DISCLAIMER

This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency thereof, nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof. Reference herein to any social initiative (including but not limited to Diversity, Equity, and Inclusion (DEI); Community Benefits Plans (CBP); Justice 40; etc.) is made by the Author independent of any current requirement by the United States Government and does not constitute or imply endorsement, recommendation, or support by the United States Government or any agency thereof.

LLNL-TR-2006166

Lawrence Livermore National Laboratory

COG Software Architecture Design Description Document

E Lent

May 2025

Disclaimer

This work performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344.

LAWRENCE
LIVERMORE
NATIONAL
LABORATORY

LLNL-TR-XXXXX

COG Software Architecture Design Description Document

Edward M. Lent

May 1, 2025

Disclaimer

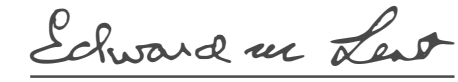
Auspices Statement

This work performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344.

COG

Software Architecture
Design Description Document

Prepared by:



Edward M. Lent
COG Code Developer

Reviewed by:

Chuck K. Lee
COG Code Developer

Approved by:

David P. Heinrichs
Nuclear Criticality Safety Division Leader

Document Revision History

Revision	Date	Description
0	9-12-09	ISQAP requirement per CSAM 08-086, *SQA Gap Analysis for COG*
1	3-19-14	Revised for version COG11.1
2	5-1-25	Revised for version COG11.3

Table of Contents

1. Introduction 1
   1.1. Purpose 1
   1.2. Scope 1
   1.3. Organization 1
2. References 2
3. Software Design 3
   3.1. Requirements 3
   3.2. Objectives 3
   3.3. Code Modules 3
      3.3.1. Principal Code Modules 3
      3.3.2. Other Code Modules (Alphabetical Listing) 5
   3.4. Operational Phases 6
   3.5. Flow Diagrams by Operational Phase 6
      3.5.1. Input Phase Modules 7
         1. Geom Module Input Phase Routines 7
         2. Sor Module Input Phase Routines 8
         3. Det Module Input Phase Routines 10
         4. RW Module Input Phase Routines 11
         5. TXS Module Input Phase Routines 12
         6. EGS Module Input Phase Routines 15
         7. Pro Module Input Phase Routines 16
      3.5.2. Transport Phase Modules 17
         1. RW Module Transport Phase Routines 17
         2. Geom Module Transport Phase Routines 20
         3. Det Module Transport Phase Routines 21
         4. Sor Module Transport Phase Routines 22
         5. TXS Module Transport Phase Routines 24
      3.5.3. Output Phase Modules 25
         1. Geom Module Output Phase Routines 26
         2. Det Module Output Phase Routines 28
         3. RW Module Output Phase Routines 29
         4. TXS Module Output Phase Routines 30
      3.5.4. Utility Routines 30
         1. Mem module routines 30
         2. TXS module routines 31
         3. RW module routines 31
      3.5.5. Externally---Provided Software 31
   3.6. Data Structures 31
      3.6.1. Geometry Data Structures 31
         1. Surface Data Storage 31
         2. Volume Data Storage 40
      3.6.2. Event History Storage (EHS) Data Structures 41
      3.6.3. Input File 43

3.6.4. Common Block Usage 43
INDEX 46

List of Tables

Table 1 Internal Surface Types 32
Table 2 Surface Definition Data Storage 35
Table 3 Sector Description (sd) Array 41
Table 4 Event History Store Record 42
Table 5 Events 43

1. INTRODUCTION

1.1. Purpose

This *COG Software Architecture Design Description Document* describes the organization and functionality of the COG Multiparticle Monte Carlo Transport Code for radiation shielding and criticality calculations, at a level of detail suitable for guiding a new code developer in the maintenance and enhancement of COG. The intended audience also includes managers and scientists and engineers who wish to have a general knowledge of how the code works. This Document is not intended for end-users.

1.2. Scope

This document covers the software implemented in the standard COG Version 10, as released through RSICC and IAEA. Software resources provided by other institutions will not be covered.

1.3. Organization

This document presents the routines grouped by modules and in the order of the three processing phases. Some routines are used in multiple phases. The routine description is presented once – the first time the routine is referenced. Since this is presented at the level of detail for guiding a new code developer, only the routines invoked by another routine that are significant for the processing phase that is being detailed are presented. An index to all routines detailed is included.

Tables for the primary data structures are also presented.

2. REFERENCES88

- CSAM 08-086, *SQA Gap Analysis for COG, Nuclear Criticality Safety Division*, Lawrence Livermore National Laboratory, September 3, 2008.

- IEEE Std 1016-1998, *IEEE Recommended Practice for Software Design Descriptions*, Institute of Electrical and Electronics Engineers, Inc., September 23, 1998.

- SBK08-085, *Memorandum of Understanding between the Nuclear Materials Technology Program (NMTP) and Nuclear Operations (NucOps)*, Lawrence Livermore National Laboratory, April 30, 2008.

- UCRL-TM-202590, *COG, A Multiparticle Monte Carlo Transport Code, User's Manual*, Fifth Edition, Lawrence Livermore National Laboratory, September 1, 2002.

- http://cog.llnl.gov

3. SOFTWARE DESIGN

3.1. Requirements

The basic requirement is to develop a code package that will implement best-of- class Monte Carlo particle transport methods to solve radiation shielding and nuclear criticality problems involving complex 3D geometry, multiparticle sources, arbitrary materials, and many detector types.

3.2. Objectives

Given that the requirements necessitate a large and complex software package, functional decomposition was used. As far as possible, major code operations were decomposed into code modules that perform single functions. For example, the handling of geometry, sources, materials, and detectors is performed in appropriately named modules, each of which is a code source directory.

3.3. Code Modules

A **Code Module** is conceptually a collection of Source routines that implement a major code function or feature, such as particle detector simulation. A **Module** is instantiated as a file system directory containing Source files, such as the **Det** directory. For every Source **Module**, there is a corresponding object library file, such as **libDet.a**. The COG code is generated by compiling all Source **Modules** into object libraries, then linking the library files.

3.3.1. Principal Code Modules

Main (Main Module)

Purpose	**Functions**
Controls the execution of the user's job.	Calls routines that read the input file, set up the computation, track the source particles through the problem's geometry, and output results.

Geom (Geometry Module)

Purpose	**Functions**
Reads and stores input file data that specify the problem's geometry and material composition.	Reads entire input file. Checks for input syntax errors. Processes and stores input-file Basic Block data. Processes and stores input-file Geometry Block data. Processes and stores input-file Mix Block data. Optionally, helps detect geometry setup errors by: making sweeps through the geometry; making 2D and perspective pictures of the user's geometry; calculating object volumes.

Sor (Source Module)

Purpose	Functions
Reads and stores input file data that specify the problem's particle sources.	Processes and stores input-file Source Block data. Produces pictures of source particle emission to aid in verifying source descriptions. During Transport phase, produces source particles for tracking through geometry.

Det (Detector Module)

Purpose	Functions
Reads and stores input file data that specify the problem's particle detectors.	Processes and stores input-file Detector Block data. Produces pictures of source particle emission to aid in verifying source descriptions. Reads/writes restart dumps. During Transport phase, scores particles that reach detectors. During Output phase, prints and plots detector results.

RW (Random Walk Module)

Purpose	Functions
Performs tracking of source particles through the problem's geometry and materials.	Processes and stores input-file Walk-XX Block data. Processes and stores input-file Activation Block data. During Transport phase, tracks one particle at a time through the problem's geometry, using optional Walk-XX Block parameters to improve performance. During Output phase, makes analysis plots.

TXS (Total Cross Section Module)

Purpose	Functions
Handles the physics data that determines collision probabilities and outcomes for particle transport.	Reads physics databases and creates particle collision data for the materials in the problem. During Transport phase, provides cross-section data and collision outcomes to the RW module.

EGS (Electron Gamma---Ray Shower Module)

Purpose	Functions
Transports electrons through the problem's geometry.	Processes and stores input-file EGS Block data. During Transport phase, tracks one electron at a time through the problem's geometry,

Pro (Proton Transport Module)

Purpose	Functions
Transports protons through the problem's geometry.	Processes and stores input-file Pro Block data. During Transport phase, tracks one proton at a time through the problem's geometry,

CPTrans (Charged Particle Transport Module)

Purpose	Functions
Transports charged particles through the problem's geometry.	Processes and stores input-file CPTrans Block data. During Transport phase, tracks one particle at a time through the problem's geometry,

3.3.2. Other Code Modules (Alphabetical Listing)

Grf (Graphics Module)

Purpose	Functions
Supports COG graphics.	Enables plotting to screen or file in choice of colors, lines, points, mappings. Communicates with lower-level PGPLOT library routines.

FREYA (FREYA Module)

Purpose	Functions
Supports FREYA option.	Allows for the 'Fission Reaction Event Yield Algorithm' to process fission events.

include (Include Module)

Purpose	Functions
Contains all "include" files needed by COG.	Collects all include files in one directory.

Lattice (Lattice Module)

Purpose	Functions
Supports lattice operations in geometry.	Reads lattice descriptions. Transports particles through geometry.

LF (Lagged Fibonacci Random Number Generator Module)

Purpose	Functions
Provides pseudo-random numbers for the Monte Carlo aspects of COG.	Processes and stores the starting random number seeds for the job. During Transport phase, provides a series of pseudo-random numbers to the Sor, RW, and TXS modules.

Mag (Magnetic Field Module)

Purpose	Functions
Calculates charged particle motion in a vacuum magnetic field.	Processes and stores input-file Mag Block data. During Transport phase, steps particle through specified magnetic fields.

Mem (Memory Management Module)

Purpose	Functions
Improves memory diagnostics	Wraps Unix memory management routines to provide better diagnosis of memory allocation errors.

MPI (Message Passing Interface Module)

Purpose	Functions
Permits COG parallel processing	Processes and stores input-file Parallel Block Data.
	Sets up a parallel Master/Slave run under the MPI system. During Transport phase, passes messages between Master and Slave processes to enable the simultaneous execution of multiple copies of COG on one input file.

PP (Parallel Processing Support Module)

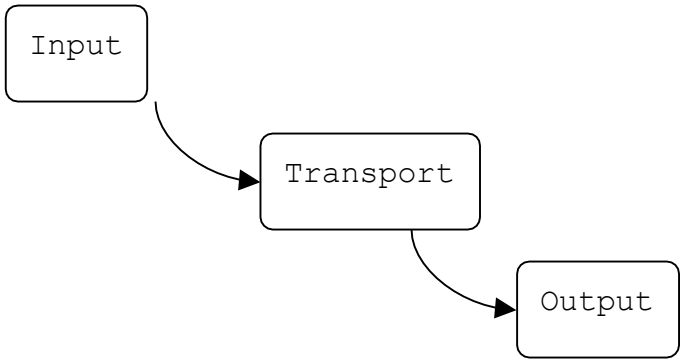
Purpose	Functions
Supports COG parallel processing	Sums up results over all Slave processes.

PEGS (PEGS Module)

Purpose	Functions
Supports electron transport processing.	Allows for the optional calling of PEGS to create the required EGS electron data file.

3.4. Operational Phases

An **Operational Phase** is a logical division of the workflow that COG performs during the running of a user's job. These divisions are chosen to represent distinct major tasks. The phase breakdown we have chosen is: Input, Transport, and Output.



Input Phase - Read and process input file and library files.
- Perform error checks: geometry pictures. Volume calculations.

Transport Phase - Calculate particle trajectories and scoring.

Output Phase - Create output listing and make pictures of results.

3.5. Flow Diagrams by Operational Phase

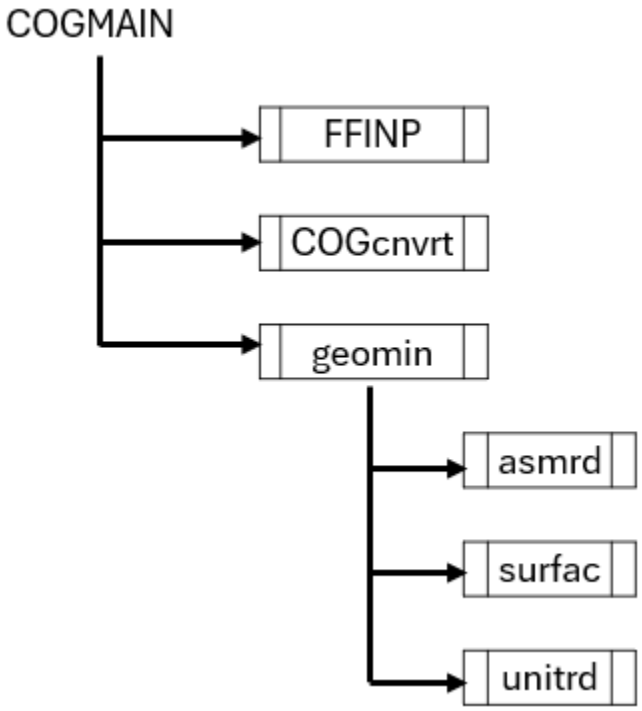
The following steps through the operational phases and describes the routines within each primary module that are used for that phase. Routines from a different module are indicated with a prefix such as RW:.

3.5.1. Input Phase Modules

Routines from the following Modules are used in the Input Phase in the following order:

- Geom
- Sor
- Det
- RW
- TXS
- EGS
- Pro

1. Geom Module Input Phase Routines



ffinp
Reads the COG input file and parses it into symbols.
Parsed symbols are stored as character strings and as numbers.
Symbols are scanned to locate Data Block names and identify the beginning and end of each Block.

COGcnvrt
Converts simple lattice instructions in the GEOMETRY Block into COG input.

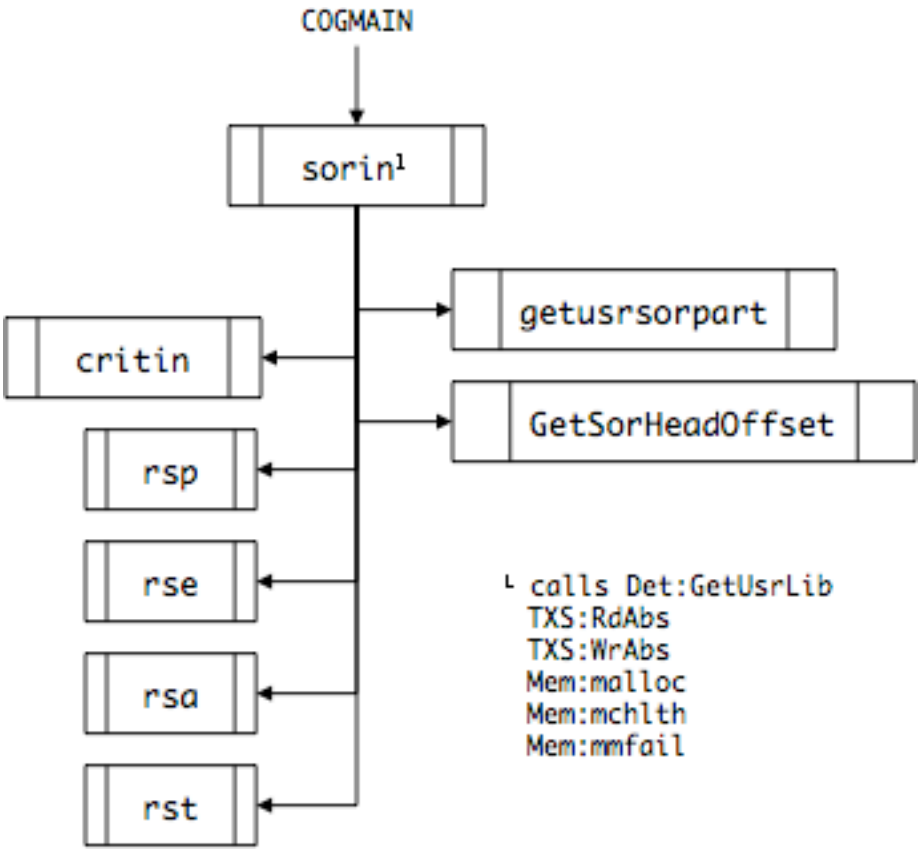
geomin
Processes these parsed Data Blocks:
ASSIGN SURFACES GEOMETRY

asmrd
Processes ASSIGN Data Blocks.
Assigns materials, densities, scoring region numbers to sectors.

surfac
Processes SURFACES Data Block.
Computes a mathematical model for each surface and stores coefficients.

unitrd
Processes GEOMETRY Data Block.
Generates and stores sector descriptions based on bounding surfaces.

2. Sor Module Input Phase Routines



sorin
Processes the parsed SOURCE Data Block. Creates the particle source from the Source description to be used during the Transport Phase.

critin
Processes the source specification for a criticality job.

----Routines that process the Source dependence on location, energy, angle, and time----

rsp
Processes the Source Position specification. Plots 1500 samples chosen from this specification.

rse
Processes the Source Energy specification. Plots samples chosen from this specification.

rsa
Processes the Source Angle specification. Plots samples chosen from this specification.

rst
Processes the Source Time specification. Plots samples chosen from this specification.

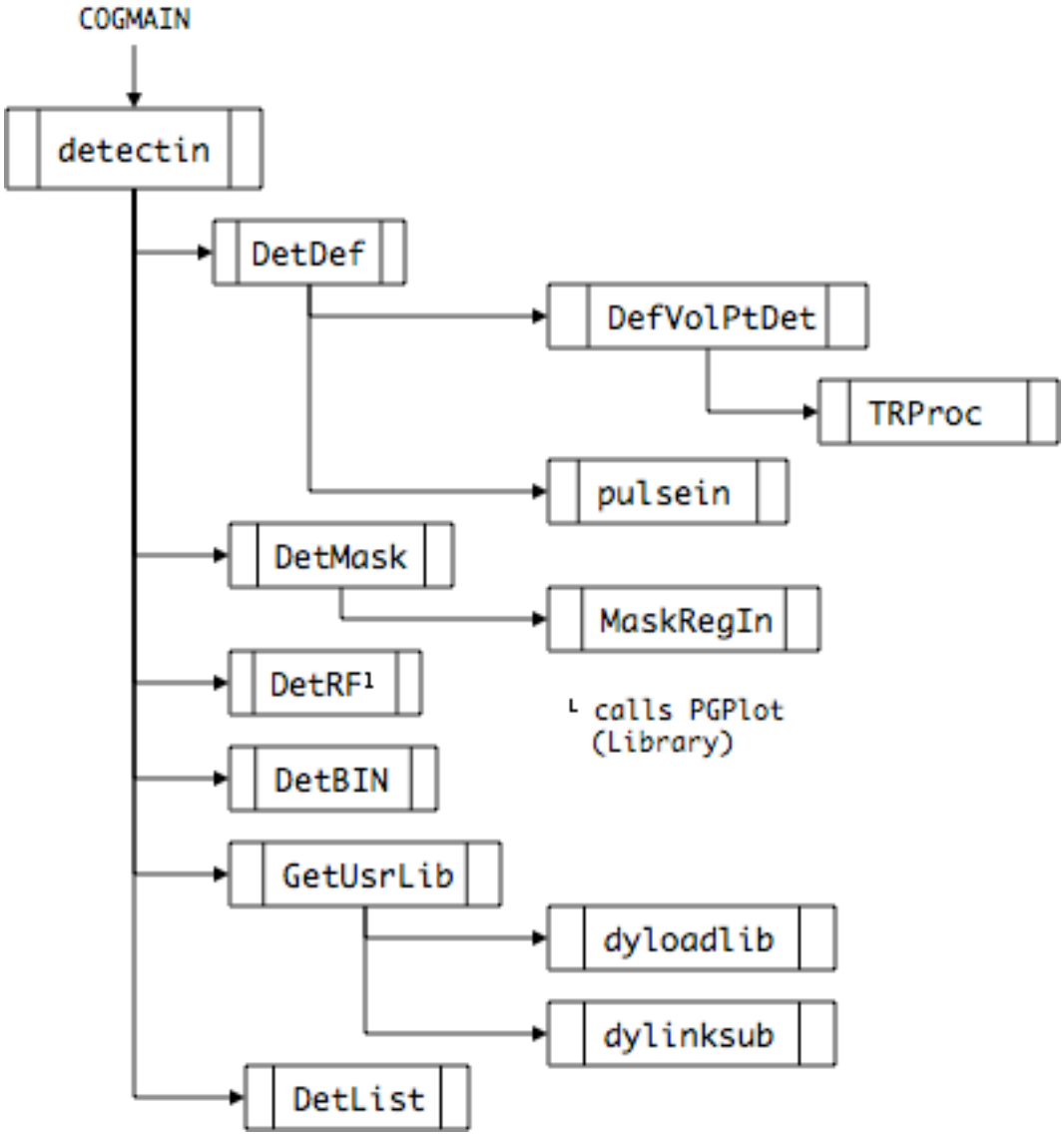
----Routines that process a User-Defined Source Routine----

getusrsorpart
Initializes the User-Defined Source routine.

----Routines that read/write a Source specification from/to a COG .sor file----

GetSorHeadOffset
Skips over the header record and positions the file for reading/writing source data.

3. Det Module Input Phase Routines



detectin
Processes the parsed DETECTOR Data Block that specifies the detectors in the problem.

DetDef
Processes the input parameters for most detector types (Boundary-Crossing, Reaction, Point)

DefVolPtDet
Processes the parameters for the Volume Point Detector type.

TRProc
Handles coordinate transformations needed for the Volume Point Detector.

pulsein
Processes the parameters for the Pulse Detector type.

DetMask
Sets up scoring masks that limit (in energy, time, angle,..) what the detector "sees".

MaskRegIn
Sets up masks that limit the Regions that the detector "sees".

DetRF
Processes the detector response functions.

DetBIN
Processes differential scoring bin structures.

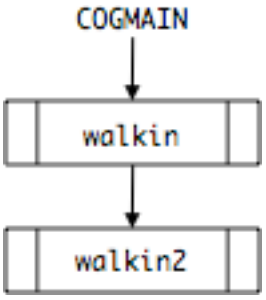
GetUsrLib
For a user-supplied detector library, loads and verifies presence of user routines.

dyloadlib
Loads user library at runtime.

dylinksub
Dynamically links COG to the user-written routines.

DetList
Processes the List option.

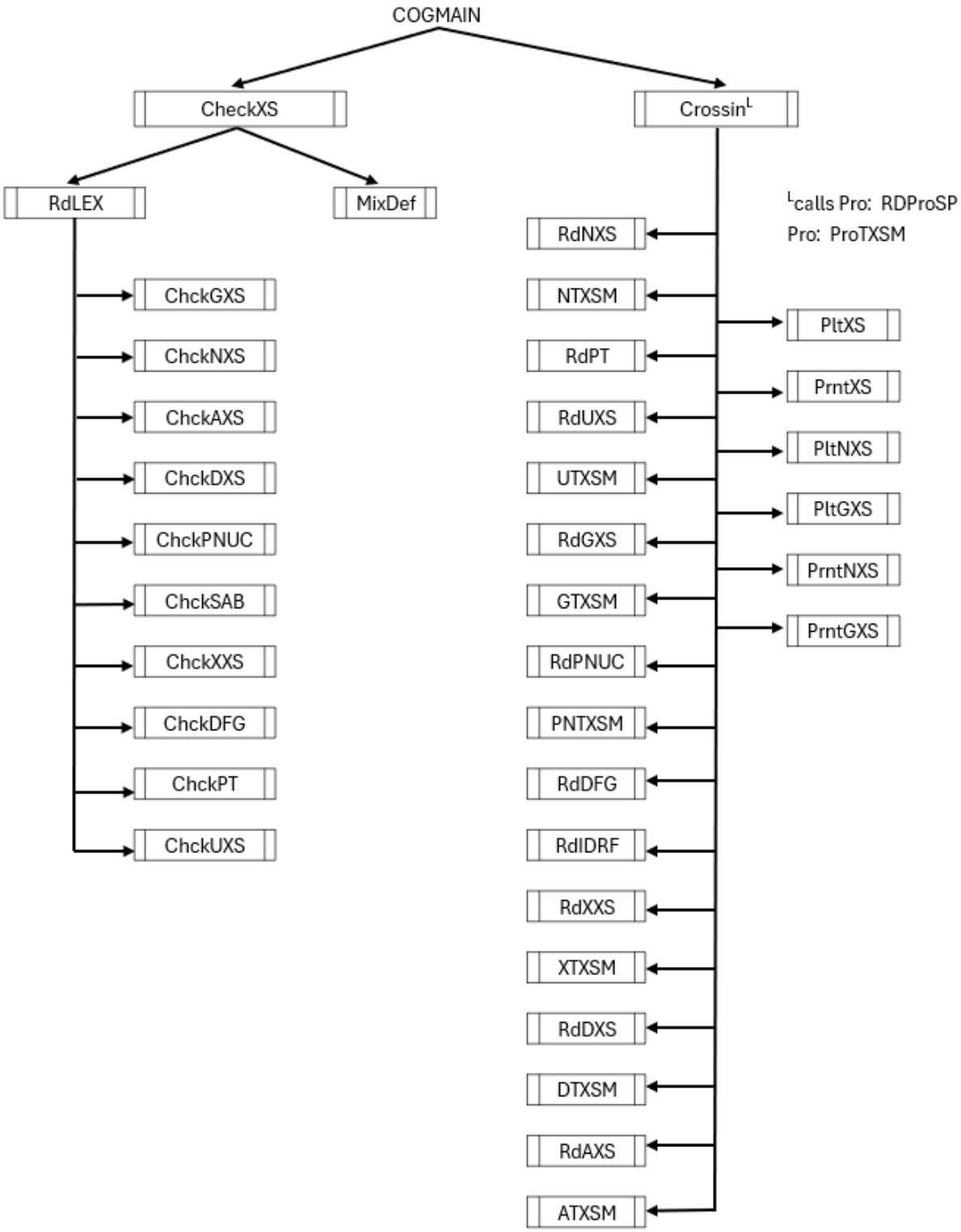
4. RW Module Input Phase Routines



walkin
Processes the parsed WALK_XXX Data Blocks that specify parameters that modify the standard random walk methods.

walkin2
Identifies and stores the random walk modification parameters.

5. TXS Module Input Phase Routines



CheckXS
Checks validity of particle type, then reads the material dictionary and prints mixture definitions.

Crossin
Manages the reading of COG cross section files for a specified particle type.

Reads COG dictionary and translates names of the materials and isotopes specified in the MIX Block into ZA names used in the data libraries. Calls various checking routines (**ChckGXS**, **ChckNXS**, **ChckAXS**, **ChckDXS**, **ChckPNUC**, **ChckSAB**, **ChckXXS**, **ChckDFG**, **ChckPT**, and/or **ChckUXS**) to ensure that the requested data (photon, neutron, photonuclear, thermal, activation, delayed fission gamma, probability table, and/or nuclear resonance fluorescence) are available in the specified COG data libraries.

MixDef
Prints mixture definitions in terms of their component isotopes ZA names.

RdDFG
Reads delayed-fission gamma data from the selected COG delayed gamma data library if BASIC Block 'neutron', 'photon', and 'delayedgamma' options are specified.

RdNXS
Reads neutron cross section data from the specified COG neutron data library if BASIC Block 'neutron' option is specified.

NTXSM
Forms total neutron cross section for each material in job if BASIC Block 'neutron' option is specified.

RdAXS
Reads neutron cross section data from the specified COG neutron data library if BASIC Block 'alpha' option is specified.

ATXSM
Forms total neutron cross section for each material in job if BASIC Block 'alpha' option is specified.

RdDXS
Reads neutron cross section data from the specified COG neutron data library if BASIC Block 'deuteron' option is specified.

DTXSM
Forms total neutron cross section for each material in job if BASIC Block 'deiteron' option is specified.

RdXXS
Reads neutron activation data from the selected COG activation data library if BASIC Block ‘neutron’ and ‘activation’ options are specified.

XTXSM
Forms total neutron activation cross section for each material in job if BASIC Block ‘neutron’ and ‘activation’ options are specified.

RdGXS
Reads photon data from the selected COG photon data library if BASIC Block ‘photon’ option is specified.

GTXSM
Forms total photon cross section for each material in job if BASIC Block ‘photon’ option is specified.

RdPNUC
Reads photonuclear data from the selected COG photonuclear data library if BASIC Block ‘neutron’, ‘photon’, and ‘photonuclear’ options are specified.

RdPT
Reads unresolved resonance region probability table data from the selected COG probability table data library if BASIC Block ‘neutron’ and ‘urrpt’ options are specified.

RdUXS
Reads nuclear resonance fluorescence data from the selected COG nuclear resonance fluorescence data library if BASIC Block ‘photon’ and ‘nrf’ options are specified.

UTXSM
Forms total nuclear resonance fluorescence cross section for each material in job if BASIC Block ‘photon’ and ‘nrf’ options are specified.

RdIRDF
Reads dosimetry data from the selected COG dosimetry data library if DETECTOR Block ‘drf-e irdf-r-r’ option is specified.

PNTXSM
Forms total photonuclear cross section for each material in job if BASIC Block ‘neutron’, ‘photon’, and ‘photonuclear’ options are specified.

PltXS
Plots total neutron and/or photon cross section for each material in job if I/O Block ‘plotmxs’ and BASIC Block ‘neutron’ and/or ‘photon’ options are specified.

PrntXS
Prints total neutron and/or photon cross section for each material in job if I/O Block ‘printmxs’ and BASIC Block ‘neutron’ and/or ‘photon’ options are specified.

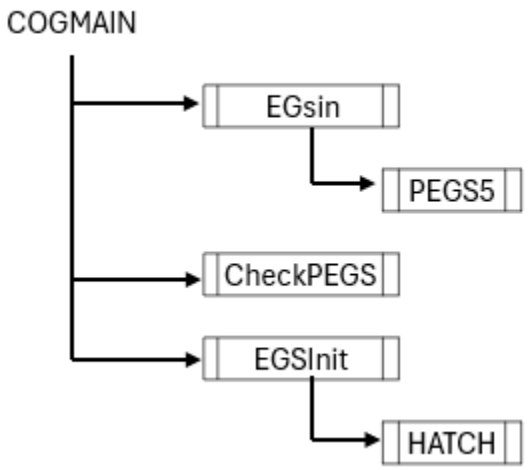
PltNXS
Plots total neutron cross section for each isotope in job if I/O Block ‘plotxs’ and BASIC Block ‘neutron’ options are specified.

PltGXS
Plots total photon cross section for each element in job if I/O Block ‘plotxs’ and BASIC Block ‘photon’ options are specified.

PrntNXS
Prints total neutron cross section for each isotope in job if I/O Block ‘printxs’ and BASIC Block ‘neutron’ options are specified.

PrntGXS
Prints total photon cross section for each element in job if I/O Block ‘printxs’ and BASIC Block ‘photon’ options are specified.

6. EGS Module Input Phase Routines



EGSin
Processes the parsed EGS Data Block that specifies parameters for electron trtansport.

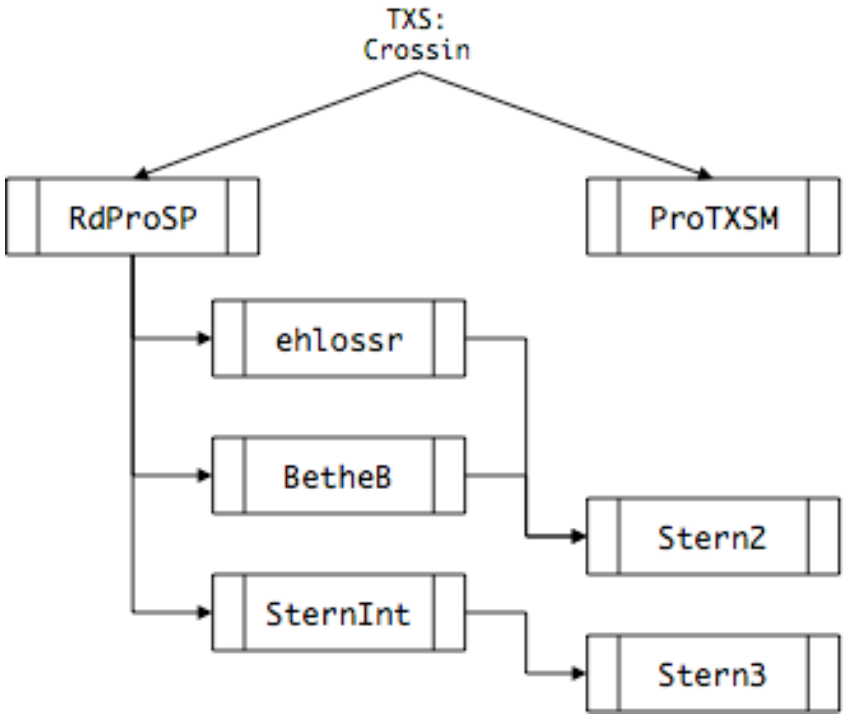
PEGS5
Creates a PEGS data file from a user input file.

CheckPEGS
Checks that the PEGS data file has materials that match the COG materials list.

EGSInit
Initializes the EGS module and reads in the PEGS data file.

HATCH
EGS routine creates physics tables for electron/gamma-ray transport.

7. Pro Module Input Phase Routines



RdProSP
Reads proton data if BASIC Block 'proton' option is specified. Generates a table of proton stopping powers to be used in proton transport.

ProTXSM
Forms total proton cross section for each material in job if BASIC Block 'proton' option is specified.

ehlossr
Computes the relativistically-correct stopping power for heavy charged particles traversing a one-isotope medium.

BetheB
Computes the Bethe-Bloch Stopping Power Formula, without the Ce/Z term, for an isotope or mixture/compound.

SternInt
Computes the Sternheimer relativistic density effect correction to the heavy ion stopping power, for non-tabular compounds, and for elements and compounds at non-standard densities.

Stern2
Computes the Sternheimer relativistic density-effect correction to the heavy ion stopping power, for an element or a mixed material.

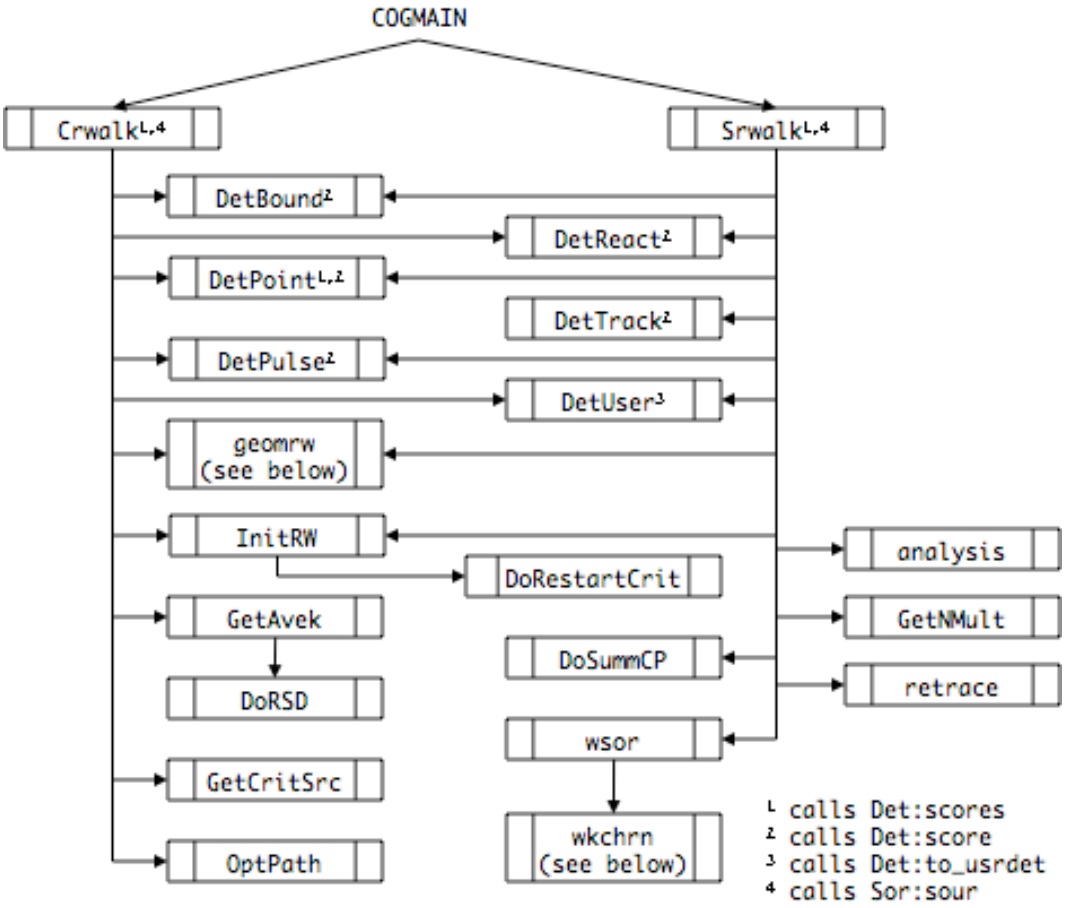
Stern3
Computes the Sternheimer relativistic density-effect correction to the heavy ion stopping power, for an array of particle energies. This version adjusts the tabular coefficients for changes in density.

3.5.2. Transport Phase Modules

Routines from the following Modules are used in the Transport Phase,

- RW
- Geom
- Det
- Sor
- TXS

1. RW Module Transport Phase Routines



Crwalk
Manages the criticality calculation. Runs batches to problem completion.

Srwalk
Manages the shielding transport calculation.

DetBound
Processes EHS and scores particles for the problem's Boundary-Crossing detectors. EHS is the Event History Store, a data array that accumulates all the events occurring during the tracking of a single source particle, and all of its secondary particles, for subsequent analysis.

DetReact
Processes EHS and scores particles for the problem's Reaction detectors.

DetPoint
Processes EHS and scores particles for the problem's Point detectors.

DetTrack
Processes EHS and scores particles for the problem's Tracklength Estimation detectors.

DetPulse
Processes EHS and scores particles for the problem's Pulse detectors.

DetUser
Proesses EHS and scores particles for the problem's User-Defined detectors.

InitRW
Initializes Random-Walk parameters

DoRestartCrit
Restarts a criticality job from a Restart Dump.

GetAvek
Computes criticality factor k at end of each batch.

DoRSD
Makes a Restart Dump during a criticality run

GetCritSrc
Produces the criticality source for the next batch.

OptPath
Computes optical path by region.

analysis
Stores particle data for subsequent analysis plots

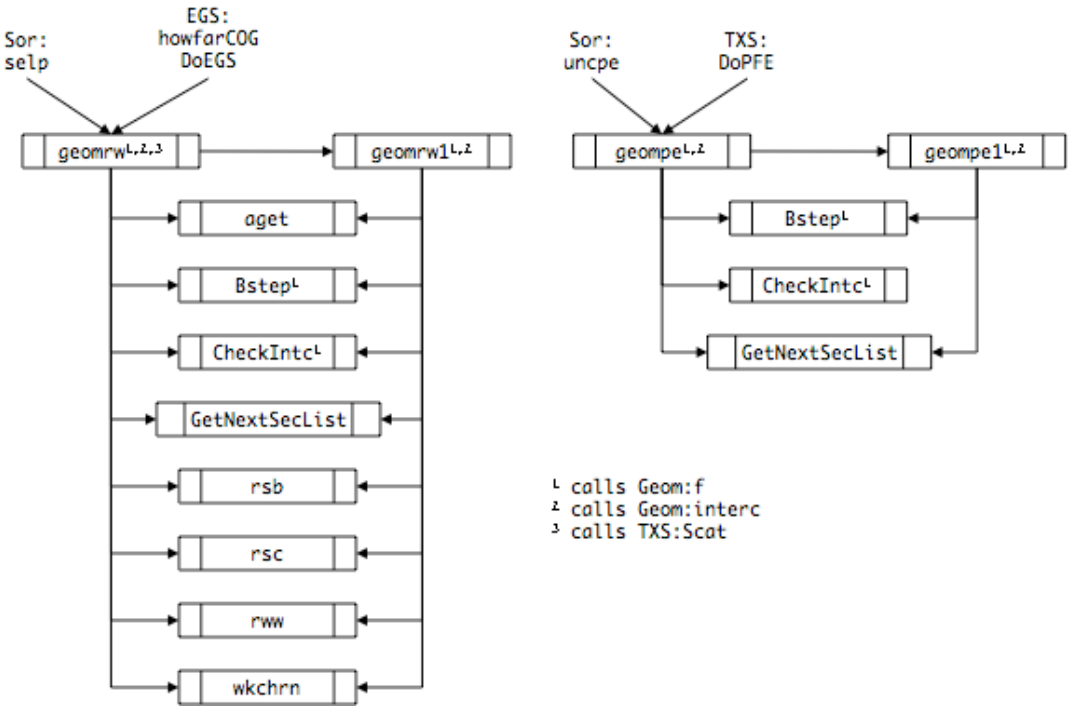
GetNMult
Calculates neutron multiplication.

DoSummCP
Creates charged particle summary records in the EHS, for subsequent detector scoring.

retrace
Prints an event history summary for a specified particle.

wsor
Selects particles from a collision-based source.

----Additional RW Module Routines called from other Modules----



geomrw
Manages particle transport through main-level geometry.

geomrw1
As above, but for lower-level geometry units.

aget
Returns particle cutoff age (if any).

BStep
Steps particle across a geometry bounding surface.

CheckIntc
Checks validity of sector-exit surface intercept calculation.

GetNextSecList
Produces ordered sector list to help locate next sector entered by particle.

rsb
Performs Russian Roulette or splitting at a geometry boundary.

rsc
Performs Russian Roulette or splitting at a collision site.

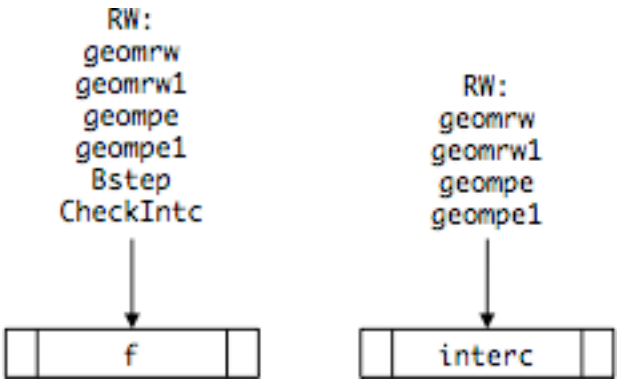
rww
Performs Russian Roulette or splitting at a collision site, using weight windows.

wkchrn
Performs the Walk-Channel calculation.

geompe
Performs virtual particle transport though the main-level geometry for Point detectors.

geompe1
As above, but for lower-level geometry units.

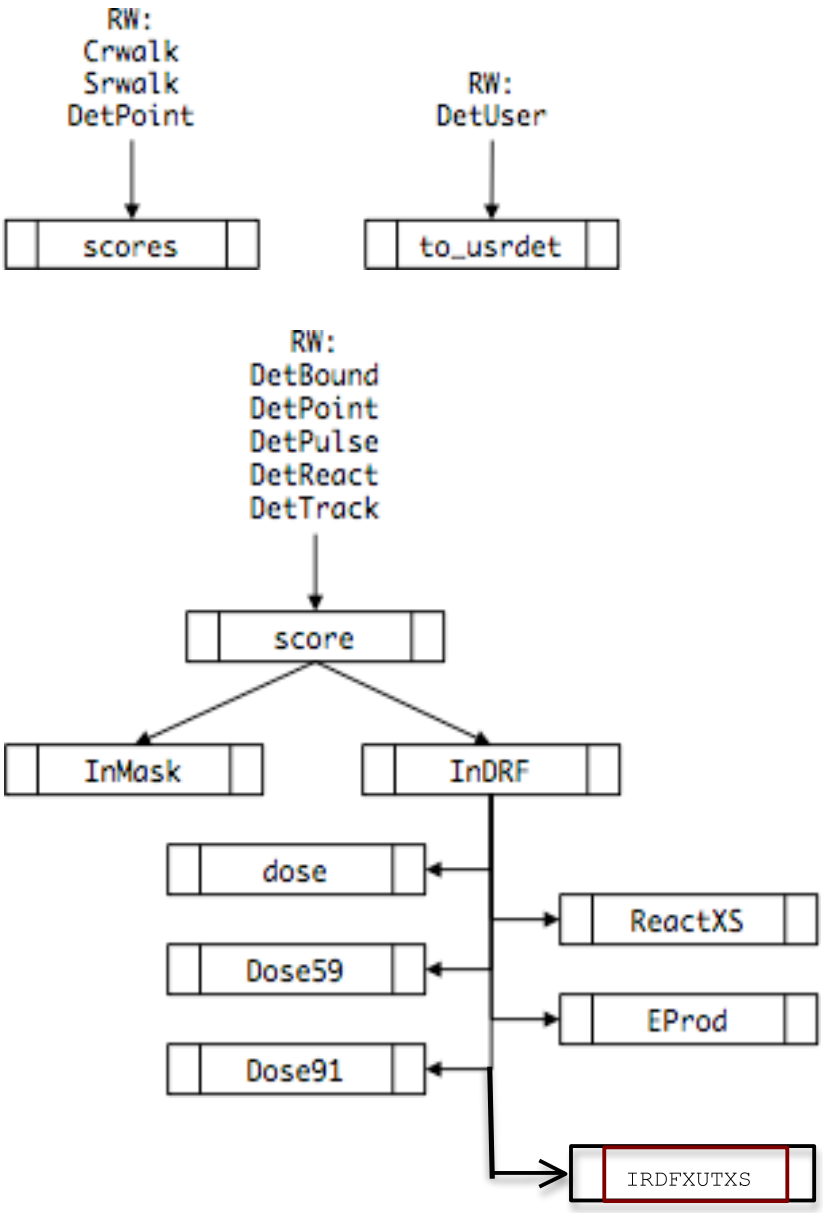
2. Geom Module Transport Phase Routines



f
Evaluates a surface equation at the location of the particle.

interc
Finds the intercept between a particle's trajectory and a boundary surface.

3. Det Module Transport Phase Routines



score
Reviews a particle history in the EHS and calculates its scoring contribution to the problem's detectors.

scores
Sums the single-particle score into the total scoring arrays.

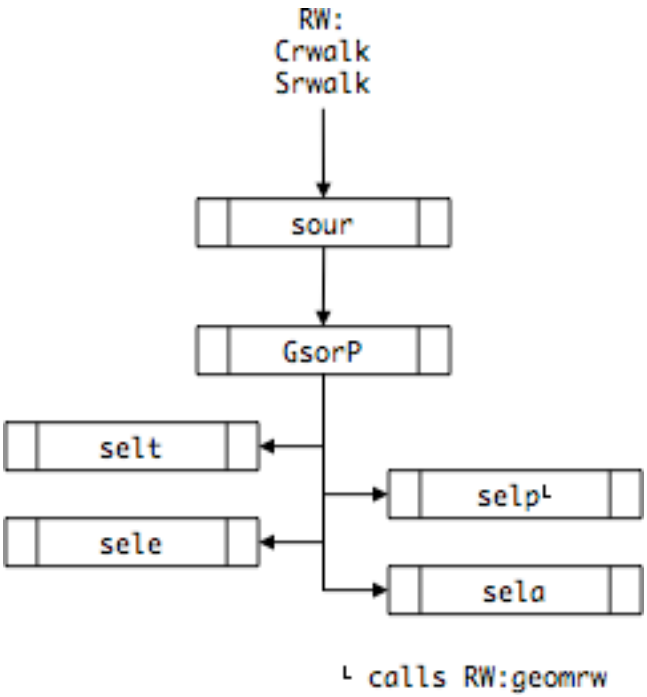
InDRF
Converts detector scores from flux units into desired detector output quantities.

InMask
Checks if scoring particles pass through detector masks.

dose, Dose59, Dose91, ReactXS, Eprod, IRDFXS
Detector response function routines that convert basic flux scores into doses (various standards), reaction rates, electron production, or dosimetry.

to_usrdet
Calls user-defined detector routines to process EHS tables and do custom scoring.

4. Sor Module Transport Phase Routines



sour
If criticality problem…
  Gets source particles from fission bank.
Elseif correlated source…
  Reads random number parameters from previously generated .sor file such that each current source particle is initialized exactly as in previous run.

Elseif retrace option…
  Uses BASIC Block 'RN' option to set the desired initial random number seeds and the SOURCE Block 'RETRACE' option to set the desired random number sequence #'s so as to regenerate (and edit) specific source particles.
Else…
  If 'WRITESOURCE' option writes .sor file.
  Calls… **GsorP**

GsorP
Generates source particles based on current source parameters

If 'CensusSorFlag' set…
  Get source particles from previously generated census file.
Elseif 'UsrSorFlag' set…
  Get source particles from user supplied source routine.
Else…
Calls… **selt**, **sele**, **selp**, and **sela**

selt
Get source particle age from input source parameters.

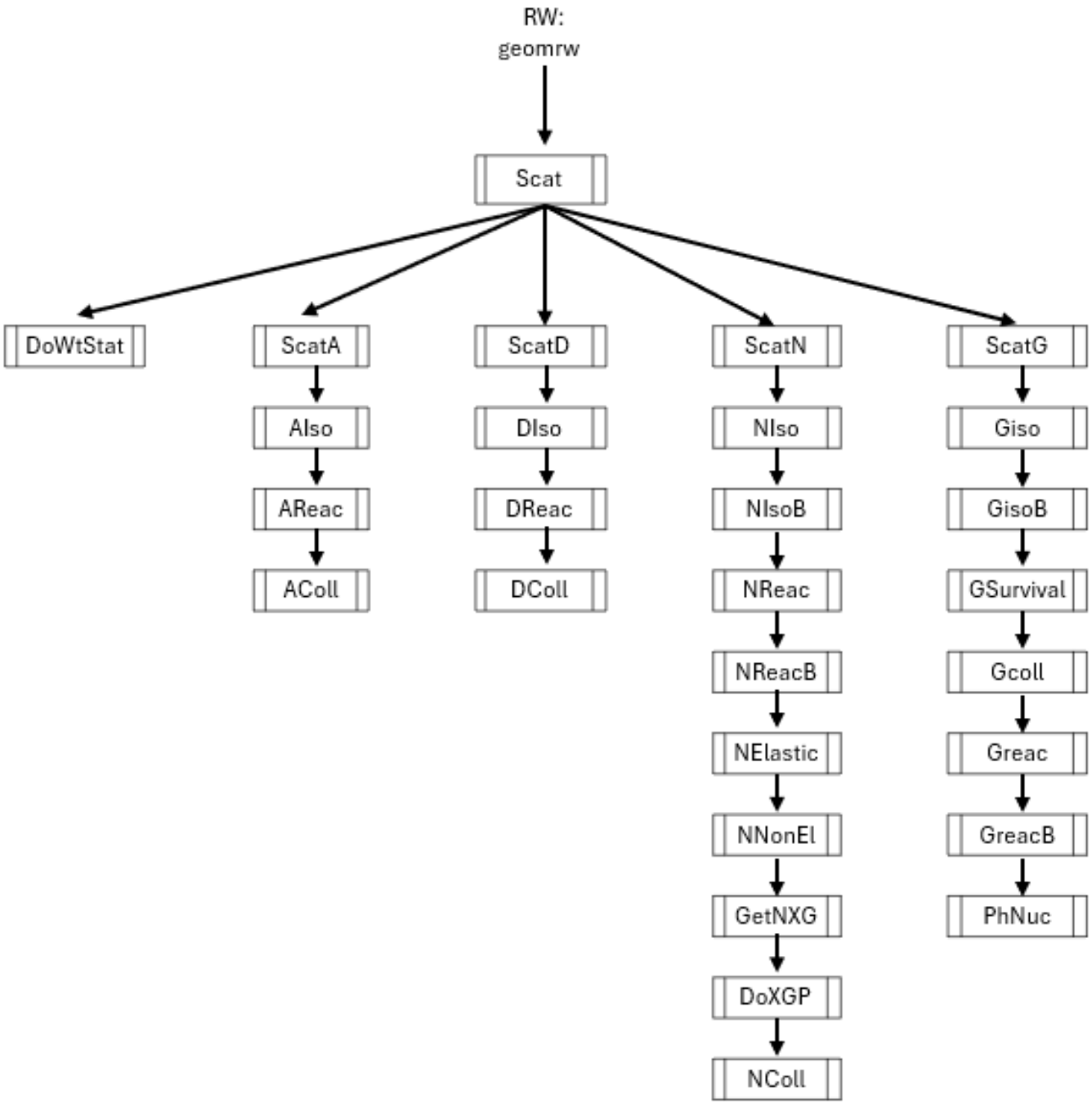
sele
Get source particle energy from input source parameters.

selp
Get source particle position from input source parameters.

sela
Get source particle direction from input source parameters.

uncpe (*calls RW:geompe*)
Computes the uncollided (source) contribution to a point flux estimate detector.

5. TXS Module Transport Phase Routines



Scat
Sets random walk flags (survival, secondary production) and random walk parameters (scattered direction bias, isotope/element bias, reaction bias), then calls…

DoWtStat
Determines min, max, and average weight by region and particle type.

ScatN
Neutron scattering routine.

ScatG
Photon scattering routine.

Niso (or **NisoB**, if isotope biasing set)
Determines in which isotope the neutron scattering event occurs.
AIso
Determines in which isotope the alpha scattering event occurs.

DIso
Determines in which isotope the deuteron scattering event occurs.

NElastic
Determines elastic reaction.

NNonEl
Determines non-elastic reactions.

NColl
Based on isotope and reaction, determines particle parameters (energy, position, direction, weight, age,…) of the scattered particle and stores the particle parameters for all secondary particles on the secondary bank for a neutron scattering event.

AColl
Based on isotope and reaction, determines particle parameters (energy, position, direction, weight, age,…) of the scattered particle and stores the particle parameters for all secondary particles on the secondary bank for an alpha scattering event.

DColl
Based on isotope and reaction, determines particle parameters (energy, position, direction, weight, age,…) of the scattered particle and stores the particle parameters for all secondary particles on the secondary bank for a deuteron scattering event.

NReac (or **NReacB**, if reaction biasing set)
Determines neutron reaction.

GetNXG
Stores particle parameters for (n,Xg) photons on secondary bank.

DoAGP
Stores particle parameters for activation photons on secondary bank.

Giso (or **GisoB**, if isotope biasing set)
Determines in which element the gamma scattering event occurs.

GSurvival
Determines elastic and non-elastic reactions.

GColl
Based on element and reaction, determines particle parameters (energy, position, direction, weight, age,…) of the scattered particle and stores the particle parameters for all secondary particles on the secondary bank for a gamma scattering event.

Greac (or **GreacB**, if reaction biasing set)
Determines reaction.

PhNuc
Stores particle parameters for photo-neutrons on secondary bank.

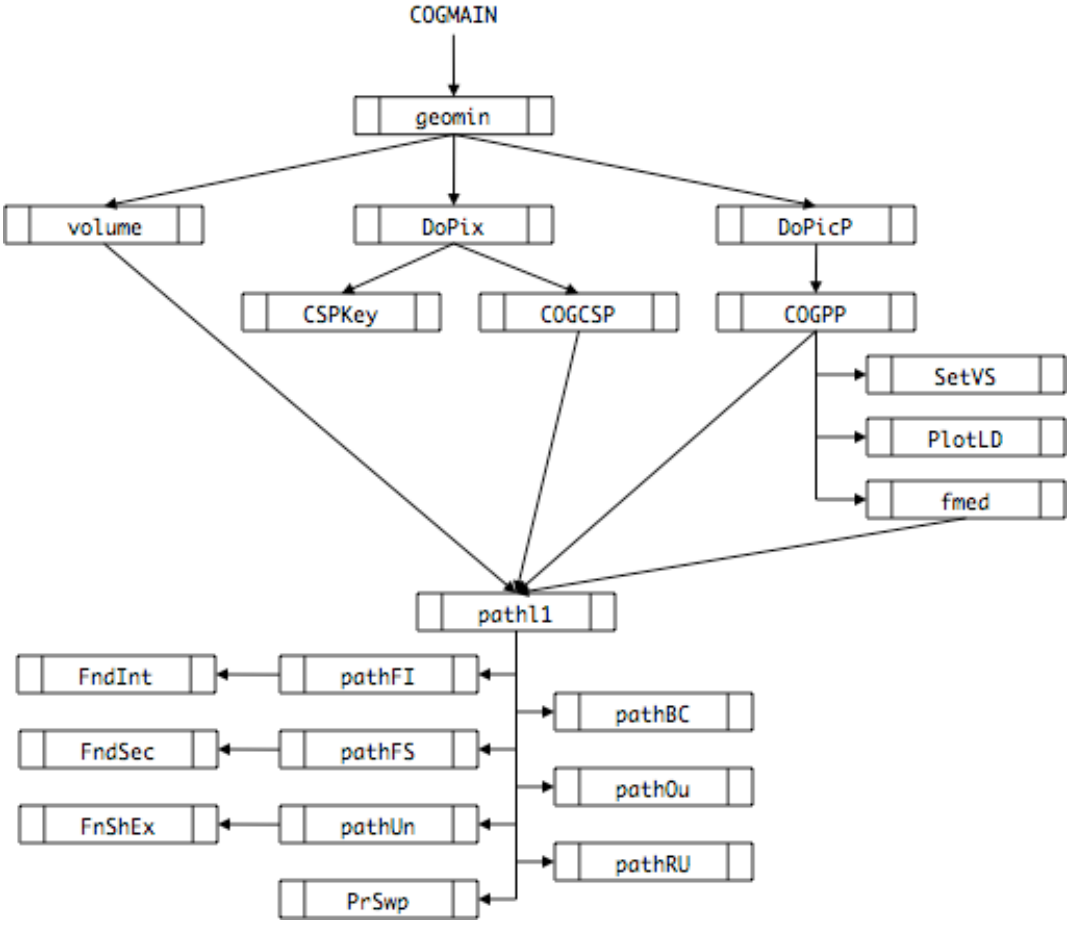
DoPFE (*calls RW:geompe*)
Driver for the computation of the collided contribution to a point flux estimate detector.

3.5.3. Output Phase Modules

Routines from the following Modules are used in the Output Phase,

- Geom
- Det
- RW
- TXS

1. Geom Module Output Phase Routines



geomin (see *Geom Module Input Phase Routines*)

COGCSP
Makes cross-section (2D) views of the geometry.

COGPP
Makes perspective views (3D) of the geometry.

CSPKey
Plots page of keys identifying patterns seen in cross-section plots.

DoPicP
Driver for making perspective views.

DoPix
Driver for making cross-section views.

fmed
For perspective views, finds the sector number of the first visible surface hit by the ray extending from the image-plane pixel at (px,py) through the pinhole and into the user's geometry.

FndInt
Finds intercepts of a sweep line with all visible surfaces of given unit.

FndSec
Finds sector(s) containing specified point P(x,y,z).

FnShEx
Finds the nearest exit surface from the unit shell sector.

pathl1
Driver for sweeping a line though the geometry and computing distances to sector boundaries. Used in sweeps, views, and volume calculations.

pathBC
Moves end of sweep line up to and across a boundary surface.

pathFI
Finds all intercepts of a sweep line with surfaces in specified unit.

pathFS
Finds sector containing a specified point P(x,y,z).

pathOu
Computes the output values for a pathl1 sweep.

pathRU
Terminates sweep through a lower-level unit and returns to higher level.

pathUn
Finds and processes intercepts in lower-level unit.

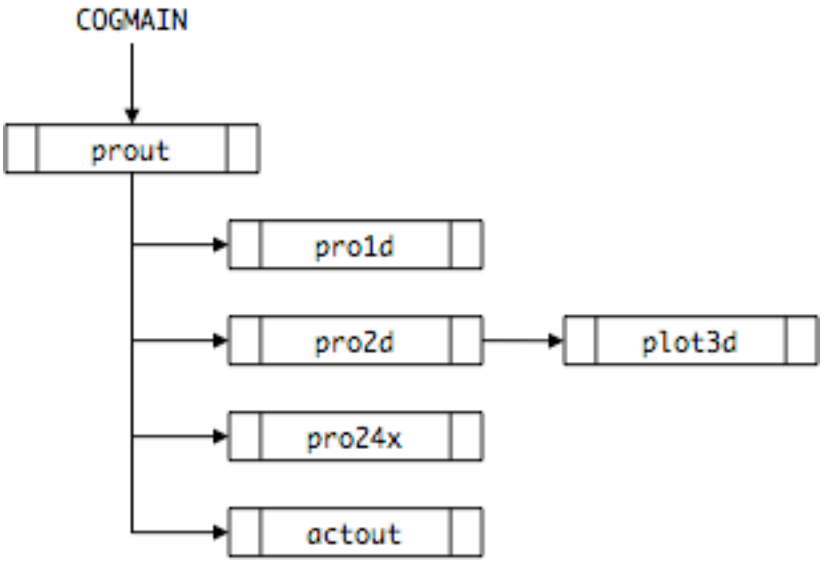
PlotLD
For perspective views, plots the edges found in the last sweep.

PrSwp
Prints results of a user-specified sweep.

SetVS
For perspective views, sets visibility of surfaces.

volume
Calculates volume (or mass) of specified sectors and prints results.

2. Det Module Output Phase Routines



actout
Prints calculated values in activation problems.

plot3d
Plots 2D bin results.

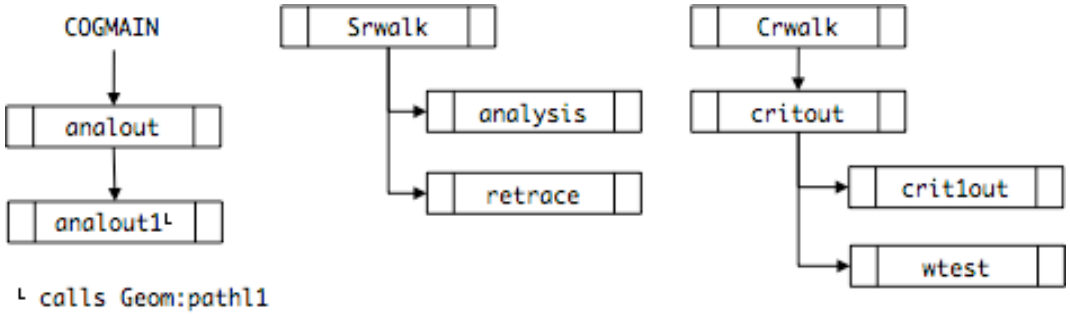
pro1d
Prints and plots detector results and statistics for 1D bins: time, energy, angle.

pro2d
Prints detector results and statistics for 2D bins: energy and time, time and angle, energy and angle.

pro24x
Prints detector results and statistics for all other 2D bins.

prout
Calculates, prints, and plots detector results. Writes .det file.

3. RW Module Output Phase Routines



Srwalk (see *RW Module Transport Phase Routine*)

Crwalk (see *RW Module Transport Phase Routine*)

analout
Prints detector results and statistics for all other 2D bins.

anal1out
Makes analysis pictures showing collision sites.

analysis (see *RW Module Transport Phase Routine*)

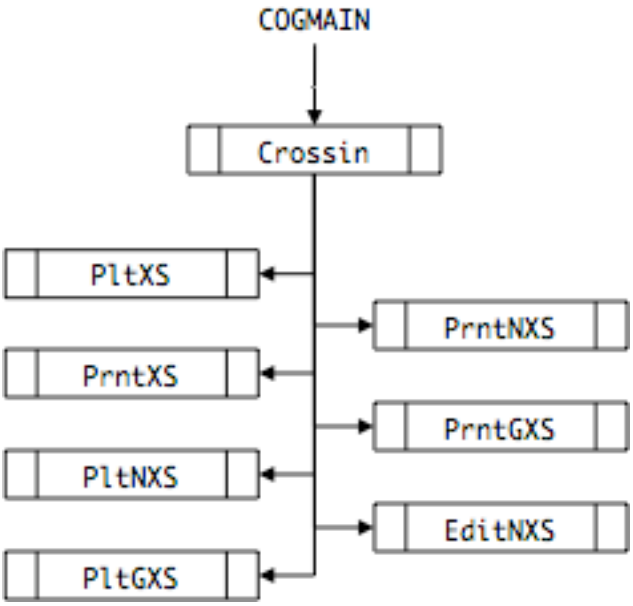
critout
Prints and plots summary of batch k's for criticality jobs.

crit1out
Prints and plots fraction of fission neutrons with energies < E.

retrace (see *RW Module Transport Phase Routine*)

wtest
Performs the statistical w-test for convergence on criticality batches.

4. TXS Module Output Phase Routines



Crossin (see *TXS Module Input Phase Routines*)

PltXS (see *TXS Module Input Phase Routines*)

PrntXS (see *TXS Module Input Phase Routines*)

PltNXS (see *TXS Module Input Phase Routines*)

PltGXS (see *TXS Module Input Phase Routines*)

PrntNXS (see *TXS Module Input Phase Routines*)

PrntGXS (see *TXS Module Input Phase Routines*)

EditNXS
Prints a list of reactions for each isotope in job if DBUG Block 'xseditflag' is set.

3.5.4. Utility Routines

These are routines that have a single simple purpose and are called by various COG modules.

1. Mem module routines

These are wrappers for the standard Unix memory-management routines and provide more detailed error messages if memory errors occur.

mallocc
Allocates a block of memory

mchlth
Resizes an allocated array.

mmfail
Issues error messages when allocation or resizing fails.

2. TXS module routines

RdAbs
Reads a file in absolute-address (unformatted) mode.

WrAbs
Writes a file in absolute-address (unformatted) mode.

3. RW module routines

RNG
Wrapper for the COG RNLF Rnadom Number Generator.
Called by: ~ 20 RW routines, 9 Sor routines, many EGS routines, 2 Det, 1 Geom, 4 Pro, numerous TXS routines.

3.5.5. Externally-Provided Software

These software libraries are provided by non-LLNL institutions and are not described further in this document.

PGPlot Graphics Subroutine Library
This library supplies routines that support COG graphics output.

EGS5 Subroutine Library
This library supplies routines that support COG electron transport.

3.6. Data Structures

3.6.1. Geometry Data Structures

1. Surface Data Storage

The following **Table 1** lists the COG surface types, the relative CPU time required to solve the associated surface equation, and a brief equation description.

The expandable **csurf** array holds defining information for all the geometrical surfaces in the problem. **Table II** describes how the surface equations are stored in the **csurf** array for the various surface types of **Table I**.

Table 1 Internal Surface Types

TABLE 1

Internal Surface Types

Number	Type	Time	Description
1	Analytic	1	Plane parallel to yz plane: $x - X_0 = 0$
2	Analytic	1	Plane parallel to xz plane: $y - Y_0 = 0$
3	Analytic	1	Plane parallel to xy plane: $z - Z_0 = 0$
4	Analytic	2	General plane: $C_0 + c_1x + c_2y + c_3z = 0$
5	Analytic	3	Sphere, center at origin: $x^2 + y^2 + z^2 - R^2 = 0$
6	Analytic	3	Sphere, center at (x_0, y_0, z_0): $(x-x_0)^2 + (y-y_0)^2 + (z-z_0)^2 - R^2 = 0$
7	Analytic	3	Cylinder, axis parallel to z-axis: $(x-x_0)^2 + (y-y_0)^2 - R^2 = 0$
8	Analytic	3	Cylinder, axis parallel to y-axis: $(x-x_0)^2 + (z-z_0)^2 - R^2 = 0$
9	Analytic	3	Cylinder, axis parallel to x-axis: $(y-y_0)^2 + (z-z_0)^2 - R^2 = 0$
10	Analytic	3	Cone, axis parallel to z-axis $(x-x_0)^2 + (y-y_0)^2 - (A + Bz)^2 = 0$
11	Analytic	3	Cone, axis parallel to y-axis: $(x-x_0)^2 + (z-z_0)^2 - (A + By)^2 = 0$
12	Analytic	3	Cone, axis parallel to x-axis: $(y-y_0)^2 + (z-z_0)^2 - (A + Bx)^2 = 0$
13	Analytic	4	General second-degree surface: $C_0 + c_1x + c_2y + C_3z + C_4x^2 + C_5xy + c_6xz + C_7yz + c_8y^2 + C_9z^2 = 0$

TABLE 1 (continued)

Internal Surface Types

Number	Type	Time	Description
14	Analytic	6	General third degree surface: Number 13 expanded to include: $c_{10}x^3 + c_{11}x^2y + c_{12}x^2z + c_{13}xy^2 + c_{14}xyz + c_{15}xz^2 + c_{16}y^3 + c_{17}y^2z + c_{18}yz^2 + c_{19}z^3$
15	Analytic	10	Elliptical torus with axis coincident with z-axis: $(x^2 + y^2 + \gamma z^2 - 2\gamma z_0 z + B_0)^2 - A_0(x^2 + y^2)$
16	Analytic	10	Elliptical torus with axis coincident with y-axis: $(x^2 + z^2 + \gamma y^2 - 2\gamma y_0 y + B_0)^2 - A_0(x^2 + z^2)$
17	Analytic	10	Elliptical torus with axis coincident with x-axis: $(y^2 + z^2 + \gamma x^2 - 2\gamma x_0 x + B_0)^2 - A_0(y^2 + z^2)$
18	Analytic	20	General fourth degree surface: Number 14 expanded to include: $c_{20}x^4 + c_{21}x^3y + c_{22}x^3z + c_{23}x^2y^2 + c_{24}x^2yz + c_{25}x^2z^2 + c_{26}xy^3 + c_{27}xy^2z + c_{28}xyz^2 + c_{29}xz^3 + c_{30}y^4 + c_{31}y^3z + c_{32}y^2z^2 + c_{33}yz^3 + c_{34}z^4$
19	Psuedo	15	Non-reentrant figure founded by N plane surfaces with N greater than four. Each plane defined with the general equation (#4). A point must be on the negative side of all M surfaces to be within this figure.
20	Psuedo	6	Second degree surface bounded by two planes. To be within the figure, a point must be on the negative side of all three surfaces.
21	Psuedo	10	Inside one second degree figure and outside another second degree figure and between two planes. A point is within the figure if it is on the negative side of both planes and on the negative side of the outer second degree surface and on the positive side of the inner such surface. Not referenced by code inputs except with the "DIRECT" statement.

TABLE 1 (continued)

Internal Surface Types

Number	Type	Time	Description
22	Psuedo	30	A surface formed by a series of second degree surfaces, each of which is to be used only between two bounding planes. At any bounding plane, the intersection of the two second degree surfaces used on either side of the plane describes the same identical curve. The defining surfaces reach a single point at each of the two ends of the figure. The surface is defined as being positive at points outside the closed figure and negative inside. The first plane is used only on its positive side while the last plane is used only on its negative side. All the second degree surfaces must be negative at points within the figure.
23	Psuedo	30	A surface formed by putting together a series of four-sided prisms. The set has a common top and bottom plane and adjacent prisms have common sides that are exactly coincident with each other. To be within the figure, the top and bottom planes must be negative as well as the two side planes. The bounding planes must be positive for the first boundary and negative for the second.
24	Psuedo	30	Like #22 except that the ends do not necessarily reach a single point and the intersections of adjacent second degree curves with their bounding plane do not necessarily result in the same curve.
25	Psuedo	30	Surface is defined by a tabular representation with a fixed x and a fixed y grid and corresponding values of z provided at the x,y intersections. The values of z between the tabular values are obtained by linear interpolations in both x and y (a second degree fit).
26	Psuedo	20	Surface defining the outside of a Yin-Yang coil configuration.
27	Psuedo	8	Similar to #21 but without the restraint of being between two plane surfaces. Again, not referenced by normal code inputs except with the "DIRECT" input.
28	Psuedo	20	Inside one non-reentrant figure bounded by N plane surfaces while being outside of another non-reentrant figure also bounded by another set of M plane surfaces. Not referenced by code inputs except with the "DIRECT" option.

Table 2 Surface Definition Data Storage

Table II
Surface Definition Data Storage

Number	Relative Location	Data Stored	Comment
1	L	x_0	Plane yz
2	L	y_0	Plane xz
3	L	z_0	Plane xy
4	L…L+3	c_0, c_1, c_2, c_3	General Plane
5	L	R^2	Sphere @ origin
6	L…L+3	x_0, y_0, z_0, R^2	General sphere
7	L…L+2	x_0, y_0, R^2	Cylinder-Z
8	L…L+2	x_0, z_0, R^2	Cylinder-Y
9	L…L+2	y_0, z_0, R^2	Cylinder-X
10	L…L+5	x_0, y_0, A, B, AB, B^2	Cone-Z
11	L…L+5	x_0, z_0, A, B, AB, B^2	Cone-Y
12	L…L+5	y_0, z_0, A, B, AB, B^2	Cone-X
13	L…L+9	$c_0, c_1, \ldots c_9$	General 2nd degree
14	L…L+19	$c_0, c_1, \ldots c_{19}$	General 3rd degree
15	L	$A_0 (= 4z_0^2)$	Elliptical Torus - Z
	L+1	$B_0 (= x_0^2 - a^2 + z^2)$	a = minor radius in plane of torus
	L+2	$\gamma (= b^2 / a^2)$	b = minor radius in plane perpendicular to torus
	L+3	$C_0 (= 2 \gamma z_0)$	
16	L	$A_0 (= 4x_0^2)$	Elliptical Torus -Y
	L+1	$B_0 (= x_0^2 - a^2 + y_0^2)$	
	L+2	$\gamma (= b^2 / a^2)$	
	L+3	$C_0 (= 2 \gamma y_0)$	

Table II (continued)
Surface Definition Data Storage

Number	Relative Location	Data Stored	Comment
17	L	A_0 (= $4y_0^2$)	Elliptical Torus - X
	L+1	B_0 (= $y_0^2 - a^2 + x^2$)	a = minor radius in plane of torus
	L+2	γ (= b^2 / a^2)	b = minor radius in plane perpendicular to torus
	L+3	C_0 (= $2 \gamma x_0$)	
18	L…L+34	$c_0, c_1, \ldots c_{34}$	General 4th degree equation
19	L	N: number of bounding surfaces	General box
	L+1…L+N	Temp. storage for $f_n(x,y,z)$	
	L+N+1… L+N+4	c_0, c_1, c_2, c_3 for 1st surface	
	L+N+5… L+N+8	As above, for 2nd surface	
	……	etc.	
20	L	Temp. storage for $f_n(x,y,z)$, top plane	Finite cylinder
	L+1	As above, for bottom plane	
	L+2	As above, for 2nd degree surface	
	L+3…L+6	c_0, c_1, c_2, c_3 for top plane	
	L+7…L+10	As above, for bottom plane	
	L+11…L+20	$c_0, c_1, \ldots c_9$ for 2nd degree surface	

Table II (continued)
Surface Definition Data Storage

Number	Relative Location	Data Stored	Comment
21	L	Temp. storage for f(x,y,z) for top plane	Volume between one 2nd degree surface and another, bounded by two planes
	L+1	As above, for bottom plane	
	L+2	As above, for outer 2nd degree surface	
	L+3	As above, for inner 2nd degree surface	
	L+4…L+7	c_0, c_1, c_2, c_3 for top plane	
	L+8…L+11	As above, for bottom plane	
	L+12…L+21	c_0, c_1,… c_9 for outer 2nd degree surface	
	L+22…L+31	As above, for inner 2nd degree surface	
22	L	N: number of 2nd degree surfaces	Surface of revolution
	L+l...L+4	c_0, c_1, c_2, c_3 coefficients for 1st plane	The defined figure does not exist on the negative side of this plane.
	L+5…L+8	As above, for 2nd plane	
	……		
	L+1+4N... L+4+4N	As above, for N+1st plane	The defined figure does not exist on the positive side of this plane.
	L+5+4N... L+14+4N	c_0, c_1,… c_9 for first 2nd degree surface	Used between 1st and 2nd planes.
	L+15+4N...	Coef. of remaining 2nd degree surfaces	

Table II (continued)
Surface Definition Data Storage

Number	Relative Location	Data Stored	Comment
23	L	N: the number of prisms	Volume between one 2nd degree surface and another, bounded by two planes
	L+l...L+4	c_0, c_1, c_2, c_3 coefficients for the top-most plane	
	L+5…L+8	As above, for the bottom-most plane	
	L+9…L+12	As above, for the first defined plane	The defined figure does not exist on the negative side of this plane.
	L+13…L+16	As above, for the 2nd defined plane	This is between the first and second prism.
	L+17…L+20	As above, for the 3rd defined plane	
	L+9+4N…	As above, for the (N+1)st defined plane	The figure is undefined on the positive side of this plane.
	L+13+4N…	As above, for outer plane of the first prism.	Must be negative to be on the inside of the prism.
	L+17+4N…	As above, for inner plane of the first prism	Must be negative to be on the inside of the prism.
	……		
	L+21+4N...	As above, for outer plane of the 2nd prism.	
	……	Etc.	
24			Identical to surface 22

Table II (continued)
Surface Definition Data Storage

Number	Relative Location	Data Stored	Comment
25	L	N: the number of intervals in the x-direction	Surface is defined by a tabular grid for z(x,y).
	L+l	M: the number of intervals in the y-direction	
	L+2…L+2+N	x-grid values (increasing order)	
	L+3+N… L+3+N+M	y-grid values (increasing order)	
	L+4+N+M…	z-values on the x,y grid.	$z(x_i,y_j)$ is stored at: L3+i+(j-1)(N+1), where L3 =L+3+N+M.
26	L…L+5	Temporary storage for 6 cylinders' coefs	Yin_Yang Coil
	L+6…L+22	Temporary storage for 17 planes' coefs	.
	L+23…L+32	c0, c1,…c9 for cylinder 1	
	L+33…L+42	As above, for cylinder 2	
	……		
	L+73…L+82	As above, for cylinder 6	
	L+83…L+86	c0, c1 ,c2, c3 for plane 1	
	……		
	L+147…L+150	c0, c1, c2, c3 for plane 17	

Table II (continued)
Surface Definition Data Storage

Number	Relative Location	Data Stored	Comment
27	L	Temporary storage for f(x,y,z) defining outer surface	
	L+l	Temporary storage for f(x,y,z) defining inner surface	
	L+2…L+11	c0, c1,c2 …c9 for outer surface	
	L+12…L+22	c0, c1,c2 …c9 for inner surface	
28	L	N: number of surfaces	
	L+1…L+2N	Temporary storage for $f_n(x,y,z)$	Outer surfaces followed by inner surfaces.
	L+2N+1…L+2N+4	c0, c1, c2, c3 for first outer surface	
	L+2N+5…L+2N+8	c0, c1, c2, c3 for second outer surface	
	……		
	L+6N+1…L+6N+4	c0, c1, c2, c3 for first inner surface	
	L+6N+5…L+6N+8	c0, c1, c2, c3 for second inner surface	
	……		

2. Volume Data Storage

The basic COG volume element is the sector, which is a volume bounded by surfaces whose descriptions are stored in the **csurf** array outlined above.

Each sector is defined by a Sector Description record stored in the **sd** array. An **sd** record has a variable length depending only on the number of surfaces used to describe the sector's boundaries.

Table 3 Sector Description (sd) Array

Location	Item	Description	Type
1	nsty	Type of sector	Integer
2	nmat nuntn	Material number (if nsty=1) Unit number (if nsty = 2)	Integer
3	nreg	Region number	Integer
4	xdf	Density factor	Real
5	neqs	Count of surfaces bounding sector	Integer
6	nsecnCog	COG-assigned sector number	Integer
7 – 8	names	Sector name	String
9	nsecnUser	User-specified sector number	Integer
10	dens	Material density	Real
11 – 36	trans (array)	Coordinate transformation Matrix	Real
37 – 36+neqs	nsureq (array)	Internal surface number, with sign, for each bounding surface	Integer
37+neqs – 36+2*neqs	nsureqi (array)	Index into array of user surface numbers, for each bounding surface	Integer
37+2*neqs – 36+3*neqs	kpos (array)	Index into ordering array which determines order of surface evaluation	Integer

3.6.2. Event History Storage (EHS) Data Structures

The **EHS** array contains the event history of each primary (source) particle and those of its secondaries (particles arising from collisions in the problem materials). COG computes detector scores at the end of each source particle trajectory, when it and all its "daughter" particles have been tracked until they escaped the system or were absorbed. Each event in the history of a source particle and its progeny - such as a boundary crossing, a scattering, or an absorption event - is recorded in the Event History Store (EHS) array.

Each event is stored as a fixed-length record in the EHS array.

Table 4 Event History Store Record

Record Item name	Description
nevent	event type
x,y,z	position of event site
u,v,w	particle direction at event site
age	particle age
eng	particle energy
ntype	particle type number
vel	particle velocity
wate	particle statistical weight
noc	number of collisions
imat	material number
ireg	region number
tdf	density factor
stot	total cross section
nreact	reaction number
xmfp	mean-free-paths left to travel
edep	energy deposition times statistical weight
nsave	number of sector containing particle
xlambda(1)-(2)	decay constants for activation
ex1	distance to next boundary
cosn	cosine of angle between trajectory and surface normal, for case nevent=40

Table 5 Events

Event # (nevent)	Description
10	particle taken from source
20	particle taken from secondary bank
30	particle taken from fission bank
40	crossed a boundary between two different regions
50	enters into a collision
51	exits from a collision
52	modified by path stretching
60	encounters a reflecting boundary
61	returns from a reflecting boundary
70	encounters an albedo boundary
71	returns from an albedo boundary
80	encounters a periodic boundary
81	returns from a periodic boundary
90	particle absorbed in an infinite absorber
91	terminated because energy out of range of cross sections
100	encounter a vacuum boundary (or its equivalent)
120	entered into a russian roulette game
121	killed by losing a russian roulette game
122	survived a russian roulette game
130	terminated by a time cut-off imposed by the geometry description
131	terminated by time constraint
135	encounters a time boundary imposed by the geometry's description
140	enters into a weight window modification
141	exits from a weight window modification
150	entered into a splitting modification
151	exited from a splitting modification
162	terminated in forced collision process
180	terminated due to a problem specification error

3.6.3. Input File

See user guide

3.6.4. Common Block Usage

See include files

4. DATA LIBRARIES

The following data libraries are available in this release:

4.1. Neutron Activation Library

ACTL92	*LLNL's ACTL 1992*

4.2. Photon Libraries

COGGXS	defaults to EPDL97
EPDL89	LLNL's EPDL-1989
EPDL97	LLNL's EPDL-1997
EPICS2017	IAEA-2017
EPICS2023	IAEA-2023

4.3. Nuclear Resonance Fluorescence Library

COGNRF	LLNL's Dr. James Hall data

4.4. Photonuclear Libraries

COGPNUC	defaults to IAEAPNUC
IAEAPNUC	IAEA
PN.ENDFB7R1	ENDF/B-VII.1
PN.MCNP.70u	MCNP's .70u
PN.ENDFB8R0	ENDF/B-VIII.0
PN.IAEA2019	IAEA-2019

4.5. Radiation Simulation Library

COGRS	developed by LLNL's Dr. E. M. Lent

4.6. Delayed Fission Gamma Libraries

DFG.ENDFB7R1	ENDF/B-VII.1
DFG.JEFF3.1.1	JEFF3.1.1
DFG.JENDL4	JENDL4
DFG.ENDFB8R0	ENDF/B-VIII.0

4.7. Neutron Libraries

ENDFB6R7	ENDF/B-VI.7
ENDFB6R8	ENDF/B-VI.8
ENDFB7R0	ENDF/B-VII.0
ENDFB7R0.BNL	BNL's ENDF/B-VII.0
ENDFB7R1	ENDF/B-VII.1

ENDFB7R1.BNL	BNL's ENDF/B-VII.1
ENDFB8R0	ENDF/B-VIII.0
ENDFB8R0.ACE	ENDF/B-VIII.0 (ACE format)
ENDFB8R1	ENDF/B-VIII.1
ENDL2008	LLNL's ENDL-2008
ENDL90	LLNL's ENDL-1
ENDL99	LLNL's ENDL-1999
JEFF3.1	JEFF3.1
JEFF3.1.1	JEFF3.1.1
JEFF3.1.2	JEFF3.1.2
JENDL3.3	JENDL3.3
JENDL4	JENDL4
MCNP.50c	MCNP's .50c
MCNP.51c	MCNP's .51c
MCNP.55c	MCNP's .55c
MCNP.66c	MCNP's .66c
MCNP.70c	MCNP's .70c
RED2002	Hybrid ENDFB/ENDL library by Dr. D. Cullen

4.8. Probability Table Libraries

PT.ENDFB7R0.BNL	BNL's ENDF/B-VII.0
PT.ENDFB7R1.BNL	BNL's ENDF/B-VII.1
PT.ENDFB8R0	ENDF/B-VII.0
PT.JEFF3.1	JEFF3.1
PT.JEFF3.1.1	JEFF3.1.1
PT.JEFF3.1.2	JEFF3.1.2
PT.MCNP.66c	MCNP's .66c
PT.MCNP.70c	MCNP's .70c

4.9. Thermal Libraries

T.ENDFB3R0	ENDF/B-III.0
T.ENDFB6R0	ENDF/B-VI.0
T.ENDFB6R2	ENDF/B-VI.2
T.ENDFB7R0	ENDF/B-VII.0
T.ENDFB7R0.BNL	BNL's ENDF/B-VII.0
T.ENDFB7R0.LANL	LANL's ENDF/B-VII.0
T.ENDFB7R1	ENDF/B-VII.1
T.ENDFB7R1.BNL	BNL's ENDF/B-VII.1
T.ENDFB8R0	ENDF/B-VIII.0
T.ENDFB8R1.RT	ENDF/B-VIII.1 (room temperature)
T.JEF2.2	JEF2.2
T.JEFF3.0	JEFF3.0
T.JEFF3.1	JEFF3.1
T.JEFF3.1.1	JEFF3.1.1
T.JEFF3.1.2	JEFF3.1.2

4.10. Dosimetry Libraries

IRDF2002	IRDF-2002
IRDFF1.02	IRDFF Release 1.02
IRDFF1.05	IRDFF Release 1.05
IRDF-II	ENDF/B-VIII.0

INDEX

A

Acoll 26
aget 19
AIso 26
analout 28
anal1out 28
analysis 18, 28
AReac 25
asmrd 8
ATXSM 14

B

BetheB 15
BStep 19

C

ChckAXS 14
ChckDXS 14
ChckGXS 12
ChckNXS 12
ChckPNUC 12
ChckSAB 12
ChckXXS 12
CheckIntc 19
CheckPEGS 14
CheckXS 12
COGCSP 26
COGPP 26
CPTrans (Charged Particle Transport Module) 5
crit1out 28
critin 8
critout 28
Crossin 12, 29
Crwalk 17, 28
CSPKey 26

D

DColl 26
DefVolPtDet 10
Det (Detector Module) 4
DetBIN 11
DetBound 17
DetDef 10
detectin 10
DetList 11
DetMask 11
DetPoint 17
DetPulse 18
DetReact 17
DetRF 11
DetTrack 18
DetUser 18
DIso 25
DoAGP 24
DoPFE 25
DoPicP 26
DoPix 26
DoRestartCrit 18
DoRSD 18

dose 21
Dose59 21
Dose91 21
DoSummCP 18
DoWtStat 24
Dreac 25
DTXSM 14
dylinksub 11
dyloadlib 11

E

EditNXS 29
EGS (Electron Gamma-Ray Shower Module) 4
EGS4 Subroutine Library 30
EGSin 14
EGSInit 15
ehlossr 15
EProd 21

F

f 20
ffinp 7
fmed 26
FndInt 26
FndSec 26
FnShEx 26
FREYA (Fission Particle Transport Module) 5

G

Gcoll 24
Geom (Geometry Module) 3
geomin 7, 25
geompe 20
geompe1 20
geomrw 19
geomrw1 19
GetAvek 18
GetCritSrc 18
GetNextSecList 19
GetNMult 18
GetNXG 24
GetSorHeadOffset 9
GetUsrLib 11
getusrsorpart 9
Giso 24
GisoB 24
Greac 24
GreacB 24
Grf (Graphics Module) 5
GsorP 22
GSurvival 24
GTXSM 13

H

HATCH 15

I

include (Include Module) 5
InDRF 21
InitRW 18
InMask 21
interc 20

L

Lattice (Lattice Module) 5, 8
LF (Lagged Fibonacci Random Number Generator Module) 5

M

Mag (Magnetic Field Module) 5
Main (Main Module) 3
mallocc 30
MaskRegIn 11
mchlth 30
Mem (Memory Management Module) 5
MixDef 13
mmfail 30
MPI (Message Passing Interface Module) 6

N

Ncoll 24
NElastic 24
Niso 24
NisoB 24
NNonEl 24
Nreac 24
NReacB 24
NTXSM 13

O

OptPath 18

P

pathBC 26
pathFI 26
pathFS 26
pathl1 26
pathOu 26
pathRU 27
pathUn 27
PEGS (PEGS Module) 6, 16
PGPlot Graphics Subroutine Library 30
PhNuc 25
plot3d 27
PlotLD 27
PltGXS 14, 29
PltNXS 14, 29
PltXS 13, 29
PNTXSM 13
PP (Parallel Processing Support Module) 6
PrntGXS 14, 29
PrntNXS 14, 29
PrntXS 14, 29
Pro (Proton Transport Module) 5
pro1d 27
pro24x 28
pro2d 28
ProTXSM 15
prout 28
PrSwp 27
pulsein 11

R

RdAbs 30
RdAXS 14
RdDFG 13
RdDXS 14
RdGXS 13
RdLEX 12
RdNXS 13
RdPNUC 13
RdProSP 15
RdXXS 13
ReactXS 21
retrace 18, 28
RNG 30
rsa9
rsb 19
rsc19
rse9
rsp 9
rst 9
RW (Random Walk Module) 4
rww 20

S

Scat 23
ScatA 25
ScatD 25
ScatG 24
ScatN 24
score 21
scores 21
sela 23
sele 23
selp 23
selt 23
SetVS 27
Sor (Source Module) 4
sorin 8
sour 22
Srwalk 17, 28
Stern2 16
Stern3 16
SternInt 16
surfac 8

T

to_usrdet 21
TRProc 11
TXS (Total Cross Section Module) 4

U

uncpe 23
unitrd 8

V

volume 27

X

XTXSM 14

W

walkin 11
walkin2 12
wkchrn 20
WrAbs 30
wsor 18
wtest 28