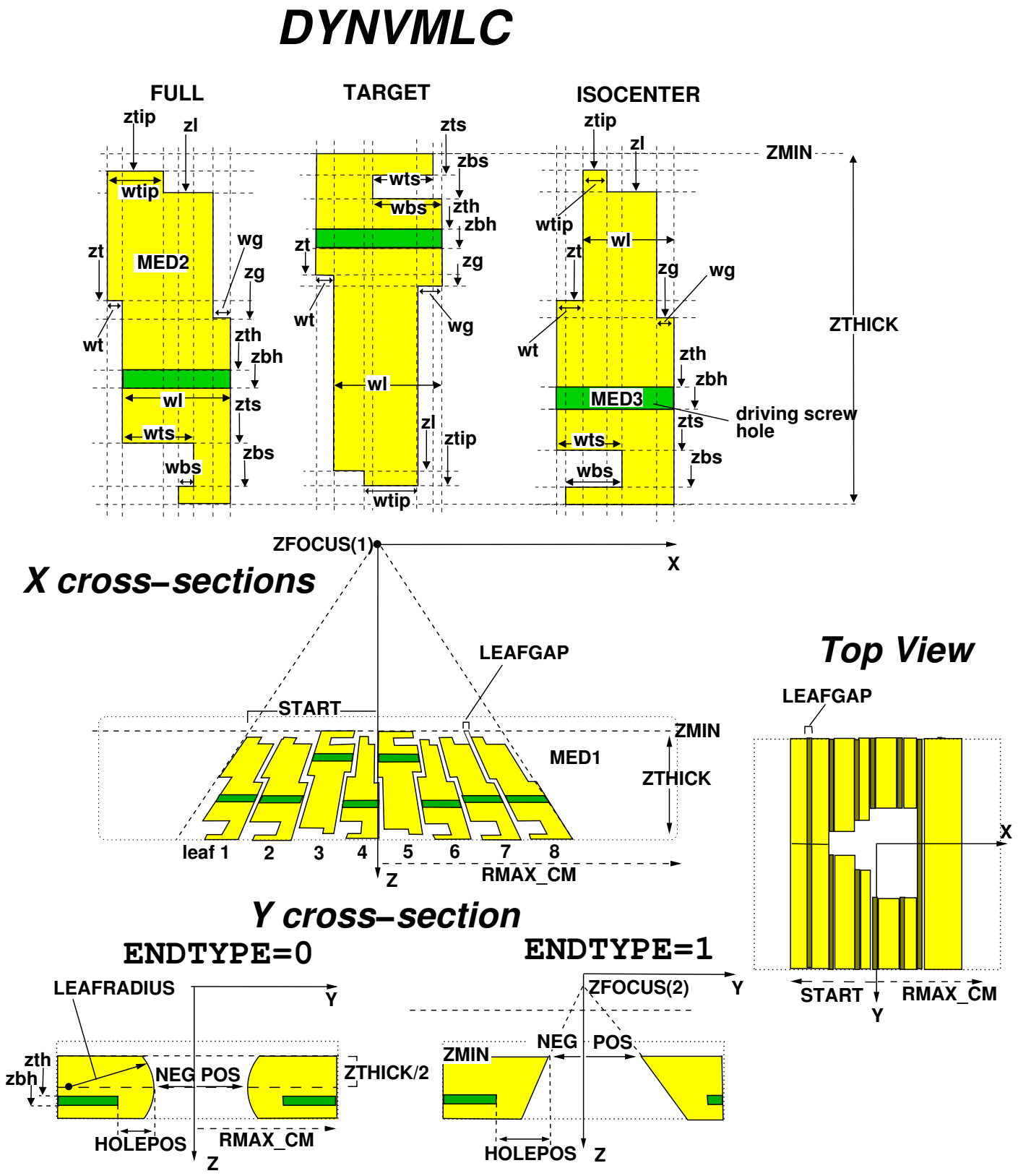


Figure 37: Example DYNVMLC with 8 leaves (NUM_LEAF=8) opening in the Y direction (ORIENT=0). At the top of the figure are detailed X cross-sections of the 3 leaf types, FULL, TARGET and ISOCENTER, showing the dimensions that the user must input. The dimension labels are explained in the text below. The Y cross-section shows the two possible leaf end types: cylindrical (ENDTYPE=0) and straight (ENDTYPE=1).

Table 3: Meaning of labels in Figure 37.

	Leaf Type		
	FULL	TARGET	ISOCENTER
widths:			
`wl`	leaf (excl. tongue)	leaf (excl. tongue)	leaf (excl. tongue)
`wt`	tongue	tongue	tongue
`wg`	groove	groove	groove
`wtip`	tip at top of leaf	tip at bottom of leaf	tip at top of leaf
`wts`	top support rail	top support rail	top support rail
`wbs`	bottom support rail	bottom support rail	bottom support rail
Z positions:			
`ztip`	top of tip	bottom of tip	top of tip
`zl`	top of leaf	bottom of leaf	top of leaf
`zt`	bottom of tongue	bottom of tongue	top of tongue
`zg`	bottom of groove	top of groove	bottom of groove
`zth`	top of driving screw hole	top of driving screw hole	top of driving screw hole
`zbh`	bottom of driving screw hole	bottom of driving screw hole	bottom of driving screw hole
`zts`	top of support rail	top of support rail	top of support rail
`zbs`	bottom of support rail	bottom of support rail	bottom of support rail

TARGET/ISOCENTER pairs (type 2).

The X cross-section of the entire leaf bank shows how the different leaf types fit together with interlocking steps and with an air gap between each leaf specified by the `LEAFGAP` input. The negative-most leaf begins at X=`START`. All leaf sides are focused on the Z axis at the user-specified focal point, `ZFOCUS(1)`. The focusing of leaf sides means that leaf widths are not uniform from leaf top to bottom (although in a realistic case, where the MLC is close to the SSD and `ZFOCUS(1)`=0, the variation in widths will be slight), and it is important to note that all X (`ORIENT`=0) or Y (`ORIENT`=1) dimensions that you input (whether they be widths for the individual leaf types or dimensions relevant to the entire leaf bank, such as `LEAFGAP` or `START`) are specified at `ZMIN`.

The Y cross-section in Figure 37 shows the two possible leaf end types: cylindrical (`ENDTYPE`=0) and straight focused (`ENDTYPE`=1). Cylindrical leaf ends have radius `LEAFRADIUS` with the origin of the radius at Z=`ZMIN`+`ZTHICK`/2 (ie at the mid-point of the leaf thickness). The negative and positive dimensions of the opening in a leaf with cylindrical ends are specified at Z=`ZMIN`+`ZTHICK`/2. Straight leaf ends are focused to Z=`ZFOCUS(2)`, and the negative and positive dimensions of the opening are specified at Z=`ZMIN`. The Y cross-section also shows how the distance from the end of the driving screw hole to the leaf end, `HOLEPOS`, is defined. In the case of cylindrical ends, it is the distance from the leaf end at Z=`ZMIN`+`ZTHICK`/2, while for straight ends, it is the distance from the end at Z=`ZMIN`. `HOLEPOS` remains the same for a given leaf type (FULL, TARGET or ISOCENTER) regardless of the leaf opening dimensions.

For static (`MODE`=0, the default) simulations, in which leaf openings are fixed for the

entire simulation, the leaf openings coordinates are specified in the GUI or input file in a manner similar to MLC, MLCQ, VARMLC, etc. Note that the user can specify the number of adjacent leaves that have the same opening dimensions, thus potentially saving input lines. For dynamic (MODE=1) or step-and-shoot (MODE=2) simulations, where multiple treatment fields are simulated in a single run, however, the user must supply a file of leaf opening data. The format of this file is:

```
TITLE
NFIELDS
FOR I=1,NFIELDS[
  INDEX(I)
  NEG, POS, NUM -- repeat until NEG, POS have been defined for all leaves
]
```

where TITLE is a title line for the file, NFIELDS is the number of treatment fields, INDEX(I) is the index of treatment field I, NEG is the lower Y (ORIENT=0) or X (ORIENT=1) coordinate of the leaf opening, POS is the upper Y or X coordinate of the leaf opening, and NUM is the number of adjacent leaves for which NEG and POS apply (defaults to 1 if left blank). Note that the NEG, POS, NUM line must be repeated until the opening coordinates for all leaves have been defined for field I.

The treatment field indices, INDEX(I)'s, are numbers in the range [0,1] (with INDEX(I)>INDEX(I-1)) which determine which field is used in a particular history. At the beginning of each history a random number in the range [0,1], RNDM1, is compared to (INDEX(I),I=1,NFIELDS). The lowest value of I for which INDEX(I)≥RNDM1 is the field used in the history. Note that this means that INDEX(I)-INDEX(I-1) is a measure of the probability, or intensity, of field I. In the case of step-and-shoot (MODE=2) runs, the leaf opening coordinates are simply those defined for field I. In the case of dynamic (MODE=1) simulations, however, RNDM1 is used to calculate intermediate opening coordinates between fields I and I-1 to simulate leaf coordinates changing while the beam is on. For example, if NEG(J,I) and NEG(J,I-1) are the lower coordinates of leaf J in fields I and I-1, respectively, then the intermediate lower coordinate used, NEG_INT is given by:

$$\texttt{NEG\_INT} = \texttt{NEG}(\texttt{J},\texttt{I}-1) + (\texttt{NEG}(\texttt{J},\texttt{I}) - \texttt{NEG}(\texttt{J},\texttt{I}-1)) * \frac{(\texttt{RNDM1} - \texttt{INDEX}(\texttt{I}-1))}{(\texttt{INDEX}(\texttt{I}) - \texttt{INDEX}(\texttt{I}-1))} \quad (12)$$

A sample file containing leaf opening data for DYNVMLC is included with the distribution.

Similar to VARMLC, DYNVMLC has an IGNOREGAPS input (appearing on the same line as ECUT, PCUT etc for the leaves) which can be set to 1 to instruct the code to ignore all air gaps within the leaf bank (ie gaps between leaves and Z-directional gaps caused by the presence of leaf tips, support rails, etc) and the driving screw holes when doing charged particle range rejection provided that the particle is in the leaf medium and:

particle X < minimum X of all leaf openings (excluding rounded or focused leaf end) or
particle X > maximum X of all leaf openings (excluding leaf end)

if the leaves are parallel to X (ORIENT=1) or:

particle Y < minimum Y of all leaf openings (excluding leaf end) or
particle Y > maximum Y of all leaf openings (excluding leaf end)

if the leaves are parallel to Y (ORIENT=0). Exact transport is preserved in the shaped leaf ends (rounded or focused) by excluding them from the volume in which air gaps and driving screw holes are ignored. Setting IGNOREGAPS=1 greatly increases the efficiency of range rejection in DYNVMLC and can reduce CPU time spent in the CM by a factor of 2. However, if you are concerned that the air gaps and driving screw holes have a significant effect on the beam field, we recommend that you run with IGNOREGAPS=0 to preserve exact transport everywhere. Note that in the case of dynamic or step-and-shoot simulations, the minimum and maximum opening coordinates must be recalculated at the beginning of each history.

The input format for DYNVMLC and an example input are given below.

```
CARDS CM_$DYNVMLC
********************
-1  Dummy line to indicate start of CM

 0  RMAX_CM(ICM_$DYNVMLC) (F10.5):  Half-width of CM boundary (cm).

 1  TITLE_$DYNVMLC (60A1):  Title of CM.

 2  ORIENT_$DYNVMLC, NGROUP_$DYNVMLC, MODE_$DYNVMLC (3I5)

          ORIENT_$DYNVMLC = 0 for leaves parallel to Y direction
                          = 1 for leaves parallel to X direction
          NGROUP_$DYNVMLC = number of groups of adjacent leaves where
                            all leaves in a group are:
                            1. FULL leaves
                            2. TARGET/ISOCENTER pairs with TARGET leaf
                               on the -X (ORIENT=0) or -Y (ORIENT=1) side
                            NGROUP_$DYNVMLC defaults to 3 if set <=0
          MODE_$DYNVMLC = 0 for single setting of leaf openings (static
                            field)
                        = 1 for dynamic mlc delivery--simulated leaf
                            movement while beam is on
                        = 2 for step-and-shoot delivery--beam off while
                            leaf positions change

 3  ZMIN_$DYNVMLC (F15.0): Z of top of MLC (excluding airgap)

 4  ZTHICK_$DYNVMLC (F15.0): Thickness of the leaves ( z-axis (cm))

 5  LEAFWIDTH_$DYNVMLC(1), WTONGUE_$DYNVMLC(1), WGROOVE_$DYNVMLC(1),
    WTIP_$DYNVMLC(1), WRAILTOP_$DYNVMLC(1), WRAILBOT_$DYNVMLC(1),
    ZTIP_$DYNVMLC(1), ZLEAF_$DYNVMLC(1), ZTONGUE_$DYNVMLC(1),
    ZGROOVE_$DYNVMLC(1), ZHOLETOP_$DYNVMLC(1), ZHOLEBOT_$DYNVMLC(1),
```

```
   HOLEPOS_FULL_$DYNVMLC, ZRAILTOP_$DYNVMLC(1), ZRAILBOT_$DYNVMLC(1)
      (15F15.0)

     For a FULL type leaf (all dimensions in cm--all widths are
                           projected back to ZMIN_$DYNVMLC):

        LEAFWIDTH_$DYNVMLC(1): Width of leaf (not including tongue)
          WTONGUE_$DYNVMLC(1): Width of tongue
          WGROOVE_$DYNVMLC(1): Width of groove
             WTIP_$DYNVMLC(1): Width of tip at top of leaf
         WRAILTOP_$DYNVMLC(1): Width of top of support rail
         WRAILBOT_$DYNVMLC(1): Width of bottom of support rail
             ZTIP_$DYNVMLC(1): Z at which tip at top of leaf begins
            ZLEAF_$DYNVMLC(1): Z of top of leaf
          ZTONGUE_$DYNVMLC(1): Z of bottom of tongue
          ZGROOVE_$DYNVMLC(1): Z of bottom of groove
         ZHOLETOP_$DYNVMLC(1): Z of top of driving screw hole
         ZHOLEBOT_$DYNVMLC(1): Z of bottom of driving screw hole
        HOLEPOS_FULL_$DYNVMLC: Distance of hole from leaf tip
         ZRAILTOP_$DYNVMLC(1): Z of top of support rail
         ZRAILBOT_$DYNVMLC(1): Z of bottom of support rail

      Note: Z positions are input in order of increasing Z.  Thus
            ZLEAF_$DYNVMLC(1)>=ZTIP_$DYNVMLC(1), etc.  See the BEAM
            manual or GUI help for restrictions on widths.


6  LEAFWIDTH_$DYNVMLC(2), WTONGUE_$DYNVMLC(2), WGROOVE_$DYNVMLC(2),
   WTIP_$DYNVMLC(2), WRAILTOP_$DYNVMLC(2), WRAILBOT_$DYNVMLC(2),
   ZRAILTOP_$DYNVMLC(2), ZRAILBOT_$DYNVMLC(2), ZHOLETOP_$DYNVMLC(2),
   ZHOLEBOT_$DYNVMLC(2), HOLEPOS_TAR_$DYNVMLC, ZTONGUE_$DYNVMLC(2),
   ZGROOVE_$DYNVMLC(2), ZLEAF_$DYNVMLC(2), ZTIP_$DYNVMLC(2) (15F15.0)

     For a TARGET type leaf (all dimensions in cm--all widths are
                           projected back to ZMIN_$DYNVMLC):

        LEAFWIDTH_$DYNVMLC(2): Width of leaf (not including tongue)
          WTONGUE_$DYNVMLC(2): Width of tongue
          WGROOVE_$DYNVMLC(2): Width of groove
             WTIP_$DYNVMLC(2): Width of tip at bottom of leaf
         WRAILTOP_$DYNVMLC(2): Width of top of support rail
         WRAILBOT_$DYNVMLC(2): Width of bottom of support rail
         ZRAILTOP_$DYNVMLC(2): Z of top of support rail
         ZRAILBOT_$DYNVMLC(2): Z of bottom of support rail
         ZHOLETOP_$DYNVMLC(2): Z of top of driving screw hole
         ZHOLEBOT_$DYNVMLC(2): Z of bottom of driving screw hole
         HOLEPOS_TAR_$DYNVMLC: Distance of hole from leaf tip
          ZTONGUE_$DYNVMLC(2): Z of bottom of tongue
          ZGROOVE_$DYNVMLC(2): Z of top of groove
```

```
            ZLEAF_$DYNVMLC(2): Z of bottom of leaf
             ZTIP_$DYNVMLC(2): Z of bottom of tip at bottom of leaf

     Note: Z positions are input in order of increasing Z.  Thus
           ZLEAF_$DYNVMLC(1)>=ZTIP_$DYNVMLC(1), etc.  See the BEAM
           manual or GUI help for restrictions on widths.


  7  LEAFWIDTH_$DYNVMLC(3), WTONGUE_$DYNVMLC(3), WGROOVE_$DYNVMLC(3),
     WTIP_$DYNVMLC(3), WRAILTOP_$DYNVMLC(3), WRAILBOT_$DYNVMLC(3),
     ZTIP_$DYNVMLC(3), ZLEAF_$DYNVMLC(3), ZTONGUE_$DYNVMLC(3),
     ZGROOVE_$DYNVMLC(3), ZHOLETOP_$DYNVMLC(3), ZHOLEBOT_$DYNVMLC(3),
     HOLEPOS_ISO_$DYNVMLC, ZRAILTOP_$DYNVMLC(3), ZRAILBOT_$DYNVMLC(3)
                 (15F15.0)

       For a ISOCENTER type leaf (all dimensions in cm--all widths are
                            projected back to ZMIN_$DYNVMLC):

          LEAFWIDTH_$DYNVMLC(3): Width of leaf (not including tongue)
            WTONGUE_$DYNVMLC(3): Width of tongue
            WGROOVE_$DYNVMLC(3): Width of groove
               WTIP_$DYNVMLC(3): Width of tip at top of leaf
           WRAILTOP_$DYNVMLC(3): Width of top of support rail
           WRAILBOT_$DYNVMLC(3): Width of bottom of support rail
               ZTIP_$DYNVMLC(3): Z at which tip at top of leaf begins
              ZLEAF_$DYNVMLC(3): Z of top of leaf
            ZTONGUE_$DYNVMLC(3): Z of top of tongue
            ZGROOVE_$DYNVMLC(3): Z of bottom of groove
           ZHOLETOP_$DYNVMLC(3): Z of top of driving screw hole
           ZHOLEBOT_$DYNVMLC(3): Z of bottom of driving screw hole
           HOLEPOS_ISO_$DYNVMLC: Distance of hole from leaf tip
           ZRAILTOP_$DYNVMLC(3): Z of top of support rail
           ZRAILBOT_$DYNVMLC(3): Z of bottom of support rail

     Note: Z positions are input in order of increasing Z.  Thus
           ZLEAF_$DYNVMLC(1)>=ZTIP_$DYNVMLC(1), etc.  See the BEAM
           manual or GUI help for restrictions on widths.

      Note: 1. For TARGET and ISOCENTER leaves to fit together,
               ZTONGUE_$DYNVMLC(3)>=ZGROOVE_$DYNVMLC(2) and
               ZTONGUE_$DYNVMLC(2)<=ZGROOVE_$DYNVMLC(3).
            2. For TARGET and FULL leaves to fit together (FULL
               leaf on -X [ORIENT=0] or -Y [ORIENT=1] side of TARGET
               leaf only) ZTONGUE_$DYNVMLC(2)<=ZGROOVE_$DYNVMLC(1)
            3. For ISOCENTER and FULL leaves to fit together (FULL
               leaf on +X [ORIENT=0] or +Y [ORIENT=1] side of ISOCENTER
               leaf only) ZTONGUE_$DYNVMLC(1)<=ZGROOVE_$DYNVMLC(3)
```

```
Repeat 8 NGROUP_$DYNVMLC times

8  NUM_LEAF_$DYNVMLC(I), LEAFTYPE  (2I5)

     NUM_LEAF_$DYNVMLC(I): Number of adjacent leaves in group I
                 LEAFTYPE: Type of leaf in group I.
                           Set to: 1 for FULL leaves
                                   2 for TARGET/ISOCENTER pair with
                                     TARGET leaf on the -X (ORIENT=0)
                                     or -Y (ORIENT=1) side

      Note: If LEAFTYPE is 2, then you must have an even number
            of leaves in the group.

9  START_$DYNVMLC (F15.0) : the start position (cm) wrt the CAX of
                    leaf 1 as projected to ZMIN_$DYNVMLC.

10 LEAFGAP_$DYNVMLC (F15.5) : The width of the interleaf air gap
          at ZMIN_$DYNVMLC.

         Note restriction: LEAFGAP_$DYNVMLC<=WTONGUE_$DYNVMLC(1,2,3),

11 ENDTYPE_$DYNVMLC (I5) : The  type of leaf end :
          0 -- rounded leaf end and
          1 -- focused divergent leaf end.

12 ZFOCUS_$DYNVMLC (F15.5) : Focal point on Z-axis of leaf ends
                  (i.e. imaginary lines drawn extending the slopes
                  of leaf ends will all intersect the Z-axis
                  at this point) - chosen if ENDTYPE_$DYNVMLC = 1.

     Note restriction: ZFOCUS_$DYNVMLC(1) < ZMIN_$DYNVMLC or
                                  > ZMIN_$DYNVMLC + ZTHICK_$DYNVMLC

   LEAFRADIUS_$DYNVMLC (F15.5) : Radius of the leaf end if
                       ENDTYPE_$DYNVMLC = 0. This must be greater
                       than or equal to half the leaf thickness.

13 ZFOCUS_$DYNVMLC(1) (F15.5): Focal point on Z-axis of leaf sides
                     imaginary lines drawn extending the slopes of
                     the leaf sides will all intersect the Z-axis
                     at this point)

     Note restriction: ZFOCUS_$DYNVMLC(1) < ZMIN_$DYNVMLC or
                             > ZMIN_$DYNVMLC + ZTHICK_$DYNVMLC

                    For focused ends the leaf position is defined
                    at ZMIN_$DYNVMLC; for rounded at ZMIN_$DYNVMLC +
                   0.5*ZTHICK_$DYNVMLC (ie center of the leaf in z)
```

```
    If MODE_$DYNVMLC=0 (static field):

   Repeat 14a until opening coordinates of all leaves are defined once.
   Leaves are numbered 1,2,...TOT_LEAF_$DYNVMLC, where numbering goes from
   leaf 1 to leaf TOT_LEAF_$DYNVMLC. Convention is lower to upper or
   left to right depending on ORIENT_$DYNVMLC i.e from negative to
   positive.  Note that for dynamic or step-and-shoot simulations, these
   are the default coordinates, used unless specified otherwise in the
   file of leaf opening data input in line 14a (see below).

14a  NEG_$DYNVMLC, POS_$DYNVMLC, NUM_$DYNVMLC (2F15.5,I5)

   NEG_$DYNVMLC:  Min. Y (ORIENT_$DYNVMLC=0) or X (ORIENT_$DYNVMLC=1)
                  of front opening in leaf I (ie the opening at
                  ZMIN_$DYNVMLC) if ENDTYPE=1, or of rounded end
                  of leaf I if ENDTYPE=0.
   POS_$DYNVMLC:  Max. Y (ORIENT_$DYNVMLC=0) or X (ORIENT_$DYNVMLC=1)
                  of front opening in leaf I if ENDTYPE=1, or of
                  rounded end of leaf I if ENDTYPE=0.
   NUM_$DYNVMLC:  Apply NEG_$DYNVMLC and POS_$DYNVMLC to leaves
                  I,...,I+NUM_$DYNVMLC-1.  Defaults to 1 if set <=0.
                  Defaults to TOT_LEAF_$DYNVMLC-I+1 if set >
                  TOT_LEAF_$DYNVMLC-I+1.

If MODE_$DYNVMLC=1 or 2 (dynamic delivery or step-and-shoot delivery):

14b  mlc_file (A256)

       mlc_file:  The full name of the file containing leaf opening
                  data.  The format of the file contents is as follows:

    MLC_TITLE (A80)
    NFIELDS_$DYNVMLC (I10)
    FOR I=1,NFIELDS_$DYNVMLC[
     INDEX_$DYNVMLC(I) (F15.0)
     NEG_$DYNVMLC, POS_$DYNVMLC, NUM_$DYNVMLC (2F15.0,I5) -- repeat this
                                                             line until
                                                             coordinates
                                                             for all leaves
                                                             have been
                                                             defined for
                                                             field I.
    ]

    where:

           MLC_TITLE:  A title line
    NFIELDS_$DYNVMLC:  Total number of fields
```

```
   INDEX_$DYNVMLC(I):  Index of field I. 0 <= INDEX_$DYNVMLC(I) <= 1 and
                       INDEX_$DYNVMLC(I) > INDEX_$DYNVMLC(I-1).  This
                       number is compared to a random number on (0,1) at
                       the start of each history; if the random number is
                       <= INDEX_$DYNVMLC(I), then field I is used.
        NEG_$DYNVMLC:  Min. Y (ORIENT_$DYNVMLC=0) or X (ORIENT_$DYNVMLC=1)
                       of front opening in leaf (ie the opening at
                       ZMIN_$DYNVMLC) if ENDTYPE=1, or of rounded end
                       of leaf if ENDTYPE=0 for leaf J in field I.
        POS_$DYNVMLC:  Max. Y (ORIENT_$DYNVMLC=0) or X (ORIENT_$DYNVMLC=1)
                       of front opening in leaf if ENDTYPE=1, or of
                       rounded end of leaf if ENDTYPE=0 for leaf J in
                       field I.
        NUM_$DYNVMLC:  Apply NEG_$DYNVMLC and POS_$DYNVMLC to leaves
                       J,...,J+NUM_$DYNVMLC-1.  Defaults to 1 if set <=0.
                       Defaults to TOT_LEAF_$DYNVMLC-J+1 if set >
                       TOT_LEAF_$DYNVMLC-J+1.

  Note that the inputs NEG_$DYNVMLC, POS_$DYNVMLC and NUM_$DYNVMLC have
  the same meanings as in 14a (static field inputs) but that they must
  now be repeated for every field I.

15   ECUT, PCUT, DOSE_ZONE, IREGION_TO_BIT in opening(s) and
                                          air gaps (2F15.5,2I5)

      ECUT, PCUT:  Cutoff energies for electrons and photons.
       DOSE_ZONE:   Dose scoring flag, 0 to not score dose
  IREGION_TO_BIT:   Bit number associated with this region

16  MED_IN (24A1):  Medium in opening(s) and air gaps
                    used to set MED_INDEX.

17  ECUT, PCUT, DOSE_ZONE, IREGION_TO_BIT in leaves, IGNOREGAPS_$DYNVMLC
                                                      (2F15.0,3I5):

      ECUT, PCUT:  Cutoff energies for electrons and photons.
       DOSE_ZONE:   Dose scoring flag, 0 to note score dose
  IREGION_TO_BIT:   Bit number associated with this region
      IGNOREGAPS:   If set to 1, ignore all air gaps and driving screw
                    holes when doing range
                    rejection in leaf material when the particle X position
                    is < min X of all leaf openings (not including leaf
                    ends) or > max X of leaf openings (not including ends)
                    (ORIENT_$DYNVMLC=1) or if the particle Y position
                    is < min Y of all leaf openings (not including leaf
                    ends) or > max Y of leaf openings (not including ends)
                    (ORIENT_$DYNVMLC=0). This approximation is designed
                    to make range rejection more efficient deep in the
                    leaves, while still preserving accurate transport
```

```
                    in the leaf ends.  Note that if you have significant
                    air gaps between leaves or are concerned with the
                    effects of the driving screw holes it is recommended
                    that you not use this option (ie run with the default
                    setting of 0).

 18  MED_IN (24A1):  Medium of leaves,
                    used to set MED_INDEX.

 19  ECUT, PCUT, DOSE_ZONE, IREGION_TO_BIT in driving screw holes
                                                        (2F15.5,2I5):

      ECUT, PCUT:  Cutoff energies for electrons and photons.
       DOSE_ZONE:   Dose scoring flag, 0 to note score dose
  IREGION_TO_BIT:   Bit number associated with this region

 20  MED_IN (24A1):  Medium in driving screw holes,
                    used to set MED_INDEX.

  Example
  *******

  The following example defines a multi-leaf tungsten collimator design
  based loosely on that used with the Varian Millenium MLC.
  Actual parameters are DIFFERENT - this serves just as a template.
  Do not attempt to use these parameters for a simulation of the real
  machine.

  The collimator starts at Z=48.25 cm and has 60 tungsten leaves opening
  in the X direction. Leaves 1-10 and 51-60 are FULL and leaves 11-50 are
  TARGET/ISOCENTER pairs.  The Z focus of the leaf sides is at Z=0 cm
  which is the position of the source. The leaf ends are rounded with a
   radius of 8 cm. In this example, leaf opening coordinates are chosen to
  create a square of width ~ 2cm centred on the beam axis.

  Electrons and photons in both the collimator and the opening regions
  will be followed down to kinetic energies of 189 keV (ECUT=0.7,
  PCUT=0.01).  Dose deposited in the tungsten leaves will be stored
  in dose zone 2, and dose deposited in the opening will be stored
  in dose zone 1.

20.5, RMAX
CL21X - Millenium MLC
1, 3, ORIENT, NGROUP
48.25, ZMIN
6.7, ZTHICK
0.5, 0.04, 0.04, 0.1354, 0.3252, 0.1227, 48.25, 48.533, 51.524, 51.732,
  52.98, 53.28, 2, 54.5474, 54.812, FULL leaf
0.25, 0.04, 0.04, 0.0929, 0.132, 0.132, 48.345, 48.6096, 49.5277,
```

```
  49.8277, 2, 51.625, 51.627, 54.7, 54.746, TARGET leaf
0.25, 0.04, 0.04, 0.0354, 0.1285, 0.1235, 48.412, 48.531, 51.631, 51.732,
  53.3293, 53.6293, 2, 54.5474, 54.812, ISOCENTER leaf
10, 1, FULL leaves
40, 2, TARGET/ISOCENTER pairs
10, 1, FULL leaves
-10.2, START
0.006, LEAFGAP
0, ENDTYPE
8, ZFOCUS or RADIUS of leaf ends
0, ZFOCUS of leaf sides
0, 0, 26
-1.0, 1.0, 8
0,0,26
0.7, 0.01, 1, 0,
AIR521ICRU
0.7, 0.01, 2, 0, 0,
W521ICRU
0.7, 0.01, 3, 0,
AIR521ICRU
```

15.3.20 SYNCVMLC

SYNCVMLC is a version of DYNVMLC which allows synchronization of leaf opening coordinates with any other synchronized CMs in the accelerator (SYNCJAWS, SYNCMLCE, SYNCHDMLC) and with the motion of sources 20 (synchronized phase space source) and 21 (synchronized BEAM treatment head source) in DOSXYZnrc. It was coded by Julio Lobo & Tony Popescu[24].

Inputs to SYNCVMLC are identical to DYNVMLC (See Section 15.3.19 above). As with DYNVMLC, SYNCVMLC can be used in static (`MODE`=0), dynamic (`MODE`=1) or step-and-shoot (`MODE`=2) modes. In dynamic and step-and-shoot modes, the leaf opening coordinates for the fields are read from a file supplied by the user. The input, `INDEX(I)`, defining the cumulative probability of a primary history being incident through all fields up to and including field `I` is now interpreted as `muIndex(I)`, the fraction of monitor units delivered up to and including field `I`. As with other synchronized CMs, for each primary history a random number `MU_RND`$\in$ [0,1] is used to determine the field number and, in dynamic mode, to calculate the leaf opening coordinates using Equation (9).

The same value of `MU_RND` is used for all synchronized CMs in the accelerator, allowing the user to synchronize opening coordinates between these CMs, and, in the case where the accelerator has been compiled as a shared library and is being used in source 20 (synchronized phase space) or source 21 (synchronized BEAM treatment head) in a DOSXYZnrc simulation, the value of `MU_RND` is also passed to the DOSXYZnrc simulation. This allows the user to synchronize the motion of the source plane with the opening coordinates of synchronized CMs. See the DOSXYZnrc Users Manual[15] for more details.

An example of a file specifying dynamic leaf settings for SYNCVMLC is included with the distribution. This file is `$OMEGA_HOME/beamnrc/CMs/sample_syncvmlc.sequence`.

15.3.21 SYNCHDMLC

SYNCHDMLC is a CM coded by Lobo & Popescu and based on an earlier code by Borges et al[52] for modeling the high-definition micro mlc (HD120) available on TrueBeam and Novalis linacs. In overall design it is similar to DYNVMLC/SYNCVMLC but features two different types of TARGET/ISOCENTER leaves: HALF TARGET/HALF ISOCENTER and QUARTER TARGET/QUARTER ISOCENTER. In addition, the last TARGET or ISOCENTER leaf in a group need not have a matching ISOCENTER or TARGET counterpart. In terms of leaf cross-section (perpendicular to the opening direction), these are similar to the TARGET/ISOCENTER leaves in DYNVMLC, with HALF TARGET/HALF ISOCENTER leaves being wider than the QUARTER TARGET/QUARTER ISOCENTER leaves.

The labels on the cross-section dimensions in Figure 38 have the same meanings as those used for DYNVMLC (See Section 15.3.19 and Table 3). In general, QUARTER leaves are used closer to middle of the mlc, and, while their Z-dimensions are similar to those for their HALF counterparts, they are significantly thinner in the X- (`ORIENT`=0) or Y- (`ORIENT`=1) dimension.

In general, SYNCHDMLC is similar to DYNVMLC and SYNCVMLC in that leaf cross-section dimensions must be defined so that the grid lines shown in Figure 38, defining the cross-section regions, remain in the same order shown in the figure. This should allow modeling of the HD120 based on manufacturer specifications assuming either non-divergent leaves (*i.e.,* leaves in which the vertical cross-section boundaries are parallel as opposed to intersecting at some value of Z<`ZMIN`), as shown in Figure 38, or else minimal divergence. Leaves may diverge to a significant extent, however, and it has come to our attention that, in an effort to more accurately model this, users may project manufacturer-specified cross-section widths (*e.g.,* `wtip`, `wts`, `wbs`, etc) from the Z values at which they are defined back to `ZMIN`. This results in a change in order in some of the vertical cross-section boundaries. In an effort to accommodate this in SYNCHDMLC, we have recently introduced the following relaxations of prior input restrictions:

1. **For HALF TARGET and QUARTER TARGET leaves**: The bottom of the support rail can extend to the left of the leaf tip (`wbs` > `wg` + `wtip`), changing the order of the 3rd and 4th (counting from the right) vertical cross-section boundaries shown in Figure 38.

2. **For HALF TARGET and QUARTER TARGET leaves**: The top of the support rail can extend past the right edge of the leaf (`wts` > `wbs`), changing the order of the 6th and 7th vertical cross-section boundaries.

3. **For QUARTER TARGET leaves**: The top of the support rail need not extend past the beginning (left edge) of the groove (`wts` < `wbs` - `wg`), effectively changing the order of vertical boundries 5 and 6, provided that, if relaxation 1 above is in effect, it does extend past the left edge of the leaf tip.

4. **For HALF ISOCENTER leaves**: The bottom of the support rail can extend past the left edge of the leaf (`wbs` > `wts`), changing the order of the 1st and 2nd vertical cross-section boundaries.

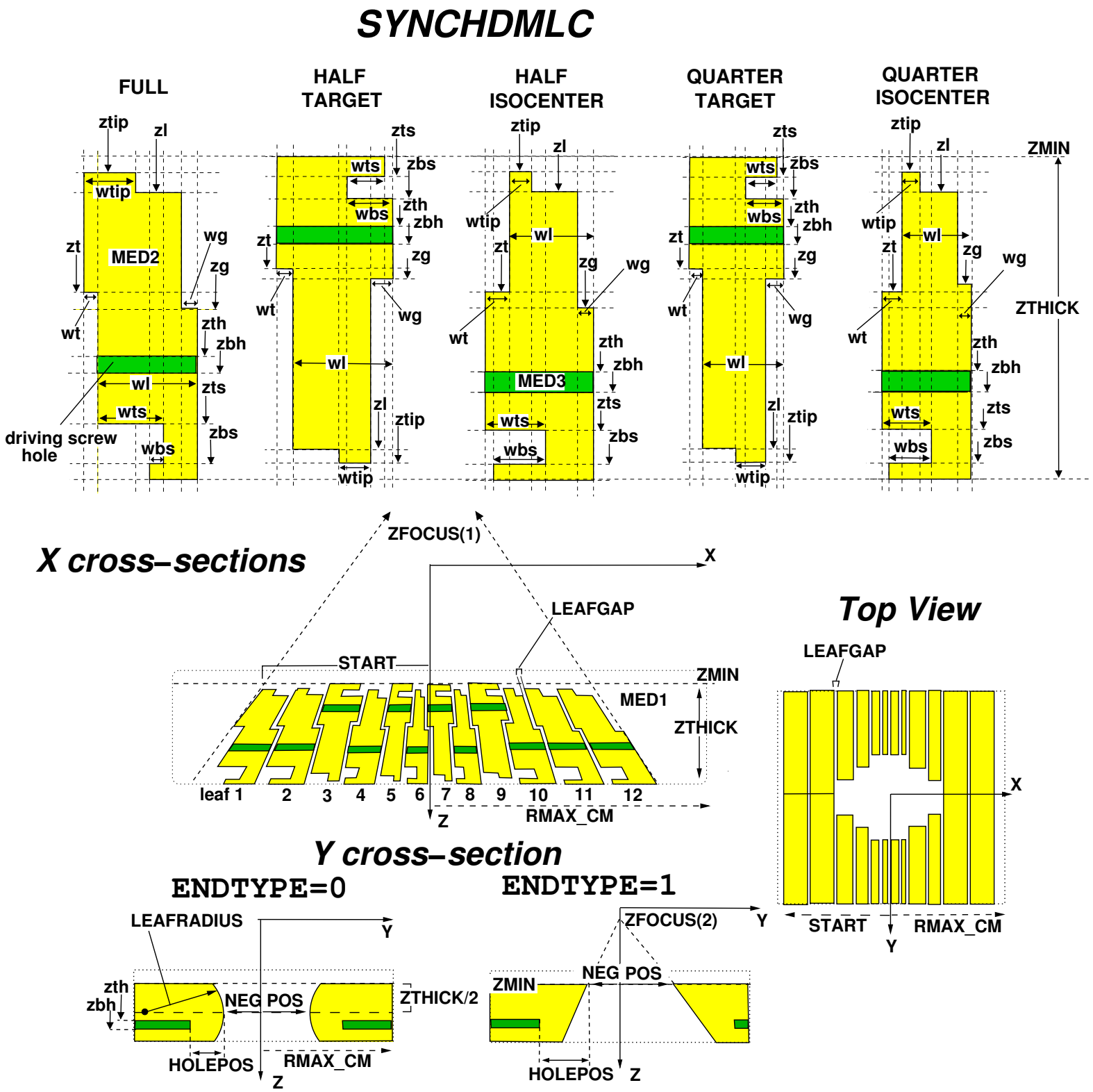


Figure 38: Example SYNCHDMLC with 12 leaves opening in the Y direction (`ORIENT`=0). The X cross-sections of the five leaf types show the dimensions that the user must input (see Table 3 for label key). QUARTER leaf cross-sections have similar Z-dimensions to HALF leaf cross-sections but are thinner in X. In this illustration leaves 1, 2, 11, 12 are FULL; leaves 3, 4, 9, 10 are HALF TARGET/HALF ISOCENTER leaves; leaves 5–8 are QUARTER TARGET/QUARTER ISOCENTER leaves. Unlike DYNVMLC and SYNCVMLC, an odd number of TARGET/ISOCENTER leaves may be specified in a group.

5. **For QUARTER ISOCENTER leaves**: The bottom of the support rail need not extend (to the right) as far as the right edge of the tongue (`wbs` < `wts` - `wt`), changing the order of the 2nd and 3rd vertical cross-section boundaries, provided that it still extends past the right edge of the leaf tip.

6. **For QUARTER ISOCENTER leaves**: The Z-position of the groove, `zg`, can be greater than that of the tongue, `zt`, allowing for duelling interpretations of manufacturer specifications for this leaf type in which the relative Z-positions of these structures are not entirely clear.

If user input dimensions require any of the above relaxations, a warning message will be output stating which restriction is being relaxed and indicating that the input cross-section differs from that shown in this manual. It is important to note that, while these relaxations in boundary ordering may allow the user to input accurate dimensions consistent with divergent MLC leaves, leaves are still modeled as if they are non-divergent, with all side surfaces focused at `ZFOCUS(1)`, and, thus, the geometry remains an approximation. Many thanks to Dr. Jarkko Ojala at Tampere University Hospital, Finland, who, through detailed examination of manufacturer specifications and measurement of MLC leaves, determined which restrictions needed to be relaxed and who helped test the modified code.

Similar to DYNVMLC, leaf cross-section dimensions must be defined so that the Z and X (`ORIENT`=0) or Y (`ORIENT`=1) grid lines shown on the cross-sections at the top of Figure 38 do not change order. The Z-positions of tongues and grooves must allow leaves to fit together in the following combinations:
FULL/FULL ($\texttt{zt}_{FULL} \leq \texttt{zg}_{FULL}$)
FULL/HALF TARGET ($\texttt{zg}_{FULL} \geq \texttt{zt}_{HALF\ TARGET}$)
HALF TARGET/HALF ISOCENTER ($\texttt{zg}_{HALF\ TARGET} \leq \texttt{zt}_{HALF\ ISOCENTER}$)
HALF ISOCENTER/HALF TARGET ($\texttt{zg}_{HALF\ ISOCENTER} \geq \texttt{zt}_{HALF\ TARGET}$)
HALF ISOCENTER/QUARTER TARGET ($\texttt{zg}_{HALF\ ISOCENTER} \geq \texttt{zt}_{QUARTER\ TARGET}$)
QUARTER TARGET/QUARTER ISOCENTER ($\texttt{zg}_{QUARTER\ TARGET} \leq \texttt{zt}_{QUARTER\ ISOCENTER}$)
QUARTER ISOCENTER/QUARTER TARGET ($\texttt{zg}_{QUARTER\ ISOCENTER} \geq \texttt{zt}_{QUARTER\ TARGET}$)
QUARTER ISOCENTER/HALF TARGET ($\texttt{zg}_{QUARTER\ ISOCENTER} \geq \texttt{zt}_{HALF\ TARGET}$)
and HALF ISOCENTER/FULL (ie $\texttt{zg}_{HALF\ ISOCENTER} \geq \texttt{zt}_{FULL}$).

As in DYNVMLC and SYNCVMLC, rather than specifying the type of each leaf in the MLC, the user is able to specify groups of adjacent leaves of the same type. Since TARGET leaves always alternate with ISOCENTER leaves within a group, for the purpose of defining a leaf group, HALF or QUARTER TARGET/ISOCENTER leaves are considered one type. Unlike DYNVMLC and SYNCVMLC, however, the number of TARGET/ISOCENTER leaves in a group does not have to be even. Thus, the final TARGET or ISOCENTER leaf in a group need not be matched by its ISOCENTER or TARGET (respectively) counterpart. If a group of TARGET/ISOCENTER is defined adjacent to another group of TARGET/ISOCENTER leaves, the first leaf in the group will automatically be chosen to match the last leaf of the previous group. Thus, if the last leaf of the previous group was a QUARTER or HALF TARGET leaf, then the first leaf of the current group will be a QUARTER or HALF ISOCENTER leaf, and vice versa. In terms of leaf groups, the only restriction is that the numbers and types of leaves cannot be specified so that a QUARTER or HALF TARGET leaf ends up immediately adjacent to a FULL leaf on the -X

(ORIENT=0) or -Y (ORIENT=1) side. Finally, note that the user need not use all leaf types in a simulated MLC. Thus, there may be no FULL leaves, HALF TARGET/ISOCENTER leaves, or QUARTER TARGET/ISOCENTER leaves in the simulation. In this case, inputs for the cross-section parameters of the unused leaves can be left as blanks or zeros.

Similar to DYNVMLC and SYNCVMLC, leaf ends can be straight, focused (`ENDTYPE`=1) or cylindrical (`ENDTYPE`=0). For straight, focused ends, the user must specify the Z-position of the focal point of the ends (`ZFOCUS(2)`). For cylindrical leaf ends, cylinders defining the leaf ends all have their origin at Z=`ZMIN`+`ZTHICK/2` (*i.e.* mid-way through the thickness of the MLC) and user-specified radius, `LEAFRADIUS`. Positive and negative leaf opening coordinates, `POS` and `NEG` are defined at Z=`ZMIN` for straight, focused leaf ends and at Z=`ZMIN`+`ZTHICK/2` for cylindrical leaf ends.

If SYNCHDMLC is used in dynamic (`MODE`=1) or step-and-shoot (`MODE`=2) mode, then leaf opening coordinates are read from a user-supplied file. The format of the file is identical to that used for DYNVMLC (and SYNCVMLC) and is given in Section 15.3.19 above. Similar to the other synchronized CMs, the field indices, `INDEX(I)` are re-interpreted as `muIndex(I)`, the fraction of monitor units delivered up to and including field `I`. For each primary history, a random number `MU_RND`$\in$ [0,1] is compared to the `muIndex(I)` and is used to choose the field to use and, in the case of dynamic mode, to calculate the opening coordinates using Equation (9).

The random number, `MU_RND`, is common for all synchronized CMs in the accelerator. Thus, the leaf settings for SYNCHDMLC can be synchronized with the settings of any other synchronized CMs. In addition, if a BEAM accelerator with SYNCHDMLC (and any other synchronized CMs) is compiled as a shared library for use in DOSXYZnrc source 20 (synchronized phase space) or source 21 (synchronized BEAM treatment head), then the value of `MU_RND` used in the BEAM simulation is passed to DOSXYZnrc. This allows the motion of the source plane in DOSXYZnrc to be synchronized with the opening coordinates in BEAMnrc. Note the user must define the coordination of source motion with CM opening coordinates. See the DOSXYZnrc Users Manual[15] for more details.

Similar to DYNVMLC, SYNCHDMLC has an `IGNOREGAPS` input parameter which, if set to 1, instructs the code to ignore air gaps, driving screw holes and side boundaries of adjacent leaves when determining the perpendicular distance to the nearest region boundary for charged particle range rejection. For this option to be in effect, charged particles must be in the leaf medium and have:
particle Y (`ORIENT`=0) or X (`ORIENT`=1) < minimum Y (`ORIENT`=0) or X (`ORIENT`=1) of all leaf openings (excluding rounded or focused leaf end) or
particle Y (`ORIENT`=0) or X (`ORIENT`=1) > maximum Y (`ORIENT`=0) or X (`ORIENT`=1) of all leaf openings (excluding leaf end)

An example of a file specifying dynamic leaf settings for SYNCHDMLC is included with the distribution. This file is `$OMEGA_HOME/beamnrc/CMs/sample_synchdmlc.sequence`.

A description of the input format for SYNCHDMLC is given below.

```
CARDS CM_$SYNCHDMLC
********************
```

```
-1  Dummy line to indicate start of CM

 0  RMAX_CM(ICM_$SYNCHDMLC) (F10.5):  Half-width of CM boundary (cm).

 1  TITLE_$SYNCHDMLC (60A1):  Title of CM.

 2  ORIENT_$SYNCHDMLC, NGROUP_$SYNCHDMLC (2I5)

          ORIENT_$SYNCHDMLC = 0 for leaves parallel to Y direction
                            = 1 for leaves parallel to X direction
          NGROUP_$SYNCHDMLC = number of groups of adjacent leaves where
                              all leaves in a group are:
                              1. FULL leaves
                              2. HALF TARGET/ISOCENTER leaves
                              3. QUARTER TARGET/ISOCENTER leaves
                              NGROUP_$SYNCHDMLC defaults to 3 if set <=0
          MODE_$SYNCHCMLC = 0 for single setting of leaf openings (static
                            field)
                          = 1 for dynamic mlc delivery--simulated leaf
                            movement while beam is on
                          = 2 for step-and-shoot delivery--beam off while
                            leaf positions change


 3  ZMIN_$SYNCHDMLC (F15.0): Z of top of MLC (excluding airgap)

 4  ZTHICK_$SYNCHDMLC (F15.0): Thickness of the leaves ( z-axis (cm))

 5  LEAFWIDTH_$SYNCHDMLC(1), WTONGUE_$SYNCHDMLC(1), WGROOVE_$SYNCHDMLC(1),
    WTIP_$SYNCHDMLC(1), WRAILTOP_$SYNCHDMLC(1), WRAILBOT_$SYNCHDMLC(1),
    ZTIP_$SYNCHDMLC(1), ZLEAF_$SYNCHDMLC(1), ZTONGUE_$SYNCHDMLC(1),
    ZGROOVE_$SYNCHDMLC(1), ZHOLETOP_$SYNCHDMLC(1), ZHOLEBOT_$SYNCHDMLC(1),
    HOLEPOS_FULL_$SYNCHDMLC, ZRAILTOP_$SYNCHDMLC(1), ZRAILBOT_$SYNCHDMLC(1)
       (15F15.0)

         For a FULL type leaf (all dimensions in cm--all widths are
                               projected back to ZMIN_$SYNCHDMLC):

            LEAFWIDTH_$SYNCHDMLC(1): Width of leaf (not including tongue)
              WTONGUE_$SYNCHDMLC(1): Width of tongue
              WGROOVE_$SYNCHDMLC(1): Width of groove
                 WTIP_$SYNCHDMLC(1): Width of tip at top of leaf
             WRAILTOP_$SYNCHDMLC(1): Width of top of support rail
             WRAILBOT_$SYNCHDMLC(1): Width of bottom of support rail
                 ZTIP_$SYNCHDMLC(1): Z at which tip at top of leaf begins
                ZLEAF_$SYNCHDMLC(1): Z of top of leaf
              ZTONGUE_$SYNCHDMLC(1): Z of bottom of tongue
              ZGROOVE_$SYNCHDMLC(1): Z of bottom of groove
             ZHOLETOP_$SYNCHDMLC(1): Z of top of driving screw hole
```

```
        ZHOLEBOT_$SYNCHDMLC(1): Z of bottom of driving screw hole
       HOLEPOS_FULL_$SYNCHDMLC: Distance of hole from leaf tip
        ZRAILTOP_$SYNCHDMLC(1): Z of top of support rail
        ZRAILBOT_$SYNCHDMLC(1): Z of bottom of support rail

    Note: Z positions are input in order of increasing Z.  Thus
          ZLEAF_$SYNCHDMLC(1)>=ZTIP_$SYNCHDMLC(1), etc.  See the BEAM
          manual or GUI help for restrictions on widths.

    Note: If there are no FULL leaves in your model, you can input
          blanks, zeroes, or arbitrary floating-point numbers for these
          cross section dimensions.


6 LEAFWIDTH_$SYNCHDMLC(2), WTONGUE_$SYNCHDMLC(2), WGROOVE_$SYNCHDMLC(2),
  WTIP_$SYNCHDMLC(2), WRAILTOP_$SYNCHDMLC(2), WRAILBOT_$SYNCHDMLC(2),
  ZRAILTOP_$SYNCHDMLC(2), ZRAILBOT_$SYNCHDMLC(2), ZHOLETOP_$SYNCHDMLC(2),
  ZHOLEBOT_$SYNCHDMLC(2), HOLEPOS_TAR_$SYNCHDMLC, ZTONGUE_$SYNCHDMLC(2),
  ZGROOVE_$SYNCHDMLC(2),ZLEAF_$SYNCHDMLC(2), ZTIP_$SYNCHDMLC(2) (15F15.0)

     For a HALF TARGET type leaf (all dimensions in cm--all widths are
                          projected back to ZMIN_$SYNCHDMLC):

       LEAFWIDTH_$SYNCHDMLC(2): Width of leaf (not including tongue)
         WTONGUE_$SYNCHDMLC(2): Width of tongue
         WGROOVE_$SYNCHDMLC(2): Width of groove
            WTIP_$SYNCHDMLC(2): Width of tip at bottom of leaf
        WRAILTOP_$SYNCHDMLC(2): Width of top of support rail
        WRAILBOT_$SYNCHDMLC(2): Width of bottom of support rail
        ZRAILTOP_$SYNCHDMLC(2): Z of top of support rail
        ZRAILBOT_$SYNCHDMLC(2): Z of bottom of support rail
        ZHOLETOP_$SYNCHDMLC(2): Z of top of driving screw hole
        ZHOLEBOT_$SYNCHDMLC(2): Z of bottom of driving screw hole
        HOLEPOS_TAR_$SYNCHDMLC: Distance of hole from leaf tip
         ZTONGUE_$SYNCHDMLC(2): Z of bottom of tongue
         ZGROOVE_$SYNCHDMLC(2): Z of top of groove
           ZLEAF_$SYNCHDMLC(2): Z of bottom of leaf
            ZTIP_$SYNCHDMLC(2): Z of bottom of tip at bottom of leaf

    Note: Z positions are input in order of increasing Z.  Thus
          ZLEAF_$SYNCHDMLC(1)>=ZTIP_$SYNCHDMLC(1), etc.  See the BEAM
          manual or GUI help for restrictions on widths.


7 LEAFWIDTH_$SYNCHDMLC(3), WTONGUE_$SYNCHDMLC(3), WGROOVE_$SYNCHDMLC(3),
  WTIP_$SYNCHDMLC(3), WRAILTOP_$SYNCHDMLC(3), WRAILBOT_$SYNCHDMLC(3),
  ZTIP_$SYNCHDMLC(3), ZLEAF_$SYNCHDMLC(3), ZTONGUE_$SYNCHDMLC(3),
  ZGROOVE_$SYNCHDMLC(3), ZHOLETOP_$SYNCHDMLC(3), ZHOLEBOT_$SYNCHDMLC(3),
  HOLEPOS_ISO_$SYNCHDMLC, ZRAILTOP_$SYNCHDMLC(3), ZRAILBOT_$SYNCHDMLC(3)
```

```
              (15F15.0)

      For a HALF ISOCENTER type leaf (all dimensions in cm--all widths are
                          projected back to ZMIN_$SYNCHDMLC):

         LEAFWIDTH_$SYNCHDMLC(3): Width of leaf (not including tongue)
           WTONGUE_$SYNCHDMLC(3): Width of tongue
           WGROOVE_$SYNCHDMLC(3): Width of groove
              WTIP_$SYNCHDMLC(3): Width of tip at top of leaf
          WRAILTOP_$SYNCHDMLC(3): Width of top of support rail
          WRAILBOT_$SYNCHDMLC(3): Width of bottom of support rail
              ZTIP_$SYNCHDMLC(3): Z at which tip at top of leaf begins
             ZLEAF_$SYNCHDMLC(3): Z of top of leaf
           ZTONGUE_$SYNCHDMLC(3): Z of top of tongue
           ZGROOVE_$SYNCHDMLC(3): Z of bottom of groove
          ZHOLETOP_$SYNCHDMLC(3): Z of top of driving screw hole
          ZHOLEBOT_$SYNCHDMLC(3): Z of bottom of driving screw hole
          HOLEPOS_ISO_$SYNCHDMLC: Distance of hole from leaf tip
          ZRAILTOP_$SYNCHDMLC(3): Z of top of support rail
          ZRAILBOT_$SYNCHDMLC(3): Z of bottom of support rail

       Note: Z positions are input in order of increasing Z.  Thus
             ZLEAF_$SYNCHDMLC(1)>=ZTIP_$SYNCHDMLC(1), etc.  See the BEAM
             manual or GUI help for restrictions on widths.

        Note: 1. For TARGET and ISOCENTER leaves to fit together,
                 ZTONGUE_$SYNCHDMLC(3)>=ZGROOVE_$SYNCHDMLC(2) and
                 ZTONGUE_$SYNCHDMLC(2)<=ZGROOVE_$SYNCHDMLC(3).
              2. For TARGET and FULL leaves to fit together (FULL
                 leaf on -X [ORIENT=0] or -Y [ORIENT=1] side of TARGET
                 leaf only) ZTONGUE_$SYNCHDMLC(2)<=ZGROOVE_$SYNCHDMLC(1)
              3. For ISOCENTER and FULL leaves to fit together (FULL
                 leaf on +X [ORIENT=0] or +Y [ORIENT=1] side of ISOCENTER
                 leaf only) ZTONGUE_$SYNCHDMLC(1)<=ZGROOVE_$SYNCHDMLC(3)

       Note: If there are no HALF leaves in your model, you can input
             blanks, zeroes, or arbitrary floating-point numbers for these
             cross section dimensions.

   8  LEAFWIDTH_$SYNCHDMLC(4), WTONGUE_$SYNCHDMLC(4), WGROOVE_$SYNCHDMLC(4),
      WTIP_$SYNCHDMLC(4), WRAILTOP_$SYNCHDMLC(4), WRAILBOT_$SYNCHDMLC(4),
      ZRAILTOP_$SYNCHDMLC(4), ZRAILBOT_$SYNCHDMLC(4), ZHOLETOP_$SYNCHDMLC(4),
      ZHOLEBOT_$SYNCHDMLC(4), HOLEPOS_QTAR_$SYNCHDMLC, ZTONGUE_$SYNCHDMLC(4),
      ZGROOVE_$SYNCHDMLC(4), ZLEAF_$SYNCHDMLC(4),ZTIP_$SYNCHDMLC(4) (15F15.0)

        For a QUARTER TARGET type leaf (all dimensions in cm--all widths are
                          projected back to ZMIN_$SYNCHDMLC):

           LEAFWIDTH_$SYNCHDMLC(4): Width of leaf (not including tongue)
```

```
          WTONGUE_$SYNCHDMLC(4): Width of tongue
          WGROOVE_$SYNCHDMLC(4): Width of groove
             WTIP_$SYNCHDMLC(4): Width of tip at bottom of leaf
        WRAILTOP_$SYNCHDMLC(4): Width of top of support rail
        WRAILBOT_$SYNCHDMLC(4): Width of bottom of support rail
        ZRAILTOP_$SYNCHDMLC(4): Z of top of support rail
        ZRAILBOT_$SYNCHDMLC(4): Z of bottom of support rail
        ZHOLETOP_$SYNCHDMLC(4): Z of top of driving screw hole
        ZHOLEBOT_$SYNCHDMLC(4): Z of bottom of driving screw hole
        HOLEPOS_QTAR_$SYNCHDMLC: Distance of hole from leaf tip
          ZTONGUE_$SYNCHDMLC(4): Z of bottom of tongue
          ZGROOVE_$SYNCHDMLC(4): Z of top of groove
            ZLEAF_$SYNCHDMLC(4): Z of bottom of leaf
             ZTIP_$SYNCHDMLC(4): Z of bottom of tip at bottom of leaf

    Note: Z positions are input in order of increasing Z.  Thus
          ZLEAF_$SYNCHDMLC(4)>=ZTIP_$SYNCHDMLC(4), etc.  See the BEAM
          manual or GUI help for restrictions on widths.


9  LEAFWIDTH_$SYNCHDMLC(5), WTONGUE_$SYNCHDMLC(5), WGROOVE_$SYNCHDMLC(5),
   WTIP_$SYNCHDMLC(5), WRAILTOP_$SYNCHDMLC(5), WRAILBOT_$SYNCHDMLC(5),
   ZTIP_$SYNCHDMLC(5), ZLEAF_$SYNCHDMLC(5), ZTONGUE_$SYNCHDMLC(5),
   ZGROOVE_$SYNCHDMLC(5), ZHOLETOP_$SYNCHDMLC(5), ZHOLEBOT_$SYNCHDMLC(5),
   HOLEPOS_QISO_$SYNCHDMLC, ZRAILTOP_$SYNCHDMLC(5), ZRAILBOT_$SYNCHDMLC(5)
               (15F15.0)

 For a QUARTER ISOCENTER type leaf (all dimensions in cm--all widths are
                           projected back to ZMIN_$SYNCHDMLC):

         LEAFWIDTH_$SYNCHDMLC(5): Width of leaf (not including tongue)
           WTONGUE_$SYNCHDMLC(5): Width of tongue
           WGROOVE_$SYNCHDMLC(5): Width of groove
              WTIP_$SYNCHDMLC(5): Width of tip at top of leaf
         WRAILTOP_$SYNCHDMLC(5): Width of top of support rail
         WRAILBOT_$SYNCHDMLC(5): Width of bottom of support rail
             ZTIP_$SYNCHDMLC(5): Z at which tip at top of leaf begins
            ZLEAF_$SYNCHDMLC(5): Z of top of leaf
          ZTONGUE_$SYNCHDMLC(5): Z of top of tongue
          ZGROOVE_$SYNCHDMLC(5): Z of bottom of groove
         ZHOLETOP_$SYNCHDMLC(5): Z of top of driving screw hole
         ZHOLEBOT_$SYNCHDMLC(5): Z of bottom of driving screw hole
         HOLEPOS_QISO_$SYNCHDMLC: Distance of hole from leaf tip
         ZRAILTOP_$SYNCHDMLC(5): Z of top of support rail
         ZRAILBOT_$SYNCHDMLC(5): Z of bottom of support rail

    Note: If there are no QUARTER leaves in your model, you can input
          blanks, zeroes, or arbitrary floating-point numbers for these
          cross section dimensions.
```

```
Repeat 10 NGROUP_$SYNCHDMLC times

10  NUM_LEAF_$SYNCHDMLC(I), LEAFTYPE  (2I5)

     NUM_LEAF_$SYNCHDMLC(I): Number of adjacent leaves in group I
          LEAFTYPE: Type of leaf in group I.
                    Set to: 1 for FULL leaves
                            2 for HALF TARGET/ISOCENTER leaves.
                              Starting leaf on -X (ORIENT=0) or -Y
                              (ORIENT=1) side depends on adjacent leaf:
                              If adjacecent leaf is a HALF or QUARTER
                              TARGET, then it will be HALF ISOCENTER.
                              Otherwise, it will be HALF TARGET.
                            3 for QUARTER TARGET/ISOCENTER leaves.
                              Starting leaf on -X (ORIENT=0) or -Y
                              (ORIENT=1) side will depend on the
                              adjacent leaf: If it is a HALF or
                              QUARTER TARGET leaf, then the starting
                              leaf will be QUARTER ISOCENTER.
                              Otherwise, it will be QUARTER TARGET.

Restrictions: 1. Cannot have a HALF or QUARTER TARGET leaf on the -ve
                 side of a FULL leaf.

9  START_$SYNCHDMLC (F15.0) : the start position (cm) wrt the CAX of
                   leaf 1 as projected to ZMIN_$SYNCHDMLC.

10  LEAFGAP_$SYNCHDMLC (F15.5) : The width of the interleaf air gap
          at ZMIN_$SYNCHDMLC.

         Note restriction: LEAFGAP_$SYNCHDMLC<=WTONGUE_$SYNCHDMLC(1,2,3),

11  ENDTYPE_$SYNCHDMLC (I5) : The  type of leaf end :
          0 -- rounded leaf end and
          1 -- focused divergent leaf end.

12  ZFOCUS_$SYNCHDMLC (F15.5) : Focal point on Z-axis of leaf ends
                  (i.e. imaginary lines drawn extending the slopes
                  of leaf ends will all intersect the Z-axis
                  at this point) - chosen if ENDTYPE_$SYNCHDMLC = 1.

     Note restriction: ZFOCUS_$SYNCHDMLC(1) < ZMIN_$SYNCHDMLC or
                                  > ZMIN_$SYNCHDMLC + ZTHICK_$SYNCHDMLC

   LEAFRADIUS_$SYNCHDMLC (F15.5) : Radius of the leaf end if
                       ENDTYPE_$SYNCHDMLC = 0. This must be greater
                       than or equal to half the leaf thickness.
```

```
13  ZFOCUS_$SYNCHDMLC(1) (F15.5): Focal point on Z-axis of leaf sides
                      imaginary lines drawn extending the slopes of
                      the leaf sides will all intersect the Z-axis
                      at this point)

      Note restriction: ZFOCUS_$SYNCHDMLC(1) < ZMIN_$SYNCHDMLC or
                             > ZMIN_$SYNCHDMLC + ZTHICK_$SYNCHDMLC

                    For focused ends the leaf position is defined
                    at ZMIN_$SYNCHDMLC; for rounded at ZMIN_$SYNCHDMLC +
                  0.5*ZTHICK_$SYNCHDMLC (ie center of the leaf in z)

 Repeat 14 until coordinates of all leaves are defined once.  Leaves
 are numbered 1,2,...TOT_LEAF_$SYNCHDMLC, where numbering goes from leaf
 1 to leaf TOT_LEAF_$SYNCHDMLC. Convention is lower to upper or
 left to right depending on ORIENT_$SYNCHDMLC i.e from negative to
 positive.

14a  NEG_$SYNCHDMLC, POS_$SYNCHDMLC, NUM_$SYNCHDMLC (2F15.5,I5)

  NEG_$SYNCHDMLC:   Min. Y (ORIENT_$SYNCHDMLC=0) or X (ORIENT_$SYNCHDMLC=1)
                 of front opening in leaf I (ie the opening at
                 ZMIN_$SYNCHDMLC) if ENDTYPE=1, or of rounded end
                 of leaf I if ENDTYPE=0.
  POS_$SYNCHDMLC:   Max. Y (ORIENT_$SYNCHDMLC=0) or X (ORIENT_$SYNCHDMLC=1)
                 of front opening in leaf I if ENDTYPE=1, or of
                 rounded end of leaf I if ENDTYPE=0.
  NUM_$SYNCHDMLC:   Apply NEG_$SYNCHDMLC and POS_$SYNCHDMLC to leaves
                 I,...,I+NUM_$SYNCHDMLC-1.  Defaults to 1 if set <=0.
                 Defaults to TOT_LEAF_$SYNCHDMLC-I+1 if set >
                 TOT_LEAF_$SYNCHDMLC-I+1.

If MODE_$SYNCHDMLC=1 or 2 (dynamic delivery or step-and-shoot delivery):

14b  mlc_file (A256)

       mlc_file:  The full name of the file containing leaf opening
                  data.  The format of the file contents is as follows:

    MLC_TITLE (A80)
    NFIELDS_$SYNCHDMLC (I10)
    FOR I=1,NFIELDS_$SYNCHDMLC[
     INDEX_$SYNCHDMLC(I) (F15.0)
     NEG_$SYNCHDMLC, POS_$SYNCHDMLC, NUM_$SYNCHDMLC (2F15.0,I5) -- repeat
                                                                  line until
                                                                  coordinates
                                                                  for all leaves
                                                                  have been
                                                                  defined for
```

```
                                                              field I.
    ]

    where:

           MLC_TITLE:  A title line
    NFIELDS_$SYNCHDMLC:  Total number of fields
   INDEX_$SYNCHDMLC(I):  Index of field I. 0 <=INDEX_$SYNCHDMLC(I) <= 1 and
                         INDEX_$SYNCHDMLC(I) > INDEX_$SYNCHDMLC(I-1).  This
                         number is compared to a random number on (0,1) at
                         the start of each history; if the random number is
                         <= INDEX_$SYNCHDMLC(I), then field I is used.
        NEG_$SYNCHDMLC:  Min. Y (ORIENT_$SYNCHDMLC=0) or
                         X (ORIENT_$SYNCHDMLC=1)
                         of front opening in leaf (ie the opening at
                         ZMIN_$SYNCHDMLC) if ENDTYPE=1, or of rounded end
                         of leaf if ENDTYPE=0 for leaf J in field I.
        POS_$SYNCHDMLC:  Max. Y (ORIENT_$SYNCHDMLC=0) or
                         X (ORIENT_$SYNCHDMLC=1)
                         of front opening in leaf if ENDTYPE=1, or of
                         rounded end of leaf if ENDTYPE=0 for leaf J in
                         field I.
        NUM_$SYNCHDMLC:  Apply NEG_$SYNCHDMLC and POS_$SYNCHDMLC to leaves
                         J,...,J+NUM_$SYNCHDMLC-1.  Defaults to 1 if set <=0.
                         Defaults to TOT_LEAF_$SYNCHDMLC-J+1 if set >
                         TOT_LEAF_$SYNCHDMLC-J+1.

Note that the inputs NEG_$SYNCHDMLC, POS_$SYNCHDMLC and NUM_$SYNCHDMLC have
  the same meanings as in 14a (static field inputs) but that they must
  now be repeated for every field I.

15   ECUT, PCUT, DOSE_ZONE, IREGION_TO_BIT in opening(s) and
                                          air gaps (2F15.5,2I5)

      ECUT, PCUT:  Cutoff energies for electrons and photons.
       DOSE_ZONE:   Dose scoring flag, 0 to not score dose
  IREGION_TO_BIT:   Bit number associated with this region

16  MED_IN (24A1):  Medium in opening(s) and air gaps
                    used to set MED_INDEX.

17  ECUT, PCUT, DOSE_ZONE, IREGION_TO_BIT in leaves, IGNOREGAPS_$SYNCHDMLC
                                                      (2F15.0,3I5):

      ECUT, PCUT:  Cutoff energies for electrons and photons.
       DOSE_ZONE:   Dose scoring flag, 0 to note score dose
  IREGION_TO_BIT:   Bit number associated with this region
      IGNOREGAPS:   If set to 1, ignore all air gaps and driving screw
                    holes when doing range
```

```
                    rejection in leaf material when the particle X position
                    is < min X of all leaf openings (not including leaf
                    ends) or > max X of leaf openings (not including ends)
                    (ORIENT_$SYNCHDMLC=1) or if the particle Y position
                    is < min Y of all leaf openings (not including leaf
                    ends) or > max Y of leaf openings (not including ends)
                    (ORIENT_$SYNCHDMLC=0). This approximation is designed
                    to make range rejection more efficient deep in the
                    leaves, while still preserving accurate transport
                    in the leaf ends.  Note that if you have significant
                    air gaps between leaves or are concerned with the
                    effects of the driving screw holes it is recommended
                    that you not use this option (ie run with the default
                    setting of 0).

18  MED_IN (24A1):  Medium of leaves,
                    used to set MED_INDEX.

19  ECUT, PCUT, DOSE_ZONE, IREGION_TO_BIT in driving screw holes
                                                        (2F15.5,2I5):

      ECUT, PCUT:  Cutoff energies for electrons and photons.
       DOSE_ZONE:   Dose scoring flag, 0 to note score dose
  IREGION_TO_BIT:   Bit number associated with this region

20  MED_IN (24A1):  Medium in driving screw holes,
                    used to set MED_INDEX.

 Example
 *******

 The following example defines a multi-leaf tungsten collimator design
 based loosely on the TrueBeam high-definition mlc (HD120) but using
 all leaf types.
 Actual parameters are DIFFERENT - this serves just as a template.
 Do not attempt to use these parameters for a simulation of the real
 machine.

 The collimator starts at Z=47.0 cm, is 7 cm thick and has 80 tungsten
 leaves opening in the X direction. Leaves 1-10 and 71-80 are FULL
 (type 1), leaves 11-24 and 57-70 are HALF TARGET/HALF ISOCENTER
 leaves, and leaves 25-56 are QUARTER TARGET/QUARTER ISOCENTER leaves.
 The Z focus of the leaf sides is at Z=0 cm which is the position of the
 source. The leaf ends are straight and also focused at Z=0 cm.
 In this example, the mlc is being run in step-and-shoot mode
 MODE=2 with the leaf opening coordinates read in from the file
 specified

20.5, RMAX
```

```
TrueBeam HDMLC
1, 5, 2, ORIENT, NGROUP, MODE
47.0, ZMIN
7.0, ZTHICK
0.5, 0.04, 0.04, 0.135, 0.325, 0.123, 47, 47.5, 49.5, 50.7, 51.0, 51.3,
  1.7, 52.5, 52.8, FULL leaf cross-section
0.25, 0.04, 0.04, 0.093, 0.132, 0.132, 47, 47.4, 49.1, 49.4, 1.7, 50.5,
  50.5, 53.6, 53.7, HALF TARGET cross-section
0.25, 0.04, 0.04, 0.075, 0.124, 0.123, 47.2, 47.4, 50.52, 50.521, 51.7,
  52.1, 1.7, 53.6, 53.8, HALF ISOCENTER cross-section
0.11, 0.04, 0.04, 0.044, 0.074, 0.083, 47.1, 47.3, 47.9, 48.1, 1.7, 50.5,
  50.5, 53.6, 53.7, QUARTER TARGET cross-section
0.11, 0.04, 0.04, 0.035, 0.08, 0.065, 47.2, 47.3, 50.53, 50.52, 52.9, 53.1,
  1.7, 53.6, 53.9,  QUARTER ISOCENTER cross-section
10, 1, FULL leaves
14, 2, HALF TARGET/ISOCENTER leaves
32, 3, QUARTER TARGET/ISOCENTER leaves
14, 2, HALF TARGET/ISOCENTER leaves
10, 1, FULL leaves
-10.5, START
0.005, LEAFGAP
1, ENDTYPE
0.0, ZFOCUS or RADIUS of leaf ends
0.0, ZFOCUS of leaf sides
/home/bwalters/egsnrc_mp/BEAM_synchdmlc/sample_synchdmlc.sequence
0.7, 0.01, 1, 0,
AIR700ICRU
0.7, 0.01, 1, 0, 0,
W700ICRU
0.7, 0.01, 1, 0,
AIR700ICRU
```

15.3.22 MESH

The MESH CM models a single-layer wire mesh placed perpendicular to the beam direction in the path of the beam. Details are shown in Figure 39 below.

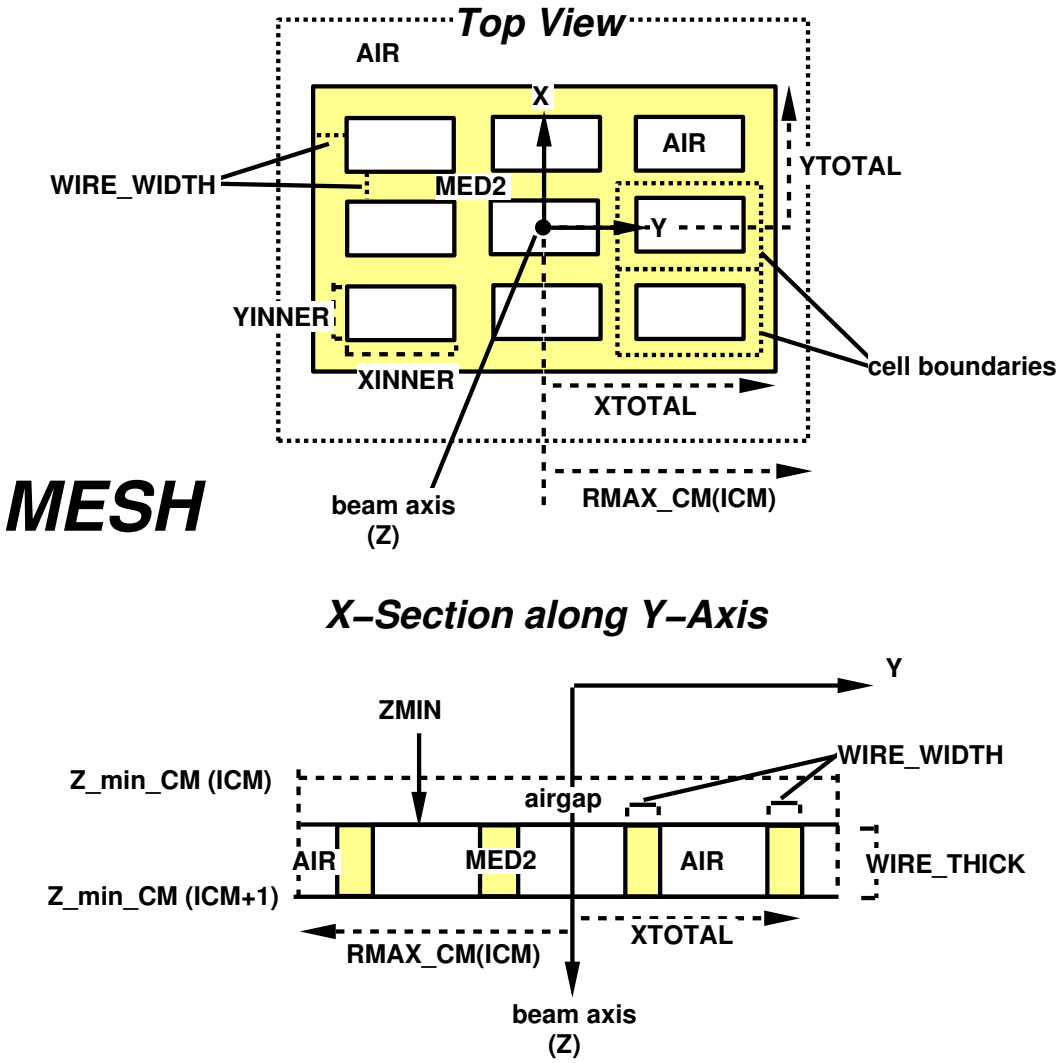


Figure 39: A MESH component module. The top view shows the identical rectangular air "holes", each with user-input dimensions `XINNER` and `YINNER`, separated from each other and the edge of the mesh by wire of user-input width, `WIRE_WIDTH`. The outer boundaries of the mesh are defined by half- widths `XTOTAL` and `YTOTAL` input by the user. Note that there is always one air hole centred on the beam axis. The top view also shows that a mesh "cell" is a single air hole plus half of the wire surrounding it. From a coding point of view, the mesh is made up of adjacent cells which behave identically with respect to particle transport. Thus the code actually only models one cell (the central one), and particles are translated from outer cells to their equivalent positions in this cell before transport. If the user-input values of `XTOTAL` and/or `YTOTAL` happen to cut across cells, the outer boundaries are automatically adjusted to accommodate these cells. The cross-section along the Y-axis shows the Z position of the front of the mesh, `ZMIN`, and the thickness of the mesh, `WIRE_THICK`, both input by the user. Note that it is possible to have an air gap above the mesh. For dose scoring and bit setting purposes, the wire is considered a single region, and the air in the holes and any air surrounding the mesh (excluding the air gap) are considered another region.

The input format for MESH and an example input are given below.

```
CARDS CM_$MESH  (MESH: Rev 1.4)
**************

-1  dummy line (filled with ****)  read in main

 0  RMAX_CM       outer boundary for CM - 1/2 side of square(read in main)

 1  TITLE_$MESH (60A1):  Title of CM.

 2  ZMIN_$MESH (F10.0):  Z position of front of MESH (excluding airgap).

 3  X_AIR_WIDTH_$MESH, Y_AIR_WIDTH_$MESH, WIRE_WIDTH_$MESH,
    WIRE_THICK_$MESH (4F15.0):

        X_AIR_WIDTH_$MESH: X width of each air region in mesh (cm)
        Y_AIR_WIDTH_$MESH: Y width of each air region in mesh (cm)
                           Y_AIR_WIDTH_$MESH defaults to
                           X_AIR_WIDTH_$MESH if it is set to 0.
        WIRE_WIDTH_$MESH:  Width of wire in the mesh (cm)
        WIRE_THICK_$MESH:  Thickness of wire in the mesh (cm)

      Note restrictions: X_AIR_WIDTH_$MESH,Y_AIR_WIDTH_$MESH,
                         WIRE_WIDTH_$MESH must all be >= 0.0001 cm

 4  XTOTAL_$MESH, YTOTAL_$MESH (2F15.0):

        XTOTAL_$MESH: Half-width of outer X dimension of mesh (cm)
        YTOTAL_$MESH: Half-width of outer Y dimension of mesh (cm)

      Note: XTOTAL_$MESH,YTOTAL_$MESH default to RMAX_CM if set to 0
            Also, if they fall within mesh cells (individual air regions +
              1/2 of the wire surrounding them) they are pushed out
              to include those cells.

 5  ECUT,PCUT,DOSE_ZONE,IR_TO_BIT for air inside and
                                  surrounding mesh (2F15.0,2I5):

       ECUT, PCUT:  Cutoff energies for electrons and photons.
        DOSE_ZONE:   Dose scoring flag, 0 to not score dose
   IREGION_TO_BIT:   Bit number associated with these regions

 6  ECUT,PCUT,DOSE_ZONE,IR_TO_BIT in local region 2 (wire region)
                                  (2F15.0,2I5):

       ECUT, PCUT:   Cutoff energies for electrons and photons.
        DOSE_ZONE:   Dose scoring flag, 0 to not score dose
   IREGION_TO_BIT:   Bit number associated with this region
```

```
  7  MED_IN (24A1):   medium of local region 2 (wire), used to set
                      MED_INDEX

  Example
  *******

The following input example describes a lead mesh placed at Z=100 cm
in a beam.  The mesh is 0.0299 cm thick, has square air regions
with dimensions 0.2159x0.2159cm, and the lead wire separating the air
regions is 0.0381cm wide.  Although the RMAX_CM is 10cm, the outer
boundary of the mesh itself, as input, is a 6x6cm square.  MESH
automatically adjusts this boundary to 6.2421x6.2421cm to accommodate
exactly 49 mesh cells (air regions + half of the wire width
surrounding them) in the X and Y directions plus the extra half-thickness
of wire surrounding the entire mesh and not belonging to any cell.

ECUT and PCUT in all regions are set to 0.521MeV and 0.01MeV
respectively.  Air regions (cell air + air surrounding mesh) will
have dose scored in zone 1 and is associated with bit # 1.  The
lead wire has dose scored in zone 2 and is associated with bit # 2.

Note that MESH cannot be the first CM in a simulation, and the
example given here must be preceeded by at least an airgap modelled
by another type of CM.

10.00000,                       Outer boundary
lead mesh: 0.0381cm with 0.2159cm air
100.0
0.2159,  0.0,  0.0381, 0.0299,  depth ensures equal mass as circle
6.0,  6.0,                      outer boundary fills entire beam
0.521,  0.01,  1,  1,              air is bit 1
0.521,  0.01,  2,  2,              wire is bit 2
PB700ICRU
```

15.3.23 MIRROR

MIRROR is used for a mirror in the accelerator. It can have arbitrary angle < 85 degrees with respect to the Z axis (for angles between 85 and 90 degrees, approximate using SLABS). The mirror portion itself can be made up of an arbitrary number of layers having different thicknesses and media. MIRROR has square symmetry.

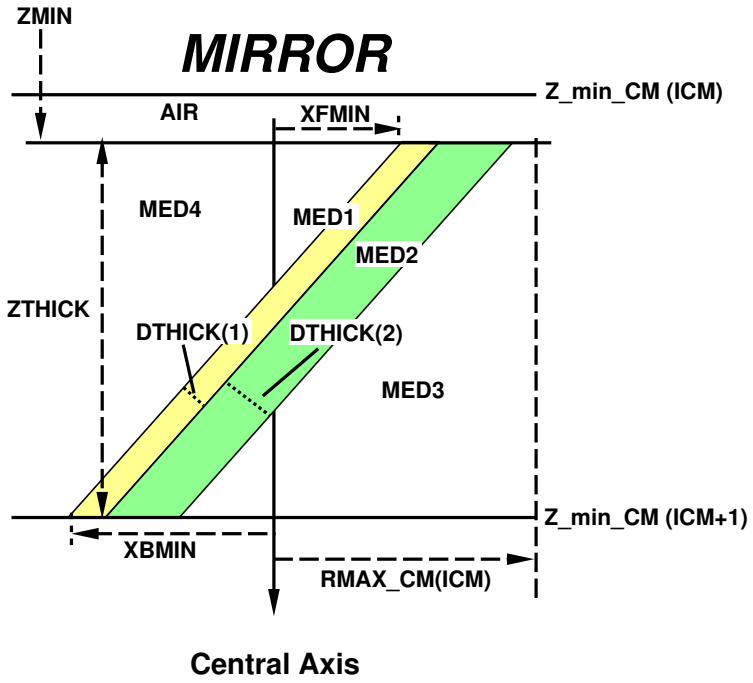


Figure 40: MIRROR example made up of 2 layers (N=2). The angle of the mirror with respect to the Z-axis is determined by XFMIN and XBMIN, the positions of the front face (layer 1) of the mirror, and ZTHICK thickness of the mirror in the Z-direction. This angle must be > 5 degrees. The thickness of layer i of the mirror is given by DTHICK(i). Note that layer i+1 of the mirror is flush with the right-most face of layer i. Thus, it is possible to specify a mirror where some layers are out of the path of the beam. It is even possible to specify XFMIN and XBMIN so that the front face of the mirror (the front face of layer 1) is out of the path of the beam; BEAMnrc checks for this and gives a warning message. The mirror extends to $+-$RMAX_CM in the Y-direction. The media in front of and behind the mirror, MED4 and MED3 in the figure, will usually be air but need not be.

The input format for MIRROR, and an example of input file are given as follows.

```
CARDS CM_$MIRROR
**************

-1  Dummy line to indicate start of CM.

 0  RMAX_CM(ICM_$MIRROR) (F10.0): Half-width of CM boundary (cm).

 1  TITLE_$MIRROR (60A1): Title of CM.

 2  ZMIN_$MIRROR,ZTHICK_$MIRROR (2F15.0):

        ZMIN_$MIRROR:   Distance from front of CM(excluding air gap) to
                        ref plane(Z=0).
        ZTHICK_$MIRROR: Z-direction span.

 3  XFMIN_$MIRROR, XBMIN_$MIRROR (2F15.0):

        XFMIN_$MIRROR:  X value at which front face of mirror
                        intersects ZMIN_$MIRROR.
        XBMIN_$MIRROR:  X value at which front face of mirror
                        intersects ZMIN_$MIRROR + ZTHICK_$MIRROR.
   Note restriction:
 5 degrees<ATAN(ZTHICK_$MIRROR/(XFMIN_$MIRROR-XBMIN_$MIRROR)<90 degrees

 4  N_$MIRROR, (I10):    Number of layers

 Repeat 5 for I=1,N_$MIRROR

 5  DTHICK_$MIRROR(I) (F15.0): Thickness of layer I in mirror,
                               in cm.  Layer 1 is the front face
                               of the mirror.

 Repeat 6 and 7 for I=1,N_$MIRROR.

 6  ECUT, PCUT,DOSE_ZONE, IREGION_TO_BIT (2F15.0, 2I5): for each layer

       ECUT, PCUT:  Cutoff energies for electrons and photons in
                    layer I.
       DOSE_ZONE:   Dose scoring flag in layer I.
  IREGION_TO_BIT:   bit to associate local region of layer I with

 7  MED_IN (24A1):  Medium of layer I, used to set MED_INDEX.

 8  ECUT, PCUT, DOSE_ZONE, IREGION_TO_BIT in region behind mirror
                                                  (2F15.0,2I5):

      ECUT, PCUT:  Cutoff energies for electrons and photons.
```

```
      DOSE_ZONE:   Dose scoring flag, 0 to score dose deposited in it
 IREGION_TO_BIT:   bit to associate local region with

9  MED_IN (24A1):  Medium of local region behind the mirror,
                   used to set MED_INDEX.

10  ECUT, PCUT, DOSE_ZONE, IREGION_TO_BIT in region in front of
                                          mirror(2F15.0,2I5):

      ECUT, PCUT:  Cutoff energies for electrons and photons.
      DOSE_ZONE:   Dose scoring flag, 0 to score dose deposited in it
 IREGION_TO_BIT:   bit to associate local region with

11  MED_IN (24A1): Medium of region in front of mirror,
                   used to set MED_INDEX.


Example
*******

The following describes a mirror composed of 1 AL slab.
The front face of the mirror intersects the front of the
CM (excluding any airgap) at X=3cm and the back of the CM at X=-3cm.
The mirror starts at 19.7cm and is 2cm in the z-direction.
The thickness of the AL slab is 0.00508cm.
Note that thickness is measured perpendicular to the
face of the mirror.  AIR is both in front of and behind
the AL mirror.  No dose is scored in this example.

10.00   RMAX_CM
MIRROR : z=19.5074 downstream, original, dif overlap
19.70, 2.0
3.0, -3.0
1
0.00508
0.0, 0.0, 0, 0
AL
0.0, 0.0, 0, 0
AIR
0.0, 0.0, 0, 0
AIR
```

15.3.24 XTUBE

A CM, which must be the first CM, for simulating an x-ray tube. XTUBE is actually similar in geometry to MIRROR. However, unlike MIRROR, XTUBE is usually the first CM in a beam model, and the source beam is usually incident on XTUBE from the side. The second CM may be any of the available `CMs` with a central opening serving as the exit window of the x-ray tube. The target of the xtube may have an arbitrary number of layers, each with a different thickness and medium. XTUBE has square symmetry.

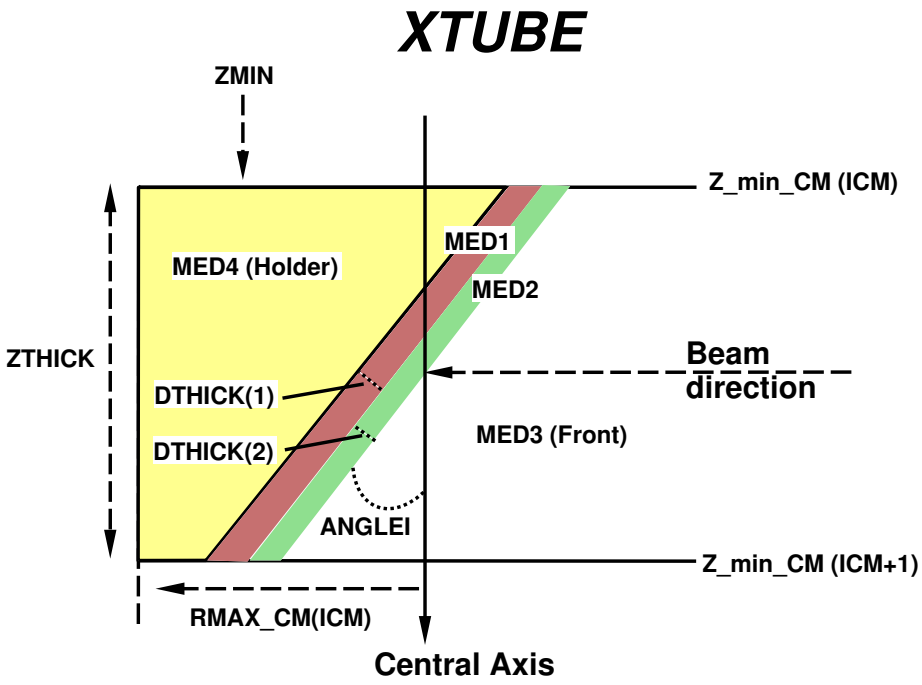


Figure 41: XTUBE component module with 2 layers in the target (`N`=2). The angle of the target relative to the Z-axis is given by `ANGLEI`. The thickness of a target layer i is given by `DTHICK(i)`. The target is positioned so that the front of the target (layer 2 in the figure) intersects with the beam at the central (Z) axis. Target layers are arranged with the front of layer i-1 flush with the back of layer i. The target and target holder extend to ± `RMAX_CM` in the Y-direction. MED3 in the figure above would usually be air or vacuum, but could be any material. MED4 specifies the material of the target holder.

The input format for XTUBE and a sample input are given below.

```
CARDS CM_XTUBE  (XTUBE SID: 2.5)
**************

-1  Dummy line to indicate start of CM.

 0  RMAX_CM(ICM_$XTUBE) (F10.0): Half-width of CM boundary (cm).

 1  TITLE_$XTUBE (60A1): Title of CM.

 2  ZMIN_$XTUBE,ZTHICK_$XTUBE (2F15.0):

      ZMIN_$XTUBE: Distance from front of CM to reference plane(Z=0).
    ZTHICK_$XTUBE: Thickness of xtube in Z-direction.

    Note that there will usually be no air gap at the front of
    this CM.

 3  ANGLEI (F15.0):  Angle between the target surface
                     and Z-axis (in degrees >= 0. and <75.)

 4  N_$XTUBE, I_XTRA_$XTUBE (I5):

          N_$XTUBE:   Number of layers in the target

 Repeat 5-7 for I=N_$XTUBE,N_$XTUBE-1,...,1
  (in XTUBE the first layer of the target is I=N_$XTUBE)

 5  DTHICK_$XTUBE(I), I_XTRA_$XTUBE (F15.0,I5)

  DTHICK_$XTUBE(I): Thickness of layer I (cm)
     I_XTRA_$XTUBE: Set to 1 if the layer has an extra central region
                    (only input for the outermost target layer)

 6  ECUT, PCUT,DOSE_ZONE, IREGION_TO_BIT (2F15.0,2I5):

       ECUT, PCUT:       Cutoff energies for electron and photon transport
                         of layer I.
        DOSE_ZONE:       Dose scoring flag of layer I.
        IREGION_TO_BIT:  Bit setting of region # corresponding to layer I.

 7  MED_IN (24A1):  Medium of layer I

 Note that inputs 8-10 below are only required if I_XTRA_$XTUBE=1
 (i.e. the outermost target layer has an extra central region)

 8  WXTRA_$XTUBE, HXTRA_$XTUBE (2F15.0):
```

```
     WXTRA_$XTUBE:  Width of extra central region in layer
                    N_$XTUBE (cm).
     HXTRA_$XTUBE:  Height of extra central region in layer
                    N_$XTUBE (cm).

 9  ECUT, PCUT,DOSE_ZONE, IREGION_TO_BIT (2F15.0,2I5):

       ECUT, PCUT:       Cutoff energies for electron and photon transport
                         in extra central region of layer N_$XTUBE.
        DOSE_ZONE:       Dose scoring flag of extra central region.
        IREGION_TO_BIT:  Bit setting of extra central region.

10  MED_IN (24A1):  Medium of extra central region.

11  ECUT, PCUT, DOSE_ZONE, IREGION_TO_BIT (2F15.0,2I5): for region in
                                                    front of target

        ECUT, PCUT:     Cutoff energies for electron and photon transport
                        in front of target
        DOSE_ZONE:      Dose scoring flag for this region.
        IREGION_TO_BIT: Bit setting of region # corresponding to this one.

12  MED_IN (24A1):  Medium in front of target

13  ECUT, PCUT, DOSE_ZONE, IREGION_TO_BIT (2F15.0,2I5): for target holder.

        ECUT, PCUT:      Cutoff energies for electron and photon transport
                         in target holder.
        DOSE_ZONE:       Dose scoring flag for target holder.
        IREGION_TO_BIT:  Bit setting of region for target holder.

14  MED_IN (24A1):  Medium of target holder

 Example
 *******

 The following example describes a tungsten target, 1mm thick,
 mounted on a copper holder.  The target is angled at 22 degrees
 with respect to the z-axis.  It spans 2cm in the z direction.
 The medium in front of the target is AIR.
 ECUT and PCUT for all regions are set to 0.521, 0.01 respectively.
 Note that the input file specifies the target to start at 10cm
 from the reference plane; if Z_min_CM(1) is not 10cm and XTUBE
 is the first CM in the beam, the code will autumatically reset the
 target to start at Z_min_CM(1).

 10.0       RMAX_CM
 XTUBE: z=10 cm, 1mm Tungsten target(1 slab), copper holder, 22 degrees
 10.0, 2.0; distance to reference plane = 10 cm, total thickness=2 cm
```

```
22.0;       angle = 22 degrees
1;          1 slab in the target
0.1;        thickness of the slab = 0.1 cm
0.521, 0.01, 0, 2; ECUT,PCUT,DOSE_ZONE,IREGION_TO_BIT for this slab
T;          medium is Tungsten
0.521, 0.01, 0, 2; ECUT,PCUT,DOSE_ZONE,IREGION_TO_BIT in front of target
AIR         medium is AIR
0.521, 0.01, 0, 2; ECUT,PCUT,DOSE_ZONE,IREGION_TO_BIT for the holder
CU;         medium for the holder is copper
```

15.3.25 SIDETUBE

SIDETUBE is a CM for modelling concentric cylinders parallel to the X-axis. As one of 3 CMs in which a radiating isotropic source (ISOURC=3) can be placed, it is designed for modelling any isotropic source that is perpendicular to the beam/collimation. The user can specify the number, radii and media of the concentric cylinders, the X positions of the cylinder ends, and the Z position of the axis of the cylinders. Note that all of the cylinders have the same end points, XMIN and XMAX. The user also specifies the medium surrounding the outer cylinder and filling up the rest of the cube in which the cylinders are contained. SIDETUBE has square symmetry about the beam axis.

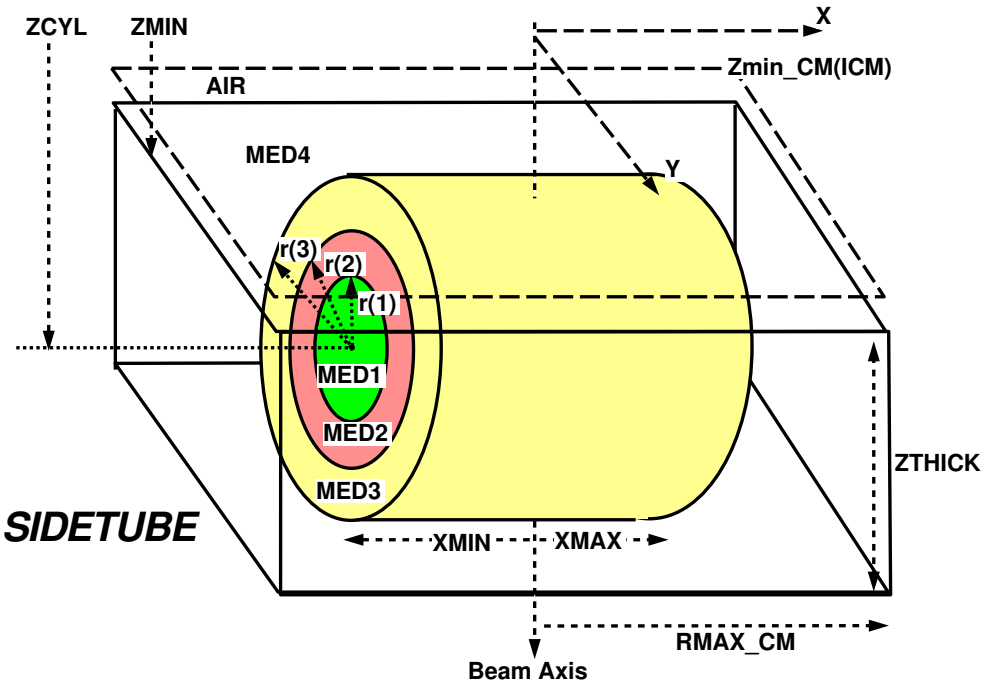


Figure 42: SIDETUBE component module with 3 concentric cylinders (N=3), their common axis parallel to the X-axis and located at Z=ZCYL. The outer radii of the cylinders are given by r(1), r(2), r(3), and the cylinders media are MED1, MED2, MED3, respectively. The ends of the cylinders are specified by XMIN and XMAX. MED4 is the user-specified medium surrounding the cylinders. Note that all of the cylinders must fit into the rectangular parallelepiped defined by thickness, ZTHICK, and half-width, RMAX_CM. ZMIN, the Z at which the cube containing the cylinders starts is not necessarily equal to Z_min_CM(ICM), allowing an air gap at the top of this CM.

The input format for SIDETUBE and a sample input are given below.

```
CARDS CM_$SIDETUBE
******************

-1  Dummy line to indicate start of CM.

 0  RMAX_CM(ICM_$SIDETUBE) (F10.0): Half-width of outer boundary
                                     of CM (cm).

 1  TITLE_$SIDETUBE (60A1):  Title of CM.

 2  ZMIN_$SIDETUBE (F10.0):  Distance from front of CM to reference plane
                             (not including air gap) in cm.

 3  ZTHICK_$SIDETUBE (F10.0): Thickness of the CM in the Z-direction
                              (not including airgap) in cm

 4  ZCYL_$SIDETUBE (F10.0):  Z position of axis of coaxial cylinders (cm)

 5  XMIN_$SIDETUBE,XMAX_$SIDETUBE (2F10.0):

        XMIN_$SIDETUBE: Lower X edge of cylinders (cm).
        XMAX_$SIDETUBE: Upper X edge of cylinders (cm).

 6  N_$SIDETUBE (I5): Number of coaxial cylinders.

 Repeat 7 for I=1,N_$SIDETUBE all on one line in order of increasing
 radius.

 7  R_$SIDETUBE(I) (F15.0): Outer radius of cylinder I (cm).

 Repeat 8-9 for I=1,N_$SIDETUBE+1.  When I=N_$SIDETUBE+1 you
 are specifying parameters for the region containing the concentric
 cylinders.

 8  ECUT, PCUT, DOSE_ZONE, IREGION_TO_BIT  (2F15.0,2I5):

        ECUT, PCUT:  Cutoff energies for electrons and photons
         DOSE_ZONE:  Dose scoring flag
    IREGION_TO_BIT:  Bit setting for the region

 9  MED_IN (24A1):  Medium of region
                    used to set MED_INDEX.
 Example
 *******

 The following example is a model of a cesium source parallel to
 the X-axis.  The source has radius 0.9cm and is made of cesium
```

```
chloride, CSCL700.  This source cylinder is contained within an
aluminum (AL700ICRU) cylinder of outer radius 1.25cm.  The
aluminum-cesium chloride assembly is contained in a stainless steel
(STEEL700ICRU) cylinder of inner radius 2.25cm and outer radius 2.4cm.
Note that the airgap between the aluminum and stainless steel is
modelled as a cylinder of air having outside radius 2.25cm.  The
cylindrical source assembly extends from X=-1.25cm to X=1.25cm.  The
volume of air in which the source assembly is located starts at
ZMIN=-2.4cm and has a thickness, ZTHICK=4.8cm, allowing it to just
contain the source assembly in the Z-direction.  Setting ZMIN=-2.4cm
allows the cylinders comprising the source assembly to be centered at
the reference plane (ZCYL=0).

In this example, ECUT, and PCUT in all regions are set to 0.7MeV
and 0.01MeV respectively.  The cesium source has dose zone 1; the
aluminum surrounding it has dose zone 2; the air between the aluminum
and the stainless steel outer tube has dose zone 3; and the stainless
steel tube has dose zone 4.

10.0,       Outer boundary
100Ci Cs source in holder + crude model of SS tube
-2.4,               ZMIN
4.8,                ZTHICK
0.,                 ZCYL
-1.125,1.125,       XMIN,XMAX
4,                  Number of cylinders
0.9,1.25,2.25,2.4,  Radii of cylinders
0.700,0.01,0,1,     ECUT, PCUT,...,MED of 1st cylinder
CSCL700
0.700,0.01,0,2,     ECUT, PCUT,...,MED of 2nd cylinder
AL700ICRU
0.700,0.01,0,3,     ECUT, PCUT,...,MED of 3rd cylinder
AIR700ICRU
0.700,0.01,0,4,     ECUT, PCUT,...,MED of 4th cylinder
STEEL700ICRU
0.700,0.01,0,0,     ECUT, PCUT,...,MED surrounding cylinders
AIR700ICRU
```

15.3.26 ARCCHM

ARCCHM is a CM designed specifically for modelling an arc-shaped array of chambers found in the prototype tomotherapy unit at the University of Wisconsin. However, it can be used to model any arc-shaped structure in the path of the beam. The arc is always concave up and curves in the Y direction with the lowest point of the arc (max. Z) always at Y=0. The arc itself consists of a single layer of chambers separated by septa sandwiched between a chamber front face and chamber back face. The user specifies the Z position of the chamber front face at Y=0, `ZSRC`, and the radius of the front face, `ZRAD1`. The angular extent of the arc and it's total span in the Z direction are determined by other user-specified parameters: the number of chambers, `NUMCHM` (which must be an even number), their width, `WIDTHCHM`, the width of the septa separating them, `WIDTHSEP`, the thickness of the chambers and septa, `ARCTHICK`, and the thicknesses of the chamber front and back, `FRONTHCK` and `BACKTHCK` respectively. The chamber Y ends are automatically added and the chamber back face extended so that they are flush with a plane at the minimum Z, `ZMIN` (calculated, not specified), of the arc. The user also specifies the minimum and maximum X limits of the arc (`XMIN1` and `XMAX2`, respectively), the thickness of the X ends of the chambers (`WIDXWALL`) and the maximum Z of the CM (`ZMAX`). The user can specify the media in all regions of the CM with the exception of the AIR above `ZMIN`. If the chambers all have the same `MED`, `ECUT`, `PCUT` and `IREGION_TO_BIT` and the septa do as well, then by setting `IREPEAT`=1, the user only has to input these parameters once for all chambers and once again for all septa. In this case, if the `DOSE_ZONE` input for the chambers is >0, then it is automatically incremented by 1 for each chamber, so that each chamber has its own dose zone.

This CM was originally coded by Marv Glass of the University of Wisconsin.

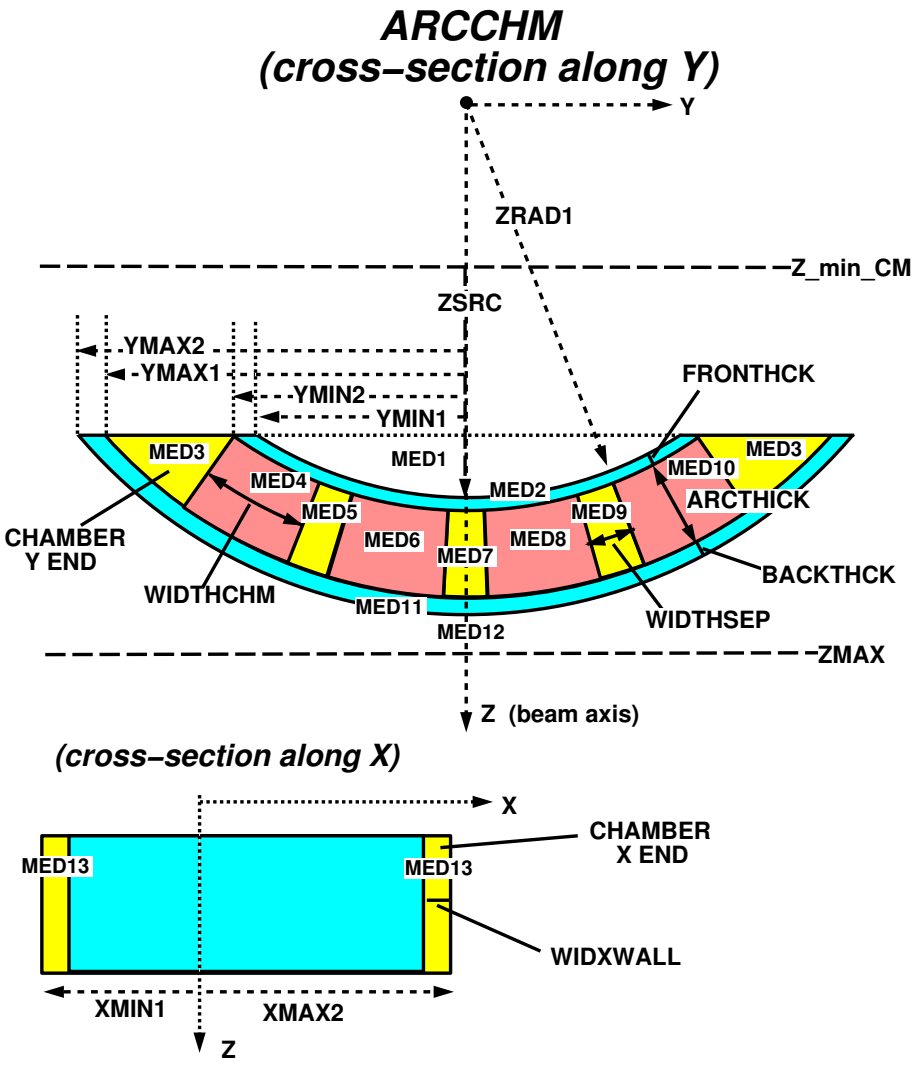


Figure 43: ARCCHM component module with 4 chambers (NUMCHM=4). The number of chambers must always be even so that they can be arrayed as shown in the Y cross-section, with a septum centred on the beam axis at Y=0. The number of septa is always 1 less than the number of chambers. The user also specifies the Z position of the front face of the chambers at Y=0, ZSRC, the radius of the arc at the front face, ZRAD1 (ZSRC=ZRAD1 for a perfectly isotropic source), the arc-width of the chambers, WIDTHCHM, and septa, WIDTHSEP, the thickness of the chambers/septa ARCTHCK, the thickness of the front face, FRONTHCK, and back face, BACKTHCK and the maximum Z of the CM, ZMAX. Other parameters such as the angular extent of the arc, the minimum and maximum Y dimensions, YMIN1, YMIN2, YMAX1 and YMAX2, and the minimum Z of the arc, ZMIN are calculated based on the user-specified parameters above. As shown, the chamber Y ends and back face are automatically extended so that they are flush with ZMIN. Note that ZMAX must be chosen to accommodate the full Z range of the arc. The cross-section in the X-direction shows the user-specified minimum and maximum X of the arc (XMIN1 and XMAX2 respectively) and width of the chamber X walls (WIDXWALL). The user can specify the medium in all regions except above ZMIN. If all the chambers have the same MED, ECUT, etc, and the septa do as well, then the number of inputs can be reduced by using the IREPEAT option.

The input format for ARCCHM and a sample input are given below.

```
CARDS CM_$ARCCHM
******************

-1  Dummy line to indicate start of CM.

 0  RMAX_CM(ICM_$ARCCHM) (F10.0): Half-width of outer boundary
                                   of CM (cm) Read in MAIN

 1  TITLE_$ARCCHM (60A1):  Title of CM.

 2  ZSRC_$ARCCHM (F15.0): Distance from front face of chamber at Y=0
                          (ie the lowest point in the arc) to reference
                          plane in cm.

 3  ZRAD1_$ARCCHM (F15.0): Radius of front face of chamber in cm.  For
                           a fully-divergent beam, this may equal
                           ZSRC_$ARCCHM.

 4  NUMCHM_$ARCCHM (I4):  Number of individual ion chambers

 5  WIDTHCHM_$ARCCHM (F15.0): Width of each ion chamber in cm.

 6  WIDTHSEP_$ARCCHM (F15.0): Width of each septum in cm.

 7  ARCTHICK_$ARCCHM (F15.0): Thickness of chambers & septa in cm.

 8  FRONTHCK_$ARCCHM (F15.0):  Thickness of front face of chamber in cm.

 9  BACKTHCK_$ARCCHM (F15.0):  Thickness of back face of chamber in cm.

10  WIDXWALL_$ARCCHM (F15.0):  Width of chamber wall along x in cm.

11  XMIN1_$ARCCHM,XMAX2_$ARCCHM (2F15.0):

        XMIN1_$ARCCHM: Min x dimension outside of x wall (cm).
        XMAX2_$ARCCHM: Max x dimension outside of x wall (cm).
     All YMIN and YMAX are calculated from ZRADs and PHI
     The min/max x dimensions inside of the x wall are calculated.
     ZMIN is calculated from ZRAD, ZSRC, and PHI

12  ZMAX_$ARCCHM (F15.0):  The max Z of the CM in cm.  Note that
                           ZMAX must be >= ZSRC+ARCTHICK+FRONTHCK+
                                           BACKTHCK

 Repeat 13-14 for the following regions:
       Region 1:  region before arc-shaped ion chamber.
       Region 2:  front face of arc-shaped ion chamber.
```

```
        Region 3:  ends of the ion chamber.
        Region 4 -- 2*NUMCHM+2: chamber or septa
              EVEN:  chamber   ODD: septa  Numbering of chambers and
              septa goes from -Y to +Y  (13-14 only have to be
              repeated twice for these regions if IREPEAT=1.  See
              below.)
        Region 2*NUMCHM+3:  back face of arc-shaped ion chamber.
        Region 2*NUMCHM+4:  region surrounding the arc.
        Region 2*NUMCHM+5:  x walls of the chamber

  13  ECUT, PCUT, DOSE_ZONE, IREGION_TO_BIT, (IREPEAT) (2F15.0,2I5,(I5)):

          ECUT, PCUT:  Cutoff energies for electrons and photons
           DOSE_ZONE:  Dose scoring flag
      IREGION_TO_BIT:  Bit setting for the region
             IREPEAT:  Only input for Region 4.  Set to 1 to apply this
                       ECUT, PCUT, IREGION_TO_BIT and MED_IN to all
                       chambers (if DOSE_ZONE > 0, then it will be
                       incremented automatically for each chamber) and the
                       following ECUT, PCUT, DOSE_ZONE, IREGION_TO_BIT and
                       MED_IN to all septa.

  14  MED_IN (24A1):  Medium of region
                      used to set MED_INDEX.

 Example
 *******

 The following example is an arc-shaped ion chamber containing 10
 individual chambers.  Used for the tomotherapy prototype under
 development at the UW.

 In this example, ECUT, and PCUT in all regions are set to 0.7 MeV
 and 0.01 MeV respectively.  Note that IREPEAT is set to 1 for the
 first chamber so that ECUT, PCUT, etc only have to be input once
 for all chambers and once for all septa.  Also, DOSE_ZONE will
 be incremented automatically for the chambers, so that the chambers
 will have dose zones 1-10.

 ************dummy line to indicate new CM**********************
 60.0,               RMAX:  outer boundary
 Tomotherapy ion chamber
 133.0,               ZSRC: distance to front of ARC
 133.0,               ZRAD1: radius of front of arc
 10,                  NUMCHM: the number of individual ion chambers
 1.0,                 WIDTHCHM: width of each individual ion chamber
 0.1,                 WIDTHSEP: width of septa
 5.0,                 ARCTHICK: thickness of the arc
 0.1,                 FRONTHCK: thickness of front face of chamber
```

```
0.2,                BACKTHCK: thickness of back face of chamber
0.2,                WIDXWALL: thickness of x wall
-5.0, 5.0,          XMIN1, XMAX2: min/max x dimension outside of x wall
139.0,              ZMAX: Z limit of CM
0.700,0.01,0,1,     ECUT, PCUT,...,MED of region 1
AIR521ICRU
0.700,0.01,0,2,     ECUT, PCUT,...,MED of front face
AL521ICRU
0.700,0.01,0,3,     ECUT, PCUT,...,MED of edges of chamber
AL521ICRU
0.700,0.01,1,1,1,   ECUT, PCUT,...,MED of all chambers
H2O521ICRU
0.700,0.01,0,1,     ECUT, PCUT,...,MED of all septa
PB521ICRU
0.700,0.01,0,2,     ECUT, PCUT,...,MED of back face
AL521ICRU
0.700,0.01,0,1,     ECUT, PCUT,...,MED of region surrounding arc
AIR521ICRU
0.700,0.01,0,2,     ECUT, PCUT,...,MED of x walls
AL521ICRU
```

16 Cross-Section Data – PEGS4

Cross-section data for many commonly-used media are included in EGSnrc installation in the files `521icru.pegs4dat` and `700icru.pegs4dat`, both located in the `$HEN_HOUSE/pegs4/data` directory. The file `521icru.pegs4dat` consists of cross-section data from a lower electron energy, `AE`, of 0.521 MeV to an upper electron energy, `UE`, of 55 MeV, while `700icru.pegs4dat` contains data from `AE`=0.700 MeV to `UE`=55 MeV. In both files the lower photon energy, `AP`, is 0.01 MeV and the upper photon energy, `UP`, is 55 MeV. These data are based on the density effect corrections in ICRU Report 37 [42].

If you wish to determine exactly what the composition of a given material is, it is specified on the lines immediately following the `MEDIUM` label in the `.pegsdat` file. For an up-to-date listing of what materials are available in these files, do as follows (only for Linux/Unix):

```
cd $HEN_HOUSE/pegs4/data
grep MEDIUM 700icru.pegs4dat
```

or

```
grep MEDIUM 521icru.pegs4dat
```

To use these data you must specify the names **exactly** as they appear in the data file being used.

The NRC EGS user-code examin (found on `$HEN_HOUSE/user_codes/examin`) can be used to tabulate or plot the various cross sections for each material.

16.1 Creating additional cross section data

Additional cross section data for BEAMnrc is created by the code PEGS4 (located in `$HEN_HOUSE/pegs4`). A complete description of the PEGS4 code is included in the EGSnrc manual[1], and users wishing to create their own PEGS4 data are referred to that manual.

Briefly, PEGS4 can be invoked on the command line by typing:

```
pegs4.exe -i inputfile [-d densityfile]
```

where `inputfile` omits the `.pegs4inp` extention and `densityfile` omits the `.density` extension. The `.pegs4inp` file specifies whether the the medium is an element (`ELEM`), compound (`COMP`) or mixture (`MIXT`); the composition of the medium (by mass, `RHOZ`, in the case of `MIXT` and number of atoms, `PZ`, in the case of `COMP`); the density (`RHO`; whether Rayleigh cross-sections are to be included (`IRAYL`=1); the stopping powers calculated (`IUNRST`=0 for restricted stopping powers, `IUNRST`=1 for unrestricted collision stopping powers, etc.); and whether ICRU 37 density corrections are to be included (`EPSTFL`=1). The `.pegs4inp` file also specifies the values of `AE`, `AP`, `UE` and `UP` to use and the name by which the medium will be referred in all input files using it. There are other input options, but they are beyond the scope of this manual.

If the user has selected to use ICRU 37 density corrections (`EPSTFL`=1), then they must include the `-d densityfile` input when running `pegs4`. Density correction files are found in the directory `$HEN_HOUSE/pegs4/density_corrections`. It is important to note that if you are using a density correction file, the value of `RHO` specified in the `.pegs4inp` file must

match the material density in the `densityfile.density` file, which is the third value on the second line of the file.

The EGSnrc distribution includes some sample input files in the directory `$HEN_HOUSE/pegs4/input`, and it is useful to use these as references if creating your own `.pegs4inp` file.

When running PEGS4 as described above PEGS4 data is automatically output to either `$HEN_HOUSE/pegs4/data/inputfile.pegs4dat` or `$EGS_HOME/pegs4/data/inputfile.pegs4dat`, depending on what directory it is being run from.

An much easier way to create PEGS4 data is to run PEGS4 from the `egs_gui` (invoked simply by typing "`egs_gui`"). A screen shot of the PEGS4 option in the `egs_gui` is shown in Figure 44.

Note that `egs_gui` offers an option to append newly-created PEGS4 data to an existing `.pegs4dat` file.

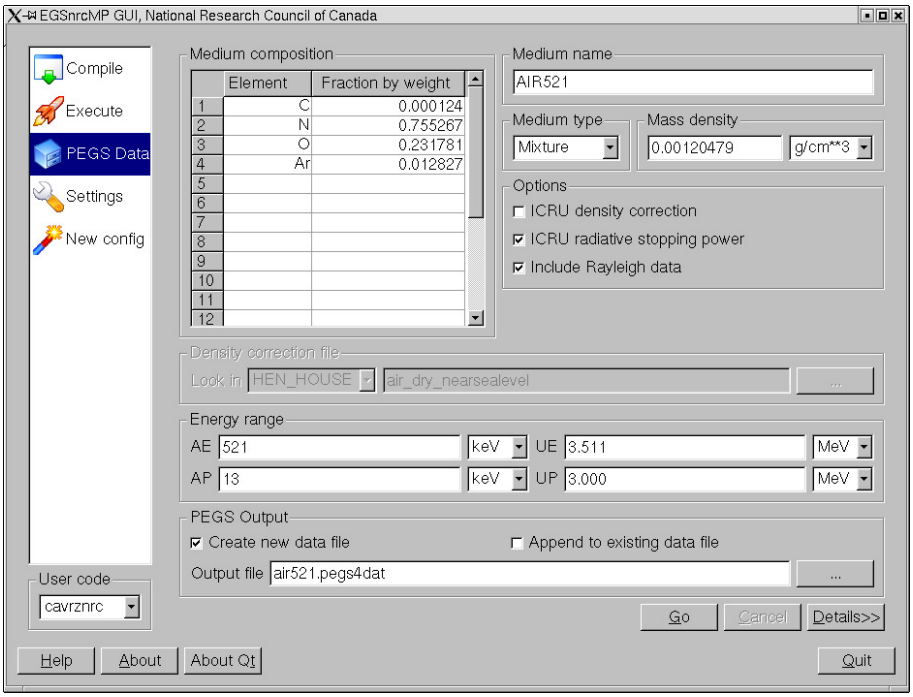


Figure 44: Screen shot of the `egs_gui` set to run PEGS4 to create an air data set. Filling in this form is much easier than creating an input file and access to the density effect corrections is easy. The GUI does not allow any value except `IUNRST = 0`. For details, see PIRS-877[2].

16.2 Choice of AE,AP

The parameters `AP` and `AE` are the low-energy thresholds for the production of secondary bremsstrahlung photons and knock-on electrons, respectively. The parameters `PCUT` and `ECUT` are required to be greater than or equal to `AP` and `AE` (see sections 10.11 and 10.10).

Selection of `AP` is simple since one can afford to use a very low value which ensures accurate photon transport. We recommend using `AP` = 0.010 MeV. Note that in practice

this means all bremsstrahlung events are simulated as discrete events.

The choice of AE is more complex since there is some computing time associated with lower values of AE and in some cases, lower values lead to more accurate simulations. The value of AE controls the statistical fluctuations in the energy loss, can affect the electron step sizes and is a lower limit on ECUT. For a general discussion see, *eg.* [36, 26].

Although the choice of AE is complex in general, it is fairly easy to give general rules for. In practice, the value of AE has little effect on dose calculations, except in the sense that AE is the lower limit on ECUT. Thus for dose calculations, one should select ECUT as discussed in section 10.10 and then use AE = ECUT. For calculations in which one is looking at electron spectra directly in an electron beam, it is important to use a fairly low value of AE in order to avoid well known artifacts[36, 26]. Figure 45 gives an example of this artifact for the 9 MeV beam from a Clinac2100C. The spectrum calculated with AE=0.521 MeV is clearly more realistic than that calculated with AE=0.700 MeV. We find that AE = 0.521 MeV is adequate - but note, one can still use ECUT much higher.

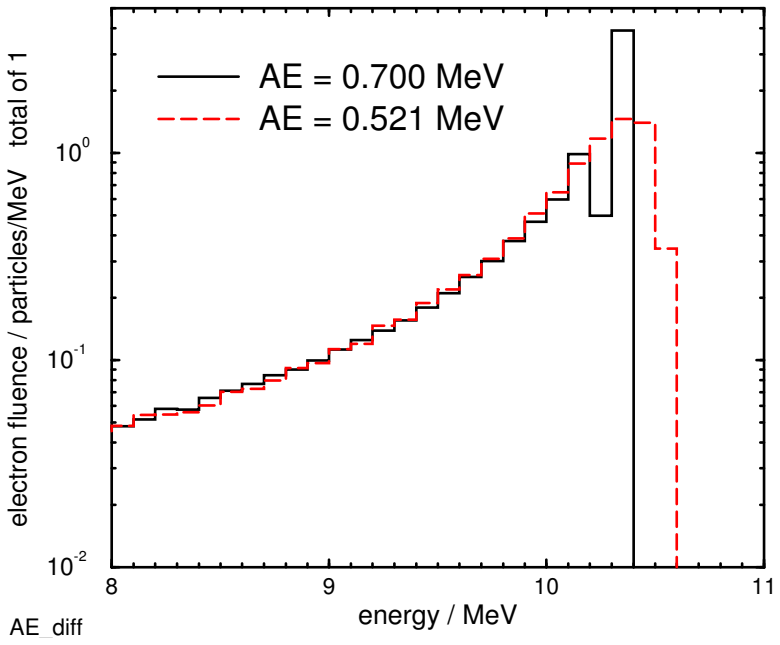


Figure 45: Example of the differences in electron spectra calculated with AE=0.521 MeV and AE=0.700 MeV. The spectra are normalized to one particle in the total spectrum. The specific example is for the 9 MeV beam from a Clinac2100C with a type III applicator. The artifact near the peak in the AE=0.700 MeV case is discussed in detail in references [36, 26].

16.3 Pegsless Mode

As of 2013, the user has the option of running BEAMnrc (and other EGSnrc user codes) in pegsless mode, thus obviating the requirement for a-priori calculation of cross-section data and generation of a .pegs4dat file. Photon interaction cross-sections have been calculated

on-the-fly since 2006. Thus, the migration to a fully pegsless implementation of EGSnrc required that the calculation of electron cross-sections be similarly done on a simulation-by-simulation basis.

To run in pegsless mode, the user must include parameters for calculating cross-sections, including specifications for all media used in the simulation, in their `.egsinp` file between the delimiters, `:start media definition:` and `:stop media definition:`. This is best illustrated with an example input:

```
:start media definition:

AE=0.521
UE=50.511
AP=0.01
UP=50.

material data file=/home/username/HEN_HOUSE/pegs4/data/material.dat

:start H2O521ICRU:
  elements = H, O
  number of atoms = 2,1
  rho = 1.0
  bremsstrahlung correction = KM
:stop H2O521ICRU:

:start AIR521ICRU:
  elements = C,N,O,AR
  mass fractions = 1.24000E-04, 7.55200E-01, 2.31800E-01, 1.28300E-02
  rho = 1.2048E-03
  bremsstrahlung correction = NRC
  gas pressure = 1.0
:stop AIR521ICRU:

:start PMMA521ICRU:
  bremsstrahlung correction = NRC
  density correction file = /home/uname/EGSnrc/HEN_HOUSE/pegs4/density_corrections/
compounds/polymethylmethacrylate__lucite___perspex___plexiglas_.density
:stop PMMA521ICRU:

:stop media definition:
```

where:

`AE,UE,AP,UP` are the kinetic energy limits for calculating photon (`AP,UP`) and electron (`AE,UE`) cross sections in MeV. These energy limits are also mentioned in the context of the EGSnrc system in the section `The COMMON Blocks`. If `AE` is not specified it defaults to the highest value of `ECUT` (electron transport cutoff energy–see Section `Pre-HATCH Call Initialisation (Step 2)` in PIRS701) specified in the simulation.

If AP is unspecified, then it defaults to the highest value of PCUT (photon transport cutoff energy–see Section Pre-HATCH Call Initialisation (Step 2) in PIRS701) specified. UE and UP default to 50.511 MeV and 50.0 MeV, respectively.

material data file is the name (including full directory path) of a file containing specifications for the media used in the simulation. Provided with the EGSnrc distribution is the material data file $HEN_HOUSE/pegs4/data/material.dat which contains specifications necessary to reproduce all of the cross-section data in 521icru.pegs4dat and 700icru.pegs4dat (provided that the appropriate values of AE, UE, AP and UP are specified–see above). Note that the format for media specifications in the material data file is similar to that used for specifying media directly in the .egsinp file, described immediately below. In the case of the material data file, however, there is some redundancy in the specification to allow the user to see the composition and density of the media.

the :start MEDNAME: and :stop MEDNAME: delimiters are used to specify medium, MEDNAME, directly in the .egsinp file. This method of specifying a medium is used if MEDNAME is not included in the material data file or if you wish to override some or all of the specifications for MEDNAME in the material data file. If no material data file is input, then all media in the simulation must be specified in this way. The inputs between these delimiters are described below. Variables in square brackets are the analogous PEGS4 variables described in the PEGS4 Manual (Section PEGS4 User Manual in PIRS701) above.

elements specifies the elements comprising the medium. Elements are specified using chemical symbols separated by commas. Case is unimportant.

number of atoms [PZ] or mass fractions [RHOZ]. For each of the elements, specify either the number of atoms in a molecule of the medium (*i.e.* stoichiometric coefficients), if the medium is a compound, or the mass fractions of the elements in the medium, if the medium is a mixture. Values are separated by commas and are input in the same order as their corresponding elements. In the example above, the composition of H2O521ICRU is defined using the number of atoms of each element, while that of AIR521ICRU is defined using the mass fraction of each element. Note that this input is omitted if the medium is an element.

rho specifies the bulk density of the medium in g/cm^3.

bremsstrahlung correction [IAPRIM] specifies the correction to apply to calculated bremsstrahlung cross-sections. Options are:

- KM [IAPRIM=0] (the default): Apply Koch and Motz[39] empirical corrections.
- NRC [IAPRIM=1]: Apply NRC corrections based on NIST/ICRU[53]. These corrections are read from the file $HEN_HOUSE/pegs4/aprime.data.
- None [IAPRIM=2]: No corrections applied.

density correction file [EPSTFL] is the name of a file containing density effects which, when applied to calculated collision stopping powers, results in agreement with collision stopping powers published in ICRU37[42]. In general, density correction files are specified including their full directory path and .density file

extension. However, the file can be specified by its prefix only if it exists in, in order of search priority:

1. `$EGS_HOME/pegs4/density_corrections`
2. `$EGS_HOME/pegs4/density_corrections/elements`
3. `$EGS_HOME/pegs4/density_corrections/compounds`
4. `$EGS_HOME/pegs4/density`
5. `$HEN_HOUSE/pegs4/density_corrections/elements`
6. `$HEN_HOUSE/pegs4/density_corrections/compounds`

Note that the density correction files for many elements, compounds and mixtures are supplied with the distribution. Density correction files have a header portion from which the composition and bulk density of the medium are read. These values override any user inputs for `elements=`, `number of atoms=` or `mass fractions=`, and `rho=`. Thus, as in the case of `PMMA521ICRU` in the example above, it is possible to specify the composition of a medium simply by specifying a density correction file.

`gas pressure` [`GASP`] is the pressure of the medium in atm if the medium is a gas. This input is only relevant (and necessary for a gas) if a density correction file is not used, in which case `gas pressure` is used to modify the calculated density effect parameters. `gas pressure` defaults to 0 (*i.e.* the medium is not a gas).

Inputs specifying media are case insensitive with the exception of the medium name (*e.g.* `H2O521ICRU` is different than `h2o521ICRU`).

Note that, along with the medium name, the medium composition, specified by `elements` and `number of atoms` or `mass fractions`, and its bulk density, specified by `rho`, are sufficient to calculate cross-sections. Thus, it is possible to completely define a medium using only these three inputs, with all other parameters reverting to their default values or remaining blank. Alternatively, since composition and bulk density can be read from the header of the density correction file, it is also possible to define a medium simply by specifying the name of the relevant density correction file.

The BEAM GUI allows you to define and modify the media definition section of the `.egsinp` file through the "Define Media" button in the "Main Inputs" window. This button is enabled when you go into "Change PEGS4 file" under the "File" menu and opt to "Go PEGSless".

Running in Pegsless Mode

BEAMnrc can be run interactively in pegless mode using the command line input:

```
BEAM_myaccel -i inputfile
```

where `inputfile` is the name of the BEAM inputfile (with no `.egsinp` extension).

Pegsless batch runs use the command line syntax:

```
exb BEAM_myaccel inputfile pegsless [short|medium|long] [batch=batch_system] [p=N]
```

This is identical to the syntax of a run with pegs data (see Section 2.7.1 above) but with the word `pegsless` in place of the name of the pegs data file.

When running in pegsless mode, BEAMnrc outputs a file, `inputfile.mederr`, which, for each medium used in the simulation, indicates where each specifying parameter has been read (*i.e.* from a material data file or directly from the `.egsinp` file). The file also includes warnings when `AE`, `UE`, `AP` or `UP` have not been specified and have been set to their default values and when a material data file has not been specified. For parallel runs in pegsless mode (parameter `p` > 1 in the batch command syntax above), the `.mederr` file is only output by the first job.

The actual values of the media specifications (including defaults) used to calculate cross-sections are output in the listing file, `inputfile.egslst`, and to the screen for interactive runs or in the log file, `inputfile.egslog`, for batch runs.

17 Distribution and Installation

For up-to-date installation instructions, see the online documentation available on the github page: https://github.com/nrc-cnrc/EGSnrc/wiki/Installation-overview

In order to install and run EGSnrc and BEAMnrc, your system must have:

1. A working Fortran compiler. If you do not have one on your system, GNU provides a suite of compilers that can be downloaded for free at http://www.gnu.org
2. A working `make` utility. Most systems have this built in, although there is one available at the GNU site as well.
3. A working C or C++ compiler. This is a requirement on Unix/Linux systems if you want to take advantage of the built-in parallel processing capability of BEAMnrc and DOSXYZnrc (and other EGSnrc user codes) and is of no use if you do not have a batch submission capability. It is also required on Unix/Linux systems (but not on Windows) if you want to use a BEAM simulation as a source with DOSXYZnrc or with any other EGSnrc user code. The compiler is available at the GNU site.
4. A working C++ compiler. This is required if you want to be able to read/write IAEA-format phase space files.
5. Tcl/Tk. This is necessary for running the BEAMnrc, DOSXYZnrc and BEAMDP GUIs. It can be downloaded from http://www.activestate.com/activetcl/downloads. See the GUI User's Manual[4] for more information. Note that most Linux distributions already include Tcl/Tk.
6. The Qt4 development tools. This is necessary for compiling the GUIs for the EGSnrc user codes and is, thus, not strictly necessary for BEAMnrc codes or if you can use the pre-compiled versions provided online. Pre-compiled versions of the Qt GUIs are available on the EGSnrc release page. Note that many current Linux distributions include Qt, since the popular Desktop Environment KDE is based on Qt.
7. The Grace plotting package (`xmgrace` command) which is available at http://plasma-gate.weizmann.ac.il/Grace. This is distributed with many Linux distributions and there is a Windows version called QtGrace.

17.0.1 A Note on Tcl/Tk

Before running the BEAMnrc, DOSXYZnrc and BEAMDP GUIs, you must first have installed Tcl/Tk (this is already installed in most Linux systems) and have included the directory `/full path to Tcl installation/bin` in your `PATH` environment variable. For more information on installing Tcl/Tk, see the GUI Manual[4].

18 Known Problems/Restrictions

If RMAX is interior to any other boundaries in the geometry, the results will be in error since volumes are not correct and also energy past RMAX is always deposited in region(1), even if still inside another region.

The value for a particle does not take into account the distance to `RMAX_CM`, i.e it is allowed to approach this boundary with large steps. This will lead to inaccuracies in the dose delivered to these regions. For accelerator simulations these errors will be small, but in unusual situations could be significant.

Variables related to the number of particle histories, such as the input variable `NCASE`, the variable `IHSTRY`, etc, have been changed to `INTEGER*8` to allow for $> 2\text{x}10^9$ histories. This is particularly useful for restarts and recombining parallel jobs, where a large number of histories may be accrued. However, some Fortran compilers do not support `INTEGER*8`. We have compiled with `INTEGER*8` successfully on Linux PC, SGI, DEC Alpha, and Sun sparc systems. On our rs6000 system, the compiler gave "length specified is not valid for the specified type" warnings, reverted to `INTEGER*4` and then compiled successfully. On our HP9000 system, the compiler gave "incompatible type-length combination" errors and compilation failed. If compilation fails due to `INTEGER*8`, go into `beamnrc.mortran` and change the macro:

```
REPLACE {$LONG_INT} WITH {INTEGER*8}
```

to:

```
REPLACE {$LONG_INT} WITH {INTEGER*4}
```

The code will now compile, but note that you are now limited to $2\text{x}10^9$ total histories.

If `ICM_SPLIT`=1, *i.e.* you want to split particles going into the first CM, and even though you must input `NSPLIT_PHOT` and `NSPLIT_ELEC`, no splitting will occur. To get around this bug, if you wish to split particles at CM 1, you can insert a dummy CM 1 and then set `ICM_SPLIT`=2.

Not all CMs output geometric information of much value for `EGS_Windows` (in particular MESH and ARCCHM).

A restarted run that uses a phase space source with particle recycling will not produce dose/uncertainty results identical to a single run with the same total number of histories. This is because the last particle used before the restart may not have been recycled the full

NRCYCL times, and restarting automatically skips to the next particle. Results will agree within uncertainty, however.

19 History of Revisions

This section gives an overview of major changes to the BEAMnrc system (including DOSXYZnrc) between releases. For releases up to and including BEAMnrc07, the files CHANGES_from_BEAMnrc06.for.BE CHANGES_from_BEAMnrc05.for.BEAMnrc06, CHANGES_from_BEAMnrc03.for.BEAMnrc05, CHANGES_from_B and similar files on $OMEGA_HOME/doc contain all the changes made to the BEAM between releases. BEAM95 was released at the first course in Oct, 1995.

19.1 Changes from BEAMnrc V4 2.3.2 (18/05/11) to BEAMnrc V4 2.4.0

Major changes between BEAMnrc V4 2.3.2 and BEAMnrc V4 2.4.0 are as follows:

- Addition of synchronized component modules, SYNCJAWS, SYNCVMLC, SYNCMLCE and SYNCHDMLC, for modeling motion of jaw or mlc leaf opening coordinates (defining fields) during an accelerator simulation. These CMs can be used in dynamic (continuous motion) or step-and-shoot (motion only while beam off) mode. If there are multiple synchronized CMs in an accelerator, then the motion of the opening coordinates in all of them can be synchronized. In addition, motion of synchronized CM opening coordinates can be synchronized with the motion of DOSXYZnrc's source 20 (synchronized phase space source–see below) and source 21 (synchronized BEAM simulation source–see below).

  Note that SYNCJAWS is a synchronized version of DYNJAWS, SYNCVMLC is a synchronized version of DYNVMLC, SYNCMLCE is a synchronized version of MLCE (which previously had no dynamic option), and SYNCHDMLC is a variation of SYNCVMLC allowing the definition of two addition leaf types (cross-sections). SYNCHDMLC is optimized for simulating the high-definition MLC (HDMLC 120) available on TrueBeam and Novalis linacs.

- Addition of source 20 (synchronized phase space source) and 21 (synchronized BEAM simulation source) to DOSXYZnrc. These sources simulate continuous motion of the source plane between control points input by the user. Each control point is associated with a fractional monitor unit index indicating the fraction of incident particles incident up to that point.

  Source 20 allows the user to interpose a geometry (usually an MLC), defined using either a BEAM accelerator compiled as a shared library or a non-EGSnrc code compiled as a shared library, between the source plane and the phantom. In the case of a BEAM accelerator, any synchronized CMs in the geometry can be synchronized with the motion of the source.

  For source 21, if there are any synchronized CMs in the BEAM simulation source, then their motion (changing opening coordinates) can be synchronized with the motion of the source plane around the DOSXYZnrc phantom.

- Fixed a bug when using IAEA-format phase space sources where the energy passed to the EGSnrc code was kinetic energy, rather than total energy. EGSnrc user codes expect total energy.
- When using a BEAM simulation source, pass the unit number for the `.egslog` file for the driving code to the `init_beamsource` routine so that initial info about the BEAM source gets output to the driving code's `.egslog` file. Previously, this info was being written to STDERR, causing the DOSXYZnrc GUI to return a message saying that a run using a BEAM simulation source had failed when it had not.
- Fixed a bug in the setting of `LATCH` bits after a bremsstrahlung event with `LATCH_OPTION`=1.

19.2 Changes from `BEAMnrc08` to `BEAMnrc09`

Major changes between `BEAMnrc08` and `BEAMnrc09` are as follows:

- Modified subroutine `do_rayleigh` in `beamnrc.mortran` so that a new Rayleigh sampling scheme can be used with DBS. Modification includes no longer killing thin photons before Rayleigh interactions to avoid spikes in the photon spectra.
- `LATCH` options now work when DBS is used. They previously did not.
- CM XTUBE was modified to allow the user to have a central region in the outermost target slab comprised of a different material. Should allow more accurate modeling of X-ray tubes.
- Source 24, phase-space source incident from a user-specified angle, added. Concept and much of the coding of the source is courtesy of Patrick Downes at University of Cardiff, Wales.
- Added source 23, BEAM simulation source from a user-specified angle.
- Fixed bug in CM MLC in which a particle right on a leaf boundary with USTEP $\sim 1 \times 10^{-6}$ resulted in an endless loop.
- Fixed bug in source 8 in DOSXYZnrc (phase-space source incident from multiple directions) causing a segmentation fault when many incident angles were specified.
- Fixed bug in BEAMnrc which caused elapsed CPU time stored in the `.egsdat` file to be wrong.
- Changed source 19 from a circular gaussian source to an elliptical source defined by separate gaussian distributions in X and Y.
- Many changes to allow compilation and running in double precision (`REAL*8`).
- Added a new component module, DYNJAWS, to allow simulation of time-dependent JAW settings. In actuality, this is most likely to be used to simulate dynamic wedges. Similar to DYNVMLC, it can be operated in dynamic (beam on during setting changes) and step-and-shoot (beam off during setting changes) modes.

- Fixed a major bug in MLCQ. The distance to the nearest region boundary was calculated incorrectly in the HOWNEAR subroutine. In at least one observed case, this led to an error in electron fluence at the bottom of the CM (incident electron beam) and also caused transport through the CM to be very slow.

19.3 Changes from `BEAMnrc07` to `BEAMnrc08`

Major changes between `BEAMnrc07` and `BEAMnrc08` are as follows:

- Recoded the opening of `.egsinp`, `.egslog`, `.egslst` and `.errors` files in BEAMnrc so that you no longer require `libg2c.a` when using BEAMnrc as a shared library source. Now, BEAMnrc searches for available Fortran units (`i.e.` those not already in use by the driving program) to associate with these files. Thus, the confusion of Fortran units, for which `libg2c.a` was necessary to resolve, is no longer an issue. This should allow BEAMnrc shared libraries to be compiled using `gfortran`, which had problems with `libg2c.a`.

- Introduced dynamic and step-and-shoot modes in the DYNVMLC component module. This allows multiple treatment fields to be simulated in a single run.

- Both BEAMnrc and DOSXYZnrc can now make use of IAEA-format phase space data. In the case of BEAMnrc, IAEA-format data can now be output at scoring planes. This will allow users to use the new IAEA online phase space database. A C++ compiler is required for this functionality.

- Added source 10 to DOSXYZnrc. This is source 9 (BEAMnrc shared library source) but from multiple angles in a single simulation.

- "NRC" bremsstrahlung cross-sections, which include electron-electron bremsstrahlung effects, have been added

- The user now has the option to specify custom Rayleigh form factors for specified media if simulation of Rayleigh scattering is desired.

- The bremsstrahlung cross section enhancement (BCSE) variance reduction technique was introduced. This increases the efficiency of x-ray tube simulations by up to an order of magnitude and can even increase the efficiency of standard 6 MV accelerator simulations by 40%.

- The option to use a rejection plane which can eliminate all fat photons which might interact in the air above the phantom has been added as part of DBS.

- A grid scoring option in each scoring plane has been added so that fluence can be scored in a rectangular grid rather than just in circular or square rings.

19.4 Changes from BEAMnrc06 to BEAMnrc07

Major changes between BEAMnrc06 and BEAMnrc07 are as follows:

- Introduced "HOWFARLESS" option for homogeneous phantom calculations in DOSXYZnrc. This option increases efficiency by ∼30% in photon beam sources from accelerators simulated using BEAMnrc and by factors of 2.5-3.5 in monenergetic electron beams.
- EXACT is now the default boundary crossing algorithm in BEAMnrc (but not in DOSXYZnrc). This was changed from PRESTA-I after it was shown[12] that the PRESTA-I BCA can result in dose overprediction by up to 2.5% in a CHAMBER phantom.
- The BEAMnrc and DOSXYZnrc GUI's now give the user access to various electron impact ionization theories (only applicable for keV X-Rays) and photon cross-section data other than Storm-Israel (PEGS4).
- The parameter $BDY_TOL in $OMEGA_HOME/beamnrc/beamnrc_user_macros.mortran, which defines the "fuzzy boundary" used in the HOWFAR routines of various CMs to ensure that particles are actually transported beyond region boundaries was changed from 1E-4 cm to 1E-5 cm. The previous value resulted in serious underestimates of contaminant electron dose in high-energy photon beams when the JAWS CM was used.
- Introduced a more efficient range rejection scheme that works together with directional bremsstrahlung splitting (DBS). It can increase the efficiency of DBS by ∼20%.
- Source 19 (circular beam with Gaussian distribution in X-Y) can now have user-specified angular spread.
- Introduced electron splitting for phase space and BEAMnrc simulation sources in DOSXYZnrc. This is used in conjunction with photon splitting (n_split) to ensure that higher-weight contaminant electrons do not compromise statistics.

19.5 Changes from BEAMnrc05 to BEAMnrc06

Apart from many smaller bug fixed and modifications, the major changes between the BEAMnrc05 and BEAMnrc06 distributions can be summarised as follows:

- Added the DYNVMLC CM, designed to simulate the Varian Millenium class of multi-leaf collimators. See the BEAM User's Manual for more details. Original coding is from Emily Heath at McGill University.
- Zdenko Sego at Carleton University has fixed and enabled the the use of multiple source model as sources in BEAMnrc (ISOURC=31). This option was disabled for many releases.
- Fixed a major bug in the phase space source for DOSXYZnrc. When particles were recycled, their Z direction cosine (wsrc) was not saved from one use to another, resulting in particles that either did not hit the geometry or went in non-physical directions.

Simulations either crashed or produced depth-dose curves wildly different from those expected.

- Added many `USER` macros to the BEAM code which allow a user to easily customize the `AUSGAB` routine, input parameters, data analysis, global variables, etc. These macros are changed in `beamnrc_user_macros.mortran`

- Custom user inputs can appear between the delimiters `:start user inputs:` and `:stop user inputs:` in the BEAMnrc `.egsinp` file and the GUI will handle them (but not give you access to them).

- Fixed some bugs in the new MLCE CM. Among them: it was possible for particles to be trapped in "dead air", not contained within the boundaries of any leaf side-surface, near the bottom of the MLC. Also, leaf overlap restrictions on input were too strict.

- Fixed some bugs in the JAWS CM which resulted in unnecessary warning messages being output. Also, fixed one bug which resulted in occasional particles being discarded when transported backwards (W¡0) to the top of the jaws.

19.6 Changes from `BEAMnrc03` to `BEAMnrc05`

Apart from many smaller bug fixed and modifications, the major changes between the `BEAMnrc03` and `BEAMnrc05` distributions can be summarised as follows:

- Ported the entire set of codes to work with the new multi-platform EGSnrc system (EGSnrcMP).

- Electron impact ionization added to the EGSnrc system.

- Added directional bremsstrahlung splitting (DBS) which increases the efficiency for photon accelerators by more than a factor of 5.

- Included the new CM MLCE which was mostly coded by Nick Reynaert at the University of Ghent.

- Added the BEAM shared library source (source 9) to DOSXYZnrc and other EGSnrc user codes. This required some reorganization of the BEAM code to allow it to be compiled as a shared library for use as a source in DOSXYZnrc and other user codes. Now, `beamnrc.mortran` contains subroutines which are called by either `beam_main.mortran` (regular simulation) or `beam_lib.mortran` (shared library source). Also involved changes to some phase space writing macros in `phsp_macros.mortran` so that, when the BEAM simulation is used a source, particles are dumped into a source "bin" instead of written to a phase space file.

- Introduced a new parallel processing approach which optimizes performance on a system with mixed CPU speeds and utilization factors.

- Updated DICOM CT image reading routine, `readCT_DICOM.c`, used by `ctcreate`. The routine is now independent of libraries from the DICOM CTN (central test node) and is compatible with the latest DICOM format. Thanks to Nick Reynaert at the University of Ghent for sending us the original code.

- Changed uncertainty analysis in BEAMDP so that it uses the history-by-history method used in all user codes.

- Added an option for source 1 in BEAMnrc (isotropically-radiating point source) to be collimated as a rectangle anywhere on the surface of CM 1. Previously, this could only be collimated as a square centred at Z=0.

- Modifications to BEAMnrc, DOSXYZnrc and BEAMDP GUI's to allow them to run codes on Windows.

- Fixed a bug in JAWS in which particles that should have been transported right through the tip of the JAWS (since they were within boundary tolerance of the edge) erroneously had their region numbers assigned to the JAW material. This resulted in many warnings.

- Fixed bugs in CONS3R CM in related to particle region numbers being incorrect when a particle is being transported close to the boundary of the cone.

- Modified the HOWFAR routines in all BEAM CMs so that if a particle is leaving the CM (through the top or bottom) with `USTEP`=0, then `USTEP` is given a small positive value ($1.E^{-16}$) to ensure that AUSGAB gets called and the particle gets scored (if there is a scoring plane at this boundary). This bug was causing significant errors in the phase space file in perverse situations.

19.7 Changes from `BEAMnrc02` to `BEAMnrc03`

- Enabled the RANLUX random number generator in BEAMnrc and DOSXYZnrc. Can now use either RANLUX or RANMAR.

- Fixed 3 bugs in VARMLC and added the `IGNOREGAPS` input which increases the efficiency of range rejection for particles deep in the leaves (i.e., far away from the openings) by ignoring the air gaps between the leaves.

- Fixed a bug in CONS3R which caused particles to be discarded unnecessarily (with a warning) when being transported close to a conical boundary. The region error counter was not being reset properly with each new initial history.

- Put a restriction that all radii in CONESTAK be $\geq$ `$BDY_TOL`. Otherwise, a region check fails and the code could enter an endless loop.

19.8 Changes from BEAM00 to BEAMnrc02

- BEAMnrc is based on the EGSnrc system.
- The statistical analysis package in BEAMnrc is based on the history by history technique which greatly reduces the uncertainty on the uncertainties assigned. See the report on history by history statistics in BEAMnrc and DOSXYZnrc [17] for a complete discussion.
- Range rejection within BEAMnrc is based on the EGSnrc calculation of range at each step but still does range rejection to ECUT, unlike the default vesion of EGSnrc which does it to AE.
- Added sources 7 and 8 to DOSXYZnrc. Source 7 is a parallel rectangular beam incident from multiple, user-selected angles. Source 8 is a phase space source incident from multiple, user-selected angles. This allows modelling of arc therapy.
- Added an option to DOSXYZnrc which allows the the user to output a `.egsphant` file from non-CT data. This allows you to display isodose contours using `dosxyz_show` in non-CT phantoms.
- Added a feature to the GUI for the JAWS CM which allows the X and Y positions of the JAWS to be set to give an arbitrary rectangular field size at a given SSD by diverging from an arbitrary focal point on the axis. Note that this is useful only for the photon jaw field size, not the electron field size (for which the jaws are typically set at a larger size).
- Upgraded VARMLC so that it can now simulate two different classes of multi-leaf collimators: those in which the tongue and groove do not extend beyond the top and bottom of the leaves (the original class which VARMLC was designed to simulate); and those in which the tongue and groove extend to either the top or bottom of the leaves (the new class).
- Corrected some serious bugs in VARMLC. Among them, the distance along the particle trajectory to the nearest vertical boundary parallel to the leaf opening direction was not calculated correctly. This was because some cases were omitted in `$VARMLC_HOWFAR`.
- Changed all variables that store summed quantities in BEAMDP from REAL*4 to REAL*8. This overcomes a potential problem when quantities from a large ($>$ 1 million) number of particles are being summed. Previously, it was possible for summed quantities to become so large that they consumed all decimal places in REAL*4 variables, and any further contributions from individual particles (especially those with low weight) were not included in the sum.
- Fixed a bug in the JAWS CM in which particles were not transported into the jaw material from the air in front of the jaws with a `$BDY_TOL` overshoot. This resulted in endless loops and `WARNING` messages when the transport into the jaws was near the jaw tips.

- Fixed a fairly major bug in DOSXYZnrc in which the `ISMOOTH` and `NRCYCL` options for phase space sources could not be used at the same time or else they caused errors. Previously, incident particle positions and direction cosines were recycled in the DOSXYZnrc phantom coordinate system. When smoothing was also turned on, then these particles were also redistributed in the phantom coordinate system. This was an error, since redistribution is only meaningful in the coordinate system of the phase space source. We solved the problem by taking care of particle recycling and redistribution before rotating the source into its position in the DOSXYZnrc phantom coordinate system.

- Fixed a bug in BEAM in which particles were scored at a scoring plane even though they were scattered back into the CM before actually crossing the scoring plane.

- Fixed a bug in CONS3R CM. Previously, particles with a mismatch between region number and radial position were transported `MIN(USTEP,1.0E-5)` and their region number was automatically changed. This caused problems for particles that were crossing a radial boundary but ended up just short of it (due to roundoff error). In this case the region number was reset to where the particle was coming from even though, clearly, the `MIN(USTEP,1.0E-5)` step took it into the new region. We now check the radial position of the particle after the `MIN(USTEP,1.0E-5)` step before changing the region number.

20 Acknowledgements

The original BEAM code was developed as part of a major collaboration with Rock Mackie's group at the University of Wisconsin under a grant from the National Institutes of Health (R01-CA52692). At NRC there have been a wide variety of people who supported the project directly or indirectly. Michel Proulx has been our able computer system manager; Alex Bielajew was involved at early stages and helped maintain the NRC EGS4 system; Jiangshen Sun did a variety of quality assurance checks on the input/output routines for the CMs. Two Research Associates played major roles in the development of the BEAM system, Bruce Faddegon was deeply involved with the initial design of the software system that evolved into BEAM and Charlie Ma was a major developer of many parts of the code system. The following graduate students made major contributions to many aspects of the coding and overall development of the code: George Ding, Jiansu Wei, Geoff Zhang and Daryoush Sheikh-Bagheri. Most of the students and RAs have at times been authors of earlier versions of this manual and their contributions to the manual remain in many places.

Elsayed Ali of Carleton University developed the BCSE variance reduction coding and drafted section 6.5 of the manual which describes this technique.

The OMEGA/BEAM installation wizard was created by Ernesto Mainegra-Hing at the NRC.

The CM `ARCCHM` was contributed by Marv Glass of University of Wisconsin.

The first version of the CM `MLCQ` was contributed by by Hugo Palmans and Kristiaan De Vlamynck of the University of Gent, Belgium.

The first version of the `VARMLC` CM was written by Ajay Kapur with Charlie Ma at Stanford University.

The CM called MLCE was mostly coded by Nick Reynaert at the University of Ghent.

We have had many users point out and sometimes correct bugs. In particular we wish to recognise John Antolak and those he worked with at MDACC in Houston for their many clear bug reports, the first version of source 19 and especially the current version of the `BLOCK` CM.

The source code and bug reports explicitly mention the contributions of many others.

21 References

[1] I. Kawrakow and D. W. O. Rogers. The EGSnrc Code System: Monte Carlo simulation of electron and photon transport. Technical Report PIRS–701 (4th printing), National Research Council of Canada, Ottawa, Canada, 2003.

[2] I. Kawrakow, E. Mainegra-Hing, and D. W. O. Rogers. EGSnrcMP: the multi-platform environment for EGSnrc. Technical Report PIRS–877, National Research Council of Canada, Ottawa, Canada, 2003.

[3] D. W. O. Rogers, B. A. Faddegon, G. X. Ding, C.-M. Ma, J. Wei, and T. R. Mackie. BEAM: A Monte Carlo code to simulate radiotherapy treatment units. *Med. Phys.*, 22:503 – 524, 1995.

[4] J. A. Treurniet, B. R. B. Walters, and D. W. O. Rogers. BEAMnrc, DOSXYZnrc and BEAMDP GUI User's Manual. *NRC Report PIRS 0623(rev C)*, 2004.

[5] I. Kawrakow. Accurate condensed history Monte Carlo simulation of electron transport. I. EGSnrc, the new EGS4 version. *Med. Phys.*, 27:485 – 498, 2000.

[6] D. W. O. Rogers, I. Kawrakow, J. P. Seuntjens, and B. R. B. Walters. NRC User Codes for EGSnrc. Technical Report PIRS–702, National Research Council of Canada, Ottawa, Canada, 2000.

[7] D. W. O. Rogers, C.-M. Ma, G. X. Ding, and B. Walters. BEAM Users Manual. *NRC Report PIRS 509a*, 1995.

[8] D. W. O. Rogers, C.-M. Ma, G. X. Ding, and B. Walters. BEAM Users Manual. *NRC Report PIRS 509(a)revB*, 1997.

[9] D. W. O. Rogers, C.-M. Ma, G. X. Ding, B. Walters, D. Sheikh-Bagheri, and G. G. Zhang. BEAM98 Users Manual. *NRC Report PIRS 509(a)revC*, 1998.

[10] D. W. O. Rogers, C.-M. Ma, G. X. Ding, B. Walters, D. Sheikh-Bagheri, and G. G. Zhang. BEAMnrc Users Manual. *NRC Report PIRS 509(a)revF*, 2001.

[11] D. W. O. Rogers, B. Walters, and I. Kawrakow. BEAMnrc Users Manual. *NRC Report PIRS 509(a)revH*, 2004.

[12] B. R. B. Walters and I. Kawrakow. Technical note: Overprediction of dose with default PRESTA-I boundary crossing in DOSXYZnrc and BEAMnrc. *Med. Phys.*, 34:647 – 650, 2007.

[13] C.-M. Ma, P. Reckwerdt, M. Holmes, D. W. O. Rogers, and B. Geiser. DOSXYZ Users Manual. *NRC Report PIRS 509b*, 1995.

[14] Blake Walters and D. W. O. Rogers. QA for the BEAM System; Component Modules, Variance Reduction Options and Source Routines. *NRC Report PIRS–509k*, 1995.

[15] B. R. B. Walters, I. Kawrakow, and D. W. O. Rogers. DOSXYZnrc Users Manual. *NRC Report PIRS 794 (rev B)*, 2005.

[16] J. A. Treurniet and D. W. O. Rogers. EGS_Windows4.0 User's Manual. *NRC Report PIRS–0669*, 1999.

[17] B. R. B. Walters, I. Kawrakow, and D. W. O. Rogers. History by history statistical estimators in the BEAM code system. *Med. Phys.*, 29:2745 – 2752, 2002.

[18] Iwan Kawrakow, D. W. O Rogers, and B.R.B. Walters. Large efficiency improvements in BEAMnrc using directional bremsstrahlung splitting. *Med. Phys.*, 31:2883 – 2898, 2004.

[19] D. W. O. Rogers, I. Kawrakow, J. P. Seuntjens, B. R. B. Walters, and E. Mainegra-Hing. NRC User Codes for EGSnrc. Technical Report PIRS–702(RevB), National Research Council of Canada, Ottawa, Canada, 2003.

[20] J. A. Treurniet and D. W. O. Rogers. BEAM, DOSXYZ and BEAMDP GUI User's Manual. *NRC Report PIRS 0623(rev A)*, 1999.

[21] Daryoush Sheikh-Bagheri and D. W. O. Rogers. BEAM Example: A 16 MV photon beam. *NRC Report PIRS–509i*, 1995.

[22] G. G. Zhang and D. W. O. Rogers. BEAM Example: A 10 MeV electron beam. *NRC Report PIRS 509h*, 1995.

[23] Blake Walters and D. W. O. Rogers. BEAM Example: Depth-Dose in a Phantom. *NRC Report PIRS–509j*, 1995.

[24] J. Lobo and I. A. Popescu. Two new dosxyznrc sources for 4d monte carlo simulations of continuously variable beam configurations, with applications to rapidarc, vmat, tomotherapy and cyberknife. *Phys. Med. Biol.*, 55:4431 – 4443, 2010.

[25] A. F. Bielajew, R. Mohan, and C. S. Chui. Improved bremsstrahlung photon angular sampling in the EGS4 code system. *National Research Council of Canada Report PIRS-0203*, 1989.

[26] D. W. O. Rogers and A. F. Bielajew. Monte Carlo techniques of electron and photon transport for radiation dosimetry. In K. R. Kase, B. E. Bjärngard, and F. H. Attix, editors, *The Dosimetry of Ionizing Radiation,Vol III*, pages 427 – 539. Academic Press, 1990.

[27] E. S. M. Ali and D. W. O. Rogers. Efficiency improvements of x-ray simulations in EGSnrc user-codes using Bremsstrahlung Cross Section Enhancement (BCSE). *Med. Phys.*, 34:2143 – 2154, 2007.

[28] B. R. B. Walters. Increasing efficiency of BEAMnrc-simulated Co-60 beams using directional source biasing. *Med. Phys.*, 42:5817 – 5827, 2015.

[29] G. Mora, A. Maio, and D. W. O. Rogers. Monte Carlo simulation of a typical ^{60}Co therapy source. *Med. Phys.*, 26:2494 – 2502, 1999.

[30] E. Mainegra-Hing and I. Kawrakow. Efficient x-ray tube simulations. *Med. Phys.*, 33:2683 – 2690, 2006.

[31] R. Capote, R. Jeraj, C.-M. Ma, D.W.O. Rogers, F. Sanchez-Doblado, J. Sempau, J. Seuntjens, and J.V. Siebers. Phase-Space Database for External Beam Radiotherapy. *IAEA Report INDC(NDS)-0484*, 2005.

[32] M. Lüscher. A portable high-quality random number generator for lattice field theory simulations. *Computer Phys. Commun.*, 79:100 – 110, 1994.

[33] F. James. RANLUX: A Fortran implementation of the high-quality pseudorandom number generator of Lüscher. *Computer Phys. Commun.*, 79:111 – 114, 1994.

[34] G. Marsaglia and A. Zaman. A new class of random number generators. *Annals of Applied Probability*, 1:462 – 480, 1991.

[35] G. Marsaglia, A. Zaman, and W. W. Tsang. Toward a universal random number generator. *Statistics and Probability Letters*, 8:35 – 39, 1990.

[36] D. W. O. Rogers. Low energy electron transport with EGS. *Nucl. Inst. Meth.*, 227:535 – 548, 1984.

[37] B. J. Foote and V. G. Smyth. The modelling of electron multiple-scattering in EGS4/PRESTA and its effect on ionization-chamber response. *Nucl. Inst. Meth.*, B100:22 – 30, 1995.

[38] A. F. Bielajew and D. W. O. Rogers. PRESTA: The Parameter Reduced Electron-Step Transport Algorithm for electron Monte Carlo transport. *Nuclear Instruments and Methods*, B18:165 – 181, 1987.

[39] H. W. Koch and J. W. Motz. Bremsstrahlung cross-section formulas and related data. *Rev. Mod. Phys.*, 31:920 – 955, 1959.

[40] S. M. Seltzer and M. J. Berger. Bremsstrahlung spectra from electron interactions with screened atomic nuclei and orbital electrons. *Nucl. Inst. Meth. Phys. Res. B* **12**, 12:95 – 134, 1985.

[41] S. M. Seltzer and M. J. Berger. Bremsstrahlung energy spectra from electrons with kinetic energy from 1 kev to 10 gev incident on screened nuclei and and orbital electrons of neutral atoms with z = 1-100. *Atomic Data and Nuclear Data Tables*, 35:345–418, 1986.

[42] ICRU. Stopping powers for electrons and positrons. ICRU Report 37, ICRU, Washington D.C., 1984.

[43] O. Klein and Y. Nishina. Über die Streuung von Strahlung durch freie Elektronen nach der neuen relativistischen Quantendynamik von Dirac. *Z. für Physik*, 52:853–868, 1929.

[44] R. Ribberfors. . *Phys. Rev. B*, 12:2067, 1975.

[45] J. W. Motz, H. A. Olsen, and H. W. Koch. Pair production by photons. *Rev. Mod. Phys.*, 41:581 – 639, 1969.

[46] F. Sauter. Über den atomaren Photoeffekt in der K-Schale nach der relativistischen Wellenmechanik Diracs. *Ann. Physik*, 11:454 – 488, 1931.

[47] E. Storm and H. I. Israel. Photon cross sections from 1 keV to 100 MeV for elements Z=1 to Z=100. *Atomic Data and Nuclear Data Tables*, 7:565 – 681, 1970.

[48] J. H. Hubbell and I. Øverbø. Relativistic atomic form factors and photon coherent scattering cross sections. *J. Phys. Chem. Ref. Data*, 9:69, 1979.

[49] ICRU. ICRU Report Committee Activities. *in ICRU News(ICRU, Bethesda MD), June*, page 20, 1990.

[50] I Kawrakow. Electron impact ionization cross sections for egsnrc. *Med. Phys. (abstract)*, 29:1230, 2002.

[51] E. S. M. Ali and D. W. O. Rogers. Implementation of photonuclear attenuation in EGSnrc. *Carleton University Technical Report CLRP 12-01*, 2012.

[52] C. Borges, M. Zarza-Moreno, E. Heath, N. Teixeira, and P. Vaz. Monte Carlo modeling and simulations of the High Definition (HD120) micro MLC and validation against measurements for a 6 MV beam. *Med. Phys.*, 39(1):415–423, 2012.

[53] D. W. O. Rogers, S. Duane, A. F. Bielajew, and W. R. Nelson. Use of ICRU-37/NBS radiative stopping powers in the EGS4 system. *National Research Council of Canada report PIRS-0177*, 1989.

Appendix A:
Specifications for Component Modules for BEAMnrc

D. W. O. Rogers, G.X. Ding, C.M. Ma and B.A. Faddegon
Ionizing Radiation Standards
National Research Council of Canada Ottawa

Printed: April 15, 2021

Abstract

This report is for BEAMnrc code developers and is not needed for those who just want to use the code. The report specifies what each component module must do, how it must do it, the tools available and the documentation to be followed. There is a separate report describing the QA to be done on a `CM` if it is modified[14]. It is assumed that the reader is an experienced `EGSnrc` user.

A.1 Overview

BEAMnrc is a general purpose code for doing Monte Carlo simulations of radiotherapy beams. One of its design features is that each part of the accelerator or source unit is considered to be a single component module which takes up an horizontal slab portion of the accelerator. These component modules are re-usable and are completely independent. They must communicate with the rest of the system in certain well specified ways. The purpose of this document is to list all the specifications of such a component module including the documentation required for such a component module. The quality assurance required for each component module is described in an associated report by Walters and Rogers (QA for the BEAM System: Component Modules, Variance Reduction Options and Source Routines).

Two MORTRAN source files make up each component module. The MORTRAN macros specific to component module CMNAME are contained in CMNAME_macros.mortran. These macros are used by BEAMnrc proper and/or the EGS subroutines and/or the component module subroutines. The set of subroutines specific to component module CMNAME are contained in CMNAME_cm.mortran.

When you build an accelerator, beam_build ensures that within each CM the string $CMNAME is replaced by a user-supplied identifier everywhere it appears in the CMs mortran source code. The subroutine and macro names, COMMON name, and variable names specific to a given component module all either begin or end with $CMNAME. Every component module must have an unambiguous identifier. This prevents duplication of any of these names when multiple component modules are used in a simulation. Using this convention, the same component module may appear many times in a BEAMnrc code.

CMNAME_macros.mortran
The file CMNAME_macros.mortran contains at least two MORTRAN replacement macros used by the component module subroutines. The macro COMIN/CM_$CMNAME/ is the COMMON block replacement macro for component module CMNAME. The $CMNAME_CM_HOWNEAR macro is required by MORTRAN SUBROUTINE ELECTR when the HOWNEAR replacement macro of BEAMnrc proper is used. This macro calculates the distance from the current location of the particle to the nearest boundary in the component module with identifier CMNAME. The macro is also usually used in subroutine HOWFAR_$CMNAME as well. Note that in most of the existing CMs, the $CMNAME_CM_HOWNEAR macro is just a call to a HOWNEAR_$CMNAME subroutine. The subroutine is located in the file $CMNAME_cm.mortran and performs the actual calculation of perpendicular distance to the nearest boundary. The use of a subroutine HOWNEAR_$CMNAME (with a call to it from the $CMNAME_CM_HOWNEAR macro) is recommended in EGSnrc because variables associated only with the calculation of HOWNEAR need only be declared inside the subroutine. Otherwise, these variables must be appended to the $DEFINE-LOCAL-VARIABLES-ELECTR macro in EGSnrc.

CMNAME_cm.mortran
The CMNAME_cm.mortran file contains the following subroutines written for each component module:

- HOWFAR_$CMNAME – a standard EGSnrc SUBROUTINE HOWFAR for a component module CMNAME. It is used during the simulation for defining the geometry via boundary

checking and setting region-dependent parameters.

- HOWNEAR_$CMNAME – a subroutine for component module $CMNAME which calculates the perpendicular distance to the nearest boundary (ie not along the particle trajectory). This is called from the macro $CMNAME_CM_HOWNEAR. See above for a more detailed description of the relationship between the HOWNEAR subroutine and the HOWNEAR macro.

- WHERE_AM_I_$CMNAME – used to determine region of particle upon entry into component module CMNAME.

- INPUT_$CMNAME – prompts for and digests input from the interactive user or the parameter-definition file (*i.e.* file.egsinp) for information related to component module CMNAME.

- ISUMRY_$CMNAME – writes summary of input for component module CMNAME to listing.

A.2 Writing Component Modules

Source for all CMs must be written in MORTRAN3. See the EGS4 manual, SLAC265, for a specification of MORTRAN3.

Component module names are capitalized, from 1 to 8 characters long, and are unambiguous, for example, the component module SLABS is already in use and new component modules must not have this name. When writing component modules it is useful to adhere to the following established convention for code formats:

- MORTRAN3 replacement macros specific to the component module are placed in CMNAME_macros.mortran where CMNAME is the name of the component module.

- Indent 3 spaces for all MORTRAN3 LOOPS, and FORTRAN DO and IF statements.

- Capitalize all MORTRAN3 (this is essential) and FORTRAN code.

- Format for replacement macros should be as follows to allow the CMtoc utility to be used: REPLACE {text} WITH {<CR>

- Incorporate extensive documentation at the start of each component module subroutine file, following the conventions established in SLABS_cm.mortran.

A.2.1 Tips

- To prepare for writing a new component module or modifying an existing one, read the in-line documentation at the beginning of beamnrc.mortran, beamnrc_user_macros.mortran, and beamnrc_cm_macros.hdr. Next read all of SLABS_cm.mortran and SLABS_macros.mortran. Also look at the variables in COMMON/CMs/, which are of general use in writing component modules.

- Write SUBROUTINE HOWFAR_$CMNAME in CMNAME_cm.mortran first, writing the COMIN/CM_$CMNAME macro, which defines all the variables needed, at the same time. The outer boundary of the component module is checked in the HOWFAR subroutine in BEAMnrc proper, and need not be checked here.

- Next write the SUBROUTINE HOWNEAR_$CMNAME for component module CMNAME in CMNAME_cm.mortran. This macro returns the nearest distance to any boundary from the current particle position (DNEAR). Then, write a macro in $CMNAME_macros.mortran called $CMNAME_CM_HOWNEAR which is just a call to subroutine HOWNEAR_$CMNAME. If your geometry is too complex to calculate the nearest perpendicular distance to a boundary, then just write the line:
REPLACE {$CMNAME_CM_HOWNEAR(#)} WITH { {P1}=0; }
in the $CMNAME_macros.mortran file, and omit the subroutine HOWNEAR_$CMNAME from CMNAME_cm.mortran.

- Write the rest of the component module subroutines for CMNAME in the order: WHERE_AM_I_$CMNAME, SUBROUTINE INPUT_$CMNAME followed by ISUMRY_$CMNAME.

- Use the macro and subroutine files from an established component module such as SLABS or CONESTAK as a template for the new component module. Much of the code can be used for the new component module with only minor modifications. This does not generally apply to the SUBROUTINE HOWFAR_$CMNAME, the most difficult of the component module subroutines to write.

- When the coding is completed, follow established quality assurance procedures closely. The procedures contained in the QA document listed at the beginning of this report are the minimum.

- Your code will be read and modified by others, so ensure that it is clearly laid out, extensively documented, and conforms closely to the established conventions.

A.3 Specifications–CMNAME_macros.mortran

For every component module, this file must be created. It contains at least the two macros specific to the particular CM specified below, but the user may write as many as convenient.

Naming conventions in CMNAME_macros.mortran:

- Use $CMNAME throughout (*eg.*$SLABS) except in those cases (usually output) where you want the original CM name to be used, usually to identify the type of CM for output purposes.

- The component module name ($CMNAME) appears at the start of all MORTRAN3 macro names (*eg.*$SLABS_CM_HOWNEAR).

- The component module name ($CMNAME) appends all local variable names (*eg.* ICM_$SLABS).

A.3.1 COMIN/CM_$CMNAME macro

This macro defines the COMIN which contains the values associated with this specific CM and a few links to the rest of the simulation. Nothing is mandatory since only the writer of a given CM ever uses these variables. Typically we find it useful to define at least the following variables:

- ICM_$CMNAME: an index specifying which CM this is, starting from 1 nearest the source. It is usually set in
  SUBROUTINE INPUT_$CMNAME.
- IR_$CMNAME: the local region number within the CM. This is often used in the HOWFAR and HOWNEAR subroutines.
- IRSTART_$CMNAME: first absolute region number for this CM. It is usually set in SUBROUTINE INPUT_$CMNAME and equals IR_start_CM(ICM) from COMIN/CMs.
- IREND_$CMNAME: last absolute region number for this CM. It is usually set in SUBROUTINE INPUT_$CMNAME and equals IR_start_CM(ICM+1)-1 from COMIN/CMs.
- TITLE_$CMNAME: title(60) (character*1). It is usually set in
  SUBROUTINE INPUT_$CMNAME.
- N_GAP_$CMNAME: flag, = 0 if no air gap this CM, = 1 if air gap at top of CM. It is usually set in SUBROUTINE INPUT_$CMNAME.

As well, all geometric parameters associated with this CM are defined in this COMIN and filled in SUBROUTINE INPUT_$CMNAME usually. These MUST have the string _$CMNAME appended at the end of their name to ensure unique names if COMINs are defined more than once.

A.3.2 $CMNAME_CM_HOWNEAR(#) macro

This macro is usually just a call to SUBROUTINE HOWNEAR_$CMNAME (See Section A.4.5 below), which returns the perpendicular distance from the particle to the nearest boundary. However, this macro is the only means by which the HOWNEAR subroutine is called and is used in EGSnrc and sometimes in the SUBROUTINE HOWFAR_$CMNAME.

A.4 Specifications–CMNAME_cm.mortran

Naming conventions in CMNAME_cm.mortran:

- Use $CMNAME throughout, that is, the name of the component module should not be directly written into the code. This permits a change of identifier simply by changing the .module file.
- $CMNAME appends all subroutine names.

- $CMNAME appears at the start of all MORTRAN macro names.
- $CMNAME appends all local variable names.

General Requirements
At the top of the source a set of comments on records starting with "I> must define the geometry accurately and the inputs required from unit 5. Use of the "I> comment ensures that we will be able to pick up the description of the input for the next edition of the users manual (*i.e.*this is the primary documentation for input - make it clear).

A.4.1 SUBROUTINE INPUT_$CMNAME

On being called, the following information is available to the routine (all via COMIN/CMs except NMED which is in COMIN/GEOM):

- ICM: the number of this CM, starting from 1 for smallest z.
- IR_start_CM(ICM): the absolute region number this CM starts at. It is set by previous CM's input routine and for the ICM=1 case it is set in main (to 2 since region 1 is the exterior).
- RMAX_CM(ICM): outer boundary of this CM, read in by main for each CM. It is either the radius or 1/2 the side for a square boundary.
- NMED: The number of media *already* asked for in any given simulation.
- MEDIA(24,I),I=1,NMED: names of media already asked for.

The following variables from COMIN/CMs must all be set, not necessarily from user input (*i.e.* the code may define them for the user).

- RMAX_CM_FLAG(ICM), flag for each CM which specifies what quantity RMAX_CM is for this CM (RMAX_CM is set by MAIN). If flag = 0, not used; flag = 1, it is radius of outer boundary of CM; flag = 2, it is distance to edge of square outer boundary of CM (1/2 of side). It is used in main SUBROUTINE HOWFAR to check the outer boundaries, after the individual HOWFAR_$CMNAME routine has done its work. One should avoid redundant calculations.
- IERR_GEOM(ICM), flag which must be initialized to zero and set non-zero to flag input problem. Set < 100, it specifies number of errors detected within CM, >100 specifies that the CM overlaps the one above it. MAIN routine will exit before transport, but input is continued and checked.
- Z_min_thick(ICM,j) and MED_min_thick(ICM,j): minimum thickness in cm and medium number of up to j = 5 regions in ICM for an electron going through CM # ICM. It is used for range rejection. It is set in each CM input routine. Region with j=1 is closest to the bottom plane. Often only one is needed or possible and this is given by the total thickness of the CM and medium is air (*i.e.* MEDIUM 1).

- Z_min_CM(ICM+1) for next CM is set as the back of this CM.
- Z_gap_THICK(ICM),thickness of air gap at front of CM, if it exists. If Z_gap_THICK(ICM)= 0, then there is no air gap at the front of this CM.
- IR_start_CM(ICM+1): first absolute region for next CM, set once the number of regions in the current CM is known.

We find it useful to set the variables in COMIN/CM_$CMNAME listed in section A.3.1.

All variables required by the geometry for this CM (with _$CMNAME at the end of the name) must be input and stored in COMIN/CM_$CMNAME.

The following EGSnrc variables must be set:

- ECUT(IRA) & PCUT(IRA) for all regions IRA in the CM (including the air gap). If not set, the defaults are AE and AP for the medium in each region.
- MED(IRA) the medium index for every region must be set
- MEDIA(24,I) must be loaded for new MEDIA that are asked for. Note the macro $MED_INPUT($CMNAME) handles this automatically. This macro requires NMED to contain the current number of different media that have already been read in and stored in MEDIA.
- RHOR(IRA) the density in each region in g/cm^3. It need only be set by the user if it is different from the value included in the PEGS4 data set. After the call to HATCH, it will include this latter value if left 0.0 prior to the HATCH call.

The following variables in COMIN/SCORE must be set.

- DOSE_ZONE(IRA) = 0 if no dose scored, otherwise =dose zone that dose from this region scored in. Note that one dose zone can include the dose deposited in many geometric regions.
- NDOSE_ZONE, largest dose scoring zone number assigned so far.
- IREGION_to_BIT(IRA): mapping from absolute regions to LATCH bits (*i.e.* which bit in variable LATCH is this region associated with).
- MAX_BIT: current largest bit being set.

The following variable in COMIN/GEOM must be set: NREG: total number of regions in the geometry model up to and including this CM (in COMIN/GEOM). It should equal IR_start_CM(ICM+1)-1.

A.4.2 SUBROUTINE ISUMRY_$CMNAME

This routine, which is called after HATCH, must summarize all data related to the use of this CM in any given run. Specifically, it must be possible to completely reconstruct the input file from the information in the output listing. Also the Revision No. of the CM must be echoed to the listing. Use already existing CMs as examples.

All output MUST fit in 80 columns and allow FORTRAN carriage control to be used for printing purposes (*i.e.* col 0 defines double spacing, new lines *etc.*). However, for screen outputs it is better to just avoid these FORTRAN controls.

It must also increment the mass of each scoring zone by the mass of each region in this CM which is in that zone. Specifically,
```
AMASS(IDD) = AMASS(IDD) + RHOR(IRA)*VOLUME(IRA)
```
where IDD is the scoring zone associated with region IRA, (IDD = DOSE_ZONE(IRA))

A.4.3 SUBROUTINE HOWFAR_$CMNAME

This is a more or less standard EGSnrc SUBROUTINE HOWFAR which applies just to this CM. See examples from already coded CMs.

The routine does NOT consider the RMAX_CM boundaries since these are handled afterwards in the main SUBROUTINE HOWFAR.

As a particle leaves the CM to enter the CMs on either side, the
SUBROUTINE WHERE_AM_I(IICM,IDIR) is useful. This routine is called from
SUBROUTINE HOWFAR_$CMNAME when a particle reaches the front (IDIR = 1) or back (IDIR = -1) of a component module. The index of the new component module, ICMNEW (passed in COMIN/CMs/), is determined and the appropriate SUBROUTINE WHERE_AM_I_$CMNAME for the new CM is called to determine the new region number, IRNEW.

A.4.4 SUBROUTINE WHERE_AM_I_$CMNAME

Subroutine WHERE_AM_I_$CMNAME determines the new region number when a particle traverses the front or back boundary of a component module. Whenever a particle is to be transported to a component module boundary in HOWFAR, the subroutine WHERE_AM_I is called. The current component module, ICM, and particle direction (IDIR, backwards or forwards) are transferred to WHERE_AM_I WHERE_AM_I determines which component module the particle is about to enter and calls the WHERE_AM_I_$CMNAME subroutine for that component module, transferring the particle direction. The region number that the particle is about to enter, IRNEW, is determined in WHERE_AM_I_$CMNAME from the knowledge of which surface the particle is entering through (front if IDIR=1, back if IDIR=-1) and the (X,Y) coordinates of the particle.

A.4.5 SUBROUTINE HOWNEAR_$CMNAME

This subroutine calculates the perpendicular (minimum) distance from the particle position to the nearest region boundary (stored globally in the variable `DNEAR`). It is always called using the `$CMNAME_CM_HOWNEAR` macro (See Section A.3.2 above) See already coded CMs for examples of how to code this subroutine.

Note: this subroutine does NOT check against the `RMAX_CM` boundary (nor does the main `CALL HOWNEAR` macro in BEAMnrc). This means that particles do not approach the outer boundary with lateral correlations turned off (if using the PRESTA-I boundary crossing algorithm) or in single-scattering mode (if using EXACT boundary crossing algorithm). It also means that the outer boundary is not used in range rejection.

A.5 Specifications–COMIN/CMs/

This `COMIN` contains the geometrical and range rejection information of interest to all component modules:

```
E_min_out(ICM)     minimum energy of electron leaving a CM ICM which can
                   reach the nearest downstream scoring region with energy
                   greater than ECUT in the scoring region. For use in range
                   rejection.  Set in MAIN because needs info from all CMs
                   past the current one.
MAX_CMs            number of CMs.
MED_IN             24-character name of last medium input in INPUT_$CMNAME.
ICM                CM index, incremented before call to INPUT_$CMNAME, set in
                   SRCHST and in HOWFAR during particle transport.
ICMNEW             Next CM, set in WHERE_AM_I, different than ICM (only?) if
                   particle transported to CM boundary.
ICM_to_SCORE(ICM)  scoring plane associated with ICM, 0 => none. Set in
                   main based on input IPLANE_to_CM values.
IERR_GEOM(ICM)     geometry-checking flag for each CM, 0=>no errors detected,
                   1-99 specifies number of errors detected within CM,
                   >100 specifies that CM above overlaps
IR_start_CM(ICM)   region number of first region in CM, set by previous CM,
                   read in subroutine INPUT_$CMNAME.
IR_to_CM(IR)       pointer used in HOWFAR which denotes the CM region IR
                   is in.  It is set in MAIN.
RMAX_CM(ICM)       outer boundary of treatment head, particles are discarded
                   if they move outside of this boundary.  Read in at
                   step 2 in MAIN.
RMAX_CM2(ICM)      square of maximum radius.
RMAX_CM_FLAG(ICM)  flag for type of outer boundary of CM:
                   0--bounds of CM are all set in HOWFAR_$CMNAME,
                   1--CM is bounded by cylinder of radius RMAX_CM(ICM),
                   2--CM is bounded by square box, walls at RMAX_CM(ICM) and
                   -RMAX_CM(ICM).  Set in INPUT_$CMNAME.
```

```
Z_min_CM(ICM)      minimum Z for each CM, set by previous CM in INPUT_$CMNAME
                   (back of previous CM) usually following convention that
                   downstream surface of source or accelerator exit window
                   is at Z = 0.0. Last value (ICM = MAX_CMs + 1) is
                   maximum Z of model.
Z_gap_THICK(ICM)   thickness of air gap at front of CM to fill in space
                   between Z_min_CM and front of CM, set in INPUT_$CMNAME
Z_min_thick(ICM,j) minimum thickness in cm of up to j = 5 regions in
                   ICM for an electron going through ICM. It is used for
                   range rejection. It is set in each CMs input routine.
                   j=1 is closest to the bottom plane. Often only
                   one = total thickness (air).
MED_min_thick(ICM,j)  medium values corresponding to min thicknesses.
ITDOSE_ON          if dose components scored, this flag is 1, otherwise 0
                   -dose components are from LATCH values or incident charge
ICM_CONTAM         If ITDOSE_ON is 1, & ICM_CONTAM is >= 1
                   then dose is broken into 2 components based on the charge
                   entering the front of ICM ICM_CONTAM.
IQ_CONTAM          charge of the particles considered to be the contaminant
                   on entering ICM = ICM_CONTAM (identified via
                   bit 30 in LATCH).
XTUBE_EXISTS       flag is 0 unless first CM in accelerator is XTUBE, in
                   which case it is 1.
ANGLE              = angle between X-ray target surface and z-axis
CMTYPE(ICM)        8 character ordered array with names of CM types
CMLIST(ICM)        8 character ordered array with identifiers for CMs
AIR_INDEX          index for the ''air'' region =1 unless 0 for vacuum
```

A.6 Useful Utilities

If you are running under Unix/Linux, then the following scripts (found in `$OMEGA_HOME/beamnrc/CMs`) are useful for developing CMs.

checkCM8 checks that once `$CMNAME` is expanded to 8 characters (as it is when changed to identifier names prior to `MORTRAN3` compilation), that all records are less than 80 columns, since that is all that `MORTRAN3` handles.

CMtoc CM "table of contents" script. Produces a combined listing of `$CMNAME_macros.mortran` and `$CMNAME_cm.mortran`, `toc:$CMNAME`, with line numbers, page headings and a table of contents at the top which is based on the strings `"toc:` and `%E` in the source file and the locations of all `REPLACE` macros.

Index