CERN–2005–010
INFN TC_05/11
SLAC–R–773
12 October 2005

ORGANISATION EUROPÉENNE POUR LA RECHERCHE NUCLÉAIRE
CERN EUROPEAN ORGANIZATION FOR NUCLEAR RESEARCH

Fluka:

a multi-particle transport code

(Program version 2005)

Alfredo Ferrari, Paola R. Sala, Alberto Fassò, Johannes Ranft

GENEVA
2005

CERN–400 copies printed–October 2005

Abstract

This report describes the 2005 version of the FLUKA particle transport code. The first part introduces the basic notions, describes the modular structure of the system, and contains an installation and beginner's guide. The second part complements this initial information with details about the various components of FLUKA and how to use them. It concludes with a detailed history and bibliography.

Preface

The INFN–CERN Collaboration Agreement for the Maintenance and Development of the FLUKA Code of December 2003 called for the publication of two major reports, focused respectively more on the technical aspects (this report), and on the description of the physics models embedded into the Code (to be published next year).

This document is a reference guide for the users of the FLUKA particle transport code, so as to provide them with a tool for the correct input preparation and use of the code, a description of the basic code structure, and of the history of its evolution. In particular, this guide is intended to document the 2005 version of FLUKA which represents a fundamental evolution in FLUKA development with respect to the previous releases, from both the scientific and technical points of view. This development is one of the main tasks carried out under the INFN–CERN Collaboration Agreement for the Maintenance and Development of the FLUKA Code. Notwithstanding the efforts of the authors, it is not possible to guarantee that this guide is free from error. Furthermore, since the FLUKA code is continually evolving, this guide will also evolve with time. This document also represents the first and basic reference to be used in citation of FLUKA, together with Ref. [62] (a full list of References starts on page 356). In using the code, the user agrees on the authorship and copyright and hence is bound to quote the above references.

This guide is organized as follows. After the FLUKA Licence, which the user is required to read carefully, Part I contains a summary description of what FLUKA is and what its capabilities are (Chapter 1), a Beginner's Guide (Chapter 2), the Installation Guide (Chapter 3), and a list of the main modules (Fortran files) of which FLUKA is composed (Chapter 4).

Part II is the real User Guide, where the user can find a detailed description of what is needed to run FLUKA: particles and materials considered in the code (Chapter 5), the rules for building FLUKA input (Chapter 6), each single FLUKA command and corresponding parameters (Chapter 7), the Geometry system (Chapter 8), the standard FLUKA output (Chapter 9), how to write the user routines (Chapter 12), etc.

At the end of Part II, the history of FLUKA is reported, putting emphasis on the third generation of the code evolution (from 1988 to the present version), giving more details about the code structure and all the fundamental physics models used in FLUKA.

It must be noted that early versions of the FLUKA hadronic event generator as implemented in other codes (in particular GEANT3) should be referenced as such (e.g. GEANT-FLUKA) and not as FLUKA. They have little in common with the present version and should be considered virtually obsolete. The proper reference to GEANT-FLUKA is Ref. [52].

Please contact A. Ferrari or A. Fassò (see email addresses below) for any comment on or criticism about this manual and/or the code.

The FLUKA Authors:

Alfredo Ferrari,	*CERN and INFN Milan*	(`Alfredo.Ferrari@cern.ch`)
Paola R. Sala,	*INFN Milan*	(`Paola.Sala@mi.infn.it`)
Alberto Fassò,	*SLAC*	(`fasso@slac.stanford.edu`)
Johannes Ranft,	*Physics Department, University of Siegen*	(`Johannes.Ranft@cern.ch`)

Acknowledgements

The authors would like to thank all the members of the FLUKA Collaboration for their invaluable help and patience in reading, checking, and amending the manuscript.

The FLUKA Collaboration:

D. Alloni, F. Ballarini, G. Battistoni, M. Campanella, M. Carboni, F. Cerutti, A. Clivio, A. Empl, A. Fassò, A. Ferrari, E. Gadioli, E. Gadioli-Erba, M.V. Garzelli, E. Giroletti, A. Mairani, A. Mostacci, S. Muraro, A. Ottolenghi, V. Parini, M. Pelliccioni, L.S. Pinsky, J. Ranft, S. Roesler, P. R. Sala, D. Scannicchio, G. Smirnov, S. Trovati, R. Villari, V. Vlachoudis, T. Wilson, N. Zapp

This work has been supported by INFN and CERN in the framework of the Collaboration Agreement for the development, maintenance and release of the FLUKA software. This work was partially supported under DOE contract DE-AC02-76-SF00515, NASA Grant NAG8-1901.

FLUKA User license, as established by the FLUKA Coordination Committee

Copyright statement and license conditions

This software results from work performed by Alberto Fassò, Alfredo Ferrari, Johannes Ranft and Paola Sala.

INFN and CERN are the exclusive source of distribution of the code, bug fixes and documentation of the FLUKA software. Each official version of FLUKA is identified by a numbering scheme specifying major and minor releases.

The FLUKA Coordination Committee or its delegates are able to grant any of the permissions noted in this License Agreement as requiring a specific consent. Any such consent may only be granted in writing.

Installation, use, reproduction, display of the FLUKA software ("FLUKA"), in source and binary forms, are permitted free of charge on a non-exclusive basis for internal scientific, non-commercial and non-weapon-related use by non-profit organizations only. Any exercise of these rights is subject to the following conditions:

1. Insertion of the FLUKA code, in whole or in part, into other codes, or its translation into any other computer language are possible only after obtaining prior written permission. Modifications of the FLUKA code are permitted for use by the licensee only, unless authorized in written.
2. FLUKA is non-transferable, non-sub-licensable and may not be distributed in any way, without express written consent, whether in original or modified form. Site-wise or collaboration-wise conditions can be agreed with the FLUKA Coordination Committee.
3. Notwithstanding the above, the licensee may modify and sub-license FLUKA User Routines to third parties in so far as their purpose is limited to the adaptation of input and output interfaces of FLUKA and their modification does not circumvent, replace, add to or modify any of the functions of FLUKA, or extract specific isolated results from any of the individual internal physics models embedded within FLUKA.
4. The licensee shall forthwith license all its modifications of FLUKA to the FLUKA copyright holders, at no cost and with no limitation of use. The licensee acknowledges that the FLUKA copyright holders may insert such modifications into future releases of FLUKA, subject to appropriate acknowledgment of the licensee's contribution.
5. Any publication by the licensee with respect to FLUKA or results obtained from it shall explicitly acknowledge FLUKA by quoting its set of references and the FLUKA copyright holders. The licensee shall not without prior written permission publish documents or results based on a modified form of FLUKA, unless the modification exclusively concerns User Routines for the adaptation of its input and output interfaces which comply with the same restrictions, as defined in section 3) as those which apply to sub-licensing. Any publication of documents or results shall be based only on official FLUKA versions as obtained from the FLUKA website (http://www.fluka.org) or from any authorized mirror. Publication here implies any legal publication to any third party, whether verbal, electronic, visual, in writing or otherwise.
6. The licensee shall ensure that the FLUKA references, copyright statement and license conditions are not altered or removed from FLUKA. Any integration of any portion of FLUKA, in modified or in unmodified form, into any other software package must preserve the internal copyright notices in those portions of FLUKA that have been employed, and must reproduce such notices within any additional global notices included along or embedded within the software into which FLUKA has been integrated.

Any portion of FLUKA so integrated, whether modified or unmodified shall continue to be subject to these license conditions.

7. Nothing in this license shall be construed as to grant any rights in any of the FLUKA versions since 1989. In addition, users are not permitted to circumvent any protection in prior distributions of FLUKA that provided for a preset expiration date of the code
8. Versions or parts of the FLUKA source code, entrusted to individuals or groups prior to the enactment of the CERN-INFN Collaboration Agreement, which are listed in Chapter 5 of Annex 1 of the EP-AB-INFN Scientific Agreement (19-02-2003), together with the agreed conditions of use, are subject to this License Agreement in addition to any other restrictions on the scope of use that may have been part of the initial use grant.
9. Commercial use of FLUKA, outside the scope of this license, must be negotiated with the copyright holders
10. DISCLAIMER
    THIS SOFTWARE IS PROVIDED BY THE FLUKA COPYRIGHT HOLDERS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, IMPLIED WARRANTIES OF MERCHANTABILITY, OF SATISFACTORY QUALITY, AND FITNESS FOR A PARTICULAR PURPOSE OR USE ARE DISCLAIMED. THE FLUKA COPYRIGHT HOLDERS AND THE AUTHORS MAKE NO REPRESENTATION THAT THE SOFTWARE AND MODIFICATIONS THEREOF, WILL NOT INFRINGE ANY PATENT, COPYRIGHT, TRADE SECRET OR OTHER PROPRIETARY RIGHT.
11. LIMITATION OF LIABILITY
    THE FLUKA COPYRIGHT HOLDERS AND THE AUTHORS SHALL HAVE NO LIABILITY FOR DIRECT, INDIRECT, SPECIAL, INCIDENTAL, CONSEQUENTIAL, EXEMPLARY, OR PUNITIVE DAMAGES OF ANY CHARACTER INCLUDING, WITHOUT LIMITATION, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES, LOSS OF USE, DATA OR PROFITS, OR BUSINESS INTERRUPTION, HOWEVER CAUSED AND ON ANY THEORY OF CONTRACT, WARRANTY, TORT (INCLUDING NEGLIGENCE), PRODUCT LIABILITY OR OTHERWISE, ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

REQUESTS MADE BY THE FLUKA AUTHORS TO ALL USERS

- All licensees are requested to report as soon as practical to the Authors any errors or bugs found in any portion of FLUKA and its accompanying documentation.
- All licensees are requested to forward copies of all comparisons that they may make between FLUKA results and data or other codes as soon as practical. The Authors agree to keep any such communications confidential unless otherwise notified by the contributing user.
- The Authors reserve the right of publishing any benchmarking and/or comparisons of the distinct separate performance of the individual internal models that are embedded into FLUKA, whether the comparisons are with data or with other codes. The Authors would also like to convey a general willingness to conduct any such benchmarking efforts either upon request or in collaboration with interested parties. In case of doubt please contact the Authors.
- Users should exclusively download FLUKA from the official FLUKA website (http://www.fluka.org) or one of the authorized mirror sites. Users are invited to regularly update their FLUKA version to profit for improvements and bug fixes.
- Users are invited to use reasonably updated versions of the code for publications. Publications of results based on those FLUKA versions that are declared unsupported and obsolete on the official FLUKA website shall be avoided.
- Users should address any request of consent to one member of the FLUKA Coordinating Committee, which at present is composed as follows:

Giuseppe Battistoni Giuseppe.Battistoni@mi.infn.it (chairman)
Enrico Chiaveri Enrico.Chiaveri@cern.ch
John Harvey John.Harvey@cern.ch

Johannes Ranft Johannes.Ranft@cern.ch
Paola Sala Paola.Sala@mi.infn.it

The use of the FLUKA code must be acknowledged explicitly by quoting the present report and the following reference:

– A. Fassò, A. Ferrari, S. Roesler, P.R. Sala, G. Battistoni, F. Cerutti, E. Gadioli, M.V. Garzelli, F. Ballarini, A. Ottolenghi, A. Empl and J. Ranft, "The physics models of FLUKA: status and recent developments", Computing in High Energy and Nuclear Physics 2003 Conference (CHEP2003), La Jolla, CA, USA, March 24–28, 2003, (paper MOMT005), eConf **C0303241** (2003), arXiv:hep-ph/0306267.

Additional FLUKA references can be added, provided they are relevant for the FLUKA version under consideration.

This set of references is subject to change in time. New ones will be communicated, when necessary, in the Release Notes of new FLUKA versions.

Contents

Part I

A summary description of FLUKA

Chapter 1

Introduction

1.1 What is FLUKA?

FLUKA is a general purpose tool for calculations of particle transport and interactions with matter, covering an extended range of applications spanning from proton and electron accelerator shielding to target design, calorimetry, activation, dosimetry, detector design, Accelerator Driven Systems, cosmic rays, neutrino physics, radiotherapy etc.

The highest priority in the design and development of FLUKA has always been the implementation and improvement of sound and modern physical models. Microscopic models are adopted whenever possible, consistency among all the reaction steps and/or reaction types is ensured, conservation laws are enforced at each step, results are checked against experimental data at single interaction level. As a result, final predictions are obtained with a minimal set of free parameters fixed for all energy/target/projectile combinations. Therefore results in complex cases, as well as properties and scaling laws, arise naturally from the underlying physical models, predictivity is provided where no experimental data are directly available, and correlations within interactions and among shower components are preserved.

FLUKA can simulate with high accuracy the interaction and propagation in matter of about 60 different particles, including photons and electrons from 1 keV to thousands of TeV, neutrinos, muons of any energy, hadrons of energies up to 20 TeV (up to 10 PeV by linking FLUKA with the DPMJET code) and all the corresponding antiparticles, neutrons down to thermal energies and heavy ions. The program can also transport polarised photons (e.g., synchrotron radiation) and optical photons. Time evolution and tracking of emitted radiation from unstable residual nuclei can be performed on line.

FLUKA can handle even very complex geometries, using an improved version of the well-known Combinatorial Geometry (CG) package. The FLUKA CG has been designed to track correctly also charged particles (even in the presence of magnetic or electric fields). Various visualisation and debugging tools are also available.

For most applications, no programming is required from the user. However, a number of user interface routines (in Fortran 77) are available for users with special requirements.

The FLUKA physical models are described in several journal and conference papers; on the technical side the stress has been put on four apparently conflicting requirements, namely efficiency, accuracy, consistency and flexibility.

Efficiency has been achieved by having a frequent recourse to table look-up sampling and a systematic use of double precision has had a great impact on overall accuracy: both qualities have benefited from a careful choice of the algorithms adopted. To attain a reasonable flexibility while minimising the need for user-written code, the program has been provided with a large number of options available to the user, and has been completely restructured introducing dynamical dimensioning.

Another feature of FLUKA, probably not found in any other Monte Carlo program, is its double capability to be used in a biased mode as well as a fully analogue code. That means that while it can be used to predict fluctuations, signal coincidences and other correlated events, a wide choice of statistical techniques are also available to investigate punchthrough or other rare events in connection with attenuations by many orders of magnitude.

1.2 A quick look at FLUKA's physics, structure and capabilities

Here we only give a very short summary list of the capabilities and limitations of FLUKA, since this is meant to be mainly a practical guide. More detailed descriptions of the physical models, algorithms and techniques will be found in cited references and hopefully in a future more comprehensive Reference Manual.

1.2.1 Physics

1.2.1.1 Hadron inelastic nuclear interactions

The FLUKA hadron-nucleon interaction models are based on resonance production and decay below a few GeV, and on the Dual Parton model above. Two models are used also in hadron-nucleus interactions. At momenta below 3–5 GeV/c the PEANUT package includes a very detailed Generalised Intra-Nuclear Cascade (GINC) and a preequilibrium stage, while at high energies the Gribov-Glauber multiple collision mechanism is included in a less refined GINC. Both modules are followed by equilibrium processes: evaporation, fission, Fermi break-up, gamma deexcitation. FLUKA can also simulate photonuclear interactions (described by Vector Meson Dominance, Delta Resonance, Quasi-Deuteron and Giant Dipole Resonance). A schematic outline is presented below:

- Inelastic cross-sections for hadron-hadron interactions are represented by parameterised fits based on available experimental data [118].
- For hadron-nucleus interactions, a mixture of tabulated data and parameterised fits is used [15, 106, 120, 121, 159].
- Elastic and charge exchange reactions are described by phase-shift analyses, and eikonal approximation.
- Inelastic hadron-hadron interactions are simulated by different event generators, depending on energy:
  - Momentum < 20 TeV/c and > 5 GeV/c:
    Dual Parton Model (DPM) [36]. The version used in FLUKA has been derived by A. Ferrari and P.R. Sala [55, 66, 67, 70] from the original version by J. Ranft and collaborators [135, 136]. A description of modifications and improvements can be found in [41, 70]
  - Momentum from threshold to 5 GeV/c:
    Resonance production and decay model [70] (Improved version of Hänssgen *et al.* model [79–85])
- Inelastic hadron-nucleus interactions are simulated by different event generators depending on energy and projectile:
  - Momentum < 20 TeV/c and > 5 GeV/c:
    Glauber-Gribov multiple scattering followed by Generalised Intranuclear Cascade (GINC)
  - Below 5 GeV/c for nucleons, anti-nucleons and pions; below 1.5 GeV kinetic for kaons:
    Preequilibrium-cascade model PEANUT (Ferrari-Sala) [55, 66]
  - In between PEANUT and DPM for kaons: Hänssgen K. *et al.* GINC modified to take into account correlations among cascade particles and more refined nuclear effects (Ferrari-Sala)
- All three models include evaporation and gamma deexcitation of the residual nucleus [68, 69]. Light residual nuclei are not evaporated but fragmented into a maximum of 6 bodies, according to a Fermi break-up model.
- Treatment of antiparticle capture: antinucleons according to resonance model, π^-, K^- and μ^- by the preequilibrium-cascade model.

1.2.1.2 Elastic Scattering

- Parameterised nucleon-nucleon cross-sections.
- Tabulated nucleon-nucleus cross sections [120, 121].
- Tabulated phase shift data for pion-proton and phase-shift analysis for kaon-proton scattering.
- Detailed kinematics of elastic scattering on hydrogen nuclei and transport of proton recoils (Ferrari-Sala).

1.2.1.3 Nucleus-Nucleus interactions

Nuclear interactions generated by ions are treated through interfaces to external event generators.

- Above 5 GeV per nucleon: DPMJET-II or DPMJET-III , with special initialisation procedure.
- Between 0.1 and 5 GeV per nucleon: modified RQMD

1.2.1.4 Transport of charged hadrons and muons

An original treatment of multiple Coulomb scattering and of ionisation fluctuations allows the code to handle accurately some challenging problems such as electron backscattering and energy deposition in thin layers even in the few keV energy range.

Energy loss:

- Bethe-Bloch theory [19–21, 29, 30]
- optional delta-ray production and transport with account for spin effects and ionisation fluctuations.

The present version includes a special treatment [57] which combines delta ray production with properly restricted ionisation fluctuations and includes corrections for particle spin and electrons/positrons and "distant collision" straggling corrections (similar to Blunck-Leisegang ones).

- Shell and other low-energy corrections derived from Ziegler [178].
- Density effect according to Sternheimer [164].
- Special transport algorithm, based on Molière's theory of multiple Coulomb scattering improved by Bethe [22, 114, 115], with account of several correlations:
  - between lateral and longitudinal displacement and the deflection angle
  - between projected angles
  - between projected step length and total deflection
- Accurate treatment of boundaries and curved trajectories in magnetic and electric fields
- Automatic control of the step
- Path length correction
- Spin-relativistic effects at the level of the second Born approximation [64].
- Nuclear size effects (scattering suppression) on option (simple nuclear charge form factors are implemented, more sophisticated ones can be supplied by the user).
- Correction for cross-section variation with energy over the step.
- Bremsstrahlung and electron pair production at high energy by heavy charged particles, treated as a continuous energy loss and deposition or as discrete processes depending on user choice.
- Muon photonuclear interactions, with or without transport of the produced secondaries.

1.2.1.5 Low-energy neutrons

For neutrons with energy lower than 20 MeV, FLUKA uses its own neutron cross-section library (P5 Legendre angular expansion, 72 neutron energy groups), containing more than 140 different materials, selected for their interest in physics, dosimetry and accelerator engineering and derived from the most recently evaluated data:

- ENEA multigroup P5 cross sections [42].
- Gamma-ray generation and different temperatures available.
- Doppler broadening for temperatures above 0 K.

Transport:

- Standard multigroup transport with photon and fission neutron generation.
- Detailed kinematics of elastic scattering on hydrogen nuclei
- Transport of proton recoils and protons from N(n,p) reaction.
- Capture photons generated according to the multigroup treatment, but transported with the more accurate EMF package which performs continuous transport in energy and allows for secondary electron generation.

For nuclei other than hydrogen, kerma factors are used to calculate energy deposition (including from low-energy fission). For details about the available materials, group structure etc., see Chap. 10.

1.2.1.6 Electrons

- FLUKA uses an original transport algorithm for charged particles [64], including a complete multiple Coulomb scattering treatment giving the correct lateral displacement even near a boundary (see hadron and muon transport above).
- The variations with energy of the discrete event cross-sections and of the continuous energy loss in each transport step are taken into account exactly.
- Differences between positrons and electrons are taken into account concerning both stopping power and bremsstrahlung [91].
- The bremsstrahlung differential cross-sections of Seltzer and Berger [157, 158] have been extended to include the finite value at "tip" energy, and the angular distribution of bremsstrahlung photons is sampled accurately.
- The Landau-Pomeranchuk-Migdal suppression effect [96, 97, 103, 104] and the Ter-Mikaelyan polarisation effect in the soft part of the bremsstrahlung spectrum [167] are also implemented.
- Electrohadron production (only above ρ mass energy 770 MeV) via virtual photon spectrum and Vector Meson Dominance Model [109]. *(The treatment of the latter effect has not been checked with the latest versions, however).*
- Positron annihilation in flight and at rest
- Delta-ray production via Bhabha and Møller scattering

Note: the present lowest transport limit for electrons is 1 keV. Although in high-Z materials the Molière multiple scattering model becomes unreliable below 20-30 keV, a single-scattering option is available which allows to obtain satisfactory results in any material also in this low energy range.

The minimum recommended energy for *primary* electrons is about 50 to 100 keV for low-Z materials and 100-200 keV for heavy materials, unless the single scattering algorithm is used. Single scattering transport allows to overcome most of the limitations at low energy for the heaviest materials at the price of some increase in CPU time.

1.2.1.7 Photons

- Pair production with actual angular distribution of electrons and positrons.
- Compton effect with account for atomic bonds through use of inelastic Hartree-Fock form factors.
- Photoelectric effect with actual photoelectron angular distribution [155], detailed interaction on six K and L single sub-shells, optional emission of fluorescence photons and approximate treatment of Auger electrons.
- Rayleigh scattering
- Photon polarisation taken into account for Compton, Rayleigh and Photoelectric effects.
- Photohadron production:
  - Vector Meson Dominance Model (Ranft [142]), modified and improved (Ferrari-Sala) using PEANUT below 770 MeV [55].
  - Quasideuteron interactions
  - Giant Dipole Resonance

Note: the present lowest transport limit for photons is 1 keV. However, fluorescence emission may be underestimated at energies lower than the K-edge in high-Z materials, because of lack of Coster-Kronig effect.
The minimum recommended energy for *primary* photons is about 5 to 10 keV.

1.2.1.8 Optical photons

- Generation and transport (on user's request) of Cherenkov, Scintillation and Transition Radiation.
- Transport of light of given wavelength in materials with user-defined optical properties.

1.2.1.9 Neutrinos

- Electron and muon (anti)neutrinos are produced and tracked on option, without interactions.
- Neutrino interactions however are implemented, but independently from tracking.

1.2.2 Geometry

A part of the code where efficiency, accuracy, consistency and flexibility have combined giving very effective results is the new FLUKA geometry. Derived from the Combinatorial Geometry package, it has been entirely rewritten. A completely new, fast tracking strategy has been developed, with special attention to charged particle transport, especially in magnetic fields. New bodies have been introduced, resulting in increased rounding accuracy, speed and even easier input preparation.

- Combinatorial Geometry (CG) from MORSE [44], with additional bodies (infinite circular and elliptical cylinder parallel to X,Y,Z axis, generic plane, planes perpendicular to the axes).
- Possibility to use body and region names instead of numbers.
- Distance to nearest boundary taken into account for improved performance.
- Accurate treatment of boundary crossing with multiple scattering and magnetic or electric fields.
- The maximum number of regions (without recompiling the code) is 10000.
- The tracking strategy has been substantially changed with respect to the original CG package. Speed has been improved and interplay with charged particle transport (multiple scattering, magnetic and electric field transport) has been properly set.
- A limited repetition capability (lattice capability) is available. This allows to avoid describing repetitive structures in all details. Only one single module has to be described and then can be repeated as many times as needed. This repetition does not occur at input stage but is hard-wired into the geometry package, namely repeated regions are not set up in memory, but the given symmetry is exploited at tracking time using the minimum amount of bodies/regions required. This allows in principle to describe geometries with even tens of thousands regions (e.g., spaghetti calorimeters) with a reasonable number of region and body definitions.
- Voxel geometry is available on option, completely integrated into CG.

Special options:

- Geometry debugger.
- Plotting of selected sections of the geometry, based on the Ispra PLOTGEOM program.
- Pseudoparticle `RAY` to scan the geometry in a given direction.

1.2.3 Transport

- Condensed history tracking for charged particles, with single scattering option.
- Time cut-off.
- Legendre angular expansion for low-energy neutron scattering.
- Transport of charged particles in magnetic and electric fields.

Transport limits:

	Secondary particles	**Primary particles**
charged hadrons	1 keV–20 TeV (*)	100 keV–20 TeV (*)
neutrons	thermal–20 TeV(*)	thermal–20 TeV (*)
antineutrons	1 keV–20 TeV (*)	10 MeV–20 TeV (*)
muons	1 keV–1000 TeV	100 keV–1000 TeV
electrons	1 keV–1000 TeV	70 keV–1000 TeV (low-Z materials) 150 keV–1000 TeV (high-Z materials)
photons	1 keV–1000 TeV	7 keV–1000 TeV
heavy ions	10 MeV/n–10000 TeV/n	100 MeV/n–10000 TeV/n

(*) 10 PeV with the DPMJET interface

1.2.4 Biasing

- Leading particle biasing for electrons and photons: region dependent, below user-defined energy threshold and for selected physical effects.
- Russian Roulette and splitting at boundary crossing based on region relative importance.
- Region-dependent multiplicity tuning in high energy nuclear interactions.
- Region-dependent biased downscattering and non-analogue absorption of low-energy neutrons.
- Biased decay length for increased daughter production.
- Biased inelastic nuclear interaction length.
- Biased interaction lengths for electron and photon electromagnetic interactions
- Biased angular distribution of decay secondary particles.
- Region-dependent weight window in three energy ranges (and energy group dependent for low-energy neutrons).

1.2.5 Optimisation

- Optimisation of the step length, user-defined or automatic, by material and/or by region.

1.2.6 Scoring

- Star density by producing particle and region.
- Energy density by region, total or from electrons/photons only.
- Star, energy and momentum transfer density in a geometry-independent binning structure (cartesian or cylindrical), averaged over the run or event by event.
- Energy deposition weighted by a quenching factor (Birks law).
- Step size independent of bin size.
- Time window.
- Coincidences and anticoincidences.
- Fluence and current scoring as a function of energy and angle, via boundary-crossing, collision and track-length estimators coincident with regions or region boundaries.
- Track-length fluence in a binning structure (cartesian or cylindrical) independent of geometry.
- Particle yield from a target or differential cross-section with respect to several different kinematic variables.
- Residual nuclei.
- Fission density.
- Momentum transfer density.
- Neutron balance.
- No limit to the number of estimators and binnings within the total memory available (but a maximum number must be fixed at compilation time).

– Energy deposition can be scored on option disregarding the particle weights (useful for studying computer performance, etc.)
– All quantities from radioactive decay of residual nuclei can be scored according to user-defined irradiation and cooling time profiles

1.2.7 Code structure, technical aspects

– The whole program, including the numerical constants, is coded in double precision (at least the versions for 32-bit word machines). The only exceptions are the low-energy neutron cross-sections, which are stored in single precision to save space.
– Consistent use of the latest recommended set of physical constant values [118].
– Dynamic memory allocation is implemented as far as possible.
– Extensive use of `INCLUDE` and of constant parameterisation
– 64-bit random number generator [102]

1.2.8 Main differences between FLUKA and earlier codes with same name

The history of FLUKA, spanning more than 40 years, is narrated in detail in Chap. 16. It is possible to distinguish three different generation of "FLUKA" codes along the years, which can be roughly identified as the FLUKA of the '70s (main authors J. Ranft and J. Routti), the FLUKA of the '80s (P. Aarnio, A. Fassò, H.-J. Möhring, J. Ranft, G.R. Stevenson), and the FLUKA of today (A. Fassò, A. Ferrari, J. Ranft and P.R. Sala). These codes stem from the same root and of course every new "generation" originated from the previous one. However, each new "generation" represented not only an improvement of the existing program, but rather a quantum jump in the code physics, design and goals. The same name "FLUKA" has been preserved as a reminder of this historical development — mainly as a homage to J. Ranft who has been involved in it as an author and mentor from the beginning until the present days — but the present code is completely different from the versions which were released before 1990, and in particular from the last one of the second generation, FLUKA87 [3, 4].

Major changes and additions have affected the physical models used, the code structure, the tracking strategy and scoring. Important additions, such as a wider range of biasing possibilities and some specialised tools for calorimeter simulation, have extended the field of its possible applications.

An exhaustive description of all these changes and new features along the years is reported in Chap. 16. However, the best gauge of the program evolution is probably the widening of the application fields, and the boost of its recognition and diffusion all over the world.

1.2.8.1 Applications

While FLUKA86-87 was essentially a specialised program to calculate shielding of high energy proton accelerators, the present version can be regarded as a general purpose tool for an extended range of applications. In addition to traditional target design and shielding, applications are now spanning from calorimetry to prediction of activation, radiation damage, isotope transmutation, dosimetry and detector studies.

Prediction of radiation damage has always been a traditional field of application of FLUKA, restricted however in earlier versions to hadron damage to accelerator components. The new capability to deal with the low-energy neutron component of the cascade has extended the field of interest to include electronics and other sensitive detector parts. In addition, radiation damage calculations and shielding design are not limited to proton accelerators any longer, but include electron accelerators of any energy, photon factories, and any kind of radiation source, be it artificial or natural.

The present version of FLUKA has been used successfully in such diverse domains as background studies for underground detectors, cosmic ray physics, shielding of synchrotron radiation hutches, calculation of dose received by aircraft crews, evaluation of organ dose in a phantom due to external radiation, detector design for radiation protection as well as for high energy physics, electron and proton radiotherapy, nuclear

transmutation, neutrino physics, shielding of free-electron lasers, calculation of tritium production at electron accelerators, energy amplifiers, maze design for medical accelerators, etc.

The recent addition of the simulation of heavy ion interactions allows also for applications to hadrotherapy.

Chapter 2

A FLUKA beginner's guide

2.1 Introduction

This Chapter is intended to provide a minimal set of instructions to install and run FLUKA for a beginner user, going through the different steps required to run a simple example. References to the relevant chapters of the manual are given to help the user in finding more detailed information on the different topics. After describing the installation procedure, instructions are given on how to build the input file for a simple case. Instructions for running and accessing the results are also reported. As a further step, the user is addressed to Chapter 12 to learn how to tailor FLUKA to specific needs by means of user routines.

introduction.tex

2.2 Installing FLUKA

A full description of the installation procedure is given in Chapter 3. The FLUKA package for LINUX and UNIX platforms is distributed as a tar file which can be downloaded from the FLUKA Website http://www.fluka.org or other authorised mirror site.

The user must prepare a directory where the tar file has to be expanded. Let us suppose that the gzipped tar file, called `flukalinux.tar.gz`, has been downloaded in a given directory and that it must be expanded in a new subdirectory called `fluka` (but any other name is valid). A possible way to proceed is the following:

```
mkdir fluka
cd fluka
tar zxvf ../flukalinux.tar.gz
```

An alternative way to expand the tar file is:

```
gunzip -dc ../flukalinux.tar.gz | tar xvf
```

At this stage, the user must define an environmental variable `FLUPRO` pointing to the directory where the distribution tar file has been opened. This directory is made available as follows:

Bash shell: use the `export` command:

```
...
cd fluka
export FLUPRO=$PWD
```

C or tc shell: use the `setenv` command:

```
...
cd fluka
setenv FLUPRO $PWD
```

Of course the definition of `FLUPRO` can be placed once for ever in the login script.

The FLUKA libraries and most data files will be located in `$FLUPRO`, the `INCLUDE` files in `$FLUPRO/flukapro/`, the default user routines in `$FLUPRO/usermvax/`, compilation and linking scripts (as well as several postprocessing programs to analyse user scores) in `$FLUPRO/flutil/`.
See details in Chapter 3. The user must produce the default FLUKA executable, to be located in the `$FLUPRO` directory, by means of the `$FLUPRO/flutil/lfluka` script:

```
cd $FLUPRO
$FLUPRO/flutil/lfluka -m fluka
```

The default executable is called `flukahp`.

2.3 Building a FLUKA input

2.3.1 Generalities about FLUKA input

FLUKA reads user input from an ASCII "standard input" file with extension `.inp`. The general characteristics and rules of FLUKA input are described in Chapter 6. The input consists of a variable number of "commands" (called also "options"), each consisting of one or more "lines" (called also "cards" for historical reasons).

Apart from FLUKA commands, the input file may contain also the description of the geometry of the simulated set-up. Also this description is provided by means of specific geometry "command cards" in a special format described in Chapter 8.

The geometry description can, on request, be kept in a separate ASCII file: this feature is especially useful when the same geometry is used in several different inputs, not only to save space but because modifications can be made in one single place.

The typical structure of a FLUKA input file is the following:

- Titles and comments for documentation purposes (optional, but recommended)
- Description of the problem geometry (solid bodies and surfaces, combined to partition space into regions) (mandatory)
- Definition of the materials (mandatory unless pre-defined materials are used)
- Material assignments (correspondence material–region, mandatory)
- Definition of the particle source (mandatory)
- Definition of the requested "detectors". Each of these is a phase space domain (region of space, particle direction and energy) where the user wants to calculate the expectation value of a physical quantity such as dose, fluence, etc. Various kinds of detectors are available, corresponding to different quantities and to different algorithms used to estimate them ("estimators"). Detectors are optional, but one at least is expected, at least in the production phase
- Definition of biasing schemes (optional)
- Definition of problem settings such as energy cut-offs, step size, physical effects not simulated by default, particles not to be transported, etc. (optional)
- Initialisation of the random number sequence (mandatory if an estimation of the statistical error is desired)
- Starting signal and number of requested histories (mandatory)

In addition, special commands are available in FLUKA for more advanced problems involving magnetic fields, time-dependent calculations, writing of history files (so-called "collision tapes"), transport of optical photons, event-by-event scoring, calling user-written routines, etc. These options are expected to be requested only by users having some previous experience with the more common commands: therefore they will be mostly ignored in this beginner's guide.

Let's first recall the general structure of the FLUKA command lines (cards). The geometry commands will be reviewed later. Each card contains:

- one keyword,
- six floating point values (called WHATs),
- one character string (called SDUM)

Not necessarily all WHATs and SDUMs are used. In some cases, a command line can be followed by a line of text (for instance a filename path or a title). Any line having an asterisk (`*`) in the first position is treated as a comment. All lines (commands, text strings and comments) are echoed on the standard output (the file with extension `.out`). In case of problems, it is a good idea to check how every line has been printed in the standard output. Often, output reveals typing or format errors by showing how the program has misinterpreted them.

In addition to the simple echo, an "interpreted" feedback to all commands is provided in the next section of the standard output. Checking this part of the output is also very useful, because it helps making sure that the user's intentions have been expressed correctly and understood by the code. See Chapter 9 on FLUKA output for a detailed description.

If a line contains an exclamation mark (`!`) all following characters are replaced by blanks. This feature can be used for short in-line comments which will not be echoed in output.

The order of input commands is generally free, with only a few exceptions reported in Chapter 6. Therefore, the order suggested in the following should not be considered as mandatory, but only one of the possible ways to write FLUKA input.

2.3.2 Input alignment

Be careful to properly align keywords and numbers. Format is not free, unless a command is issued at the beginning of input: see option `GLOBAL` in section 7.33 or `FREE` in section 7.30). Even in the free format for the input file, the part of the input describing the geometry can still be written in fixed format (which is different from the general FLUKA input format, see Chapter 8). There is the possibility of having free format also for the geometry part: this can also be activated using the `GLOBAL` command (see 7.33).

In fixed format, in order to ensure proper alignment, a comment line showing a scale can be used anywhere in the input file, for instance:

```
*...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
```

In numerical fields, blanks are treated as zero. Therefore, numbers written with a decimal period and without an exponent (e.g., `1.2345`) can be placed anywhere inside their respective format fields. For instance, the following two input lines are equally acceptable:

```
BEAM      200.      0.2       1.5       1.2       0.7       1.0       PROTON
*...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
BEAM            200.       0.2       1.5       1.2       0.7       1.0PROTON
```

If a number is written in exponential notation (e.g., `2.3E5`) or in integer form (without decimal point), it must be aligned to the right of the field. Depending on the platform and the compiler, sometimes the number is correctly interpreted even if the alignment rule is not respected. **However this is not guaranteed and the right alignment rule should always be followed**. For instance in:

```
*...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
BEAM          2.E2        0.2       1.5       1.2       0.7       1.  PROTON
```

the first value might be interpreted as `2.E200`. Another case is the following:

```
*...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
BEAM      200.      0.2       1.5       1.2       0.7       1         PROTON
```

here the last numerical field would be interpreted as `1000000000`. To avoid mistakes due to this kind of input errors, the present FLUKA versions now recognises such potential problems and the program is forced to stop. At the same time a message is printed on the standard output file, as shown here for the above example:

```
  *** The 6th field  1         of the following input card ***
 BEAM      200.     0.2       1.5       1.2       0.7       1         PROTON
  *** does not contain a valid formatted fortran real number!!! ***
  *** It is ambiguous and it could be read differently on different compilers ***
  *** depending how strictly the blank=0 formatted input rule is implemented  ***
```

Keywords (character strings such as `BEAM` and `PROTON`) *must be aligned to the left of their field* and must be in upper case. An exception is the continuation character "`&`" used in some commands, which can be placed anywhere within its 10-characters field.

2.3.3 A simple example

Let us now consider a simple starting application. We want to calculate the charged pion fluence produced by a monochromatic proton beam of momentum 50 GeV/c impinging on a 5 cm thick beryllium target of simple shape: a small parallelepiped (20 cm × 20 cm × 5 cm). A further simplification will be made for the purpose of this example, neglecting all the surrounding environment and substituting it with ideal vacuum.

We will guide the reader through the different parts of a possible input file suited for this application. The information which follows is meant to serve as a guide, but does not cover all the important points. It is recommended that for each option card selected, the user read carefully the relevant manual entry, and especially the explanatory notes.

2.4 The title

Typically, an input file begins with a TITLE card (p. 207) followed by one line of text describing the problem, or identifying that particular run, or providing some kind of generic information. In our case, for example

```
*...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
TITLE
Charged pion fluence inside and around a proton-irradiated Be target
```

Further information can be provided by comment cards, but the title, because it is printed at the top of the standard output, is a useful reminder of what the corresponding file is about. In addition, the title is printed in all separate output files (error file, estimator files etc.) together with the date and time of the FLUKA run, allowing to keep track of their origin.

Commands such as GLOBAL and DEFAULTS, if needed, must immediately follow. Since this is intended as a beginner's guide, these can be ignored here (for most common problems the defaults provided are sufficient). Let us recall that without specifying a DEFAULT value, the NEW–DEFA set of FLUKA parameters is loaded (see section 7.13).

2.5 Definition of the primary particles

All "events" or "histories" are initiated by primary particles, which in the simplest case are monoenergetic, monodirectional and start from a single point in space (pencil beam). The particle energy (or momentum) is defined by option BEAM (p. 64), and its starting position and direction by option BEAMPOS (p. 69). These two cards can be used also to define particle beams having a simple angular or momentum distribution (Gaussian or rectangular), or a simple transverse profile (Gaussian, rectangular or annular). Isotropic or semi-isotropic angular emission can be described as a special case of an angular rectangular distribution.

For particle sources with more complex distributions in energy, space and direction, the user must write, compile and link a special routine (SOURCE), following the instructions given in 12.2.19, and input a card SOURCE (p. 197).

A summary of the input data concerning primary particles is printed in the standard output under the title "Beam properties".

The beam definition for our example can be the following (monochromatic, monodirectional proton beam of momentum 50 GeV/c):

```
*...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
BEAM         50.E+00                                                 PROTON
```

In our example, the beam starting point is given by:

```
*...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
BEAMPOS          0.0       0.0     -50.0
```

In the cartesian geometry used by FLUKA, the previous card means that the beam is injected at x, y, z coordinates: 0, 0, -50 cm and is directed along the positive z axis. Of course, the choice of the point of

injection, the direction, etc., must be coherent with the geometrical description of the set-up, as discussed in the following section.

2.6 The geometry

The input for the Combinatorial Geometry (bodies, regions and optional region volumes, see respectively 8.2.4, 8.2.6, 8.2.8) must be immediately preceded by a GEOBEGIN card (see 7.31 and 8.2.1) and immediately followed by a GEOEND card (see 7.32 and 8.2.10). These two cards follow the normal FLUKA format. It is recalled that the format for the geometry has its own special rules, described in Chapter 8.

Comment lines in the geometry input have an asterisk in first position as in the rest of FLUKA input (but on-line comments are not allowed). Body numerical data can be written in two different formats, a "short" one (field length 10) and a "long" one (field length 22). The latter one is to be preferred when higher precision is needed, for instance when using bodies such as truncated cones, cylinders or planes not aligned with axes. It must be realised that using too few decimals can cause geometry errors when bodies are combined into regions (portions of space not defined or doubly defined).

The whole geometry must be surrounded by a region of "blackhole" limited by a closed body (generally an RPP parallelepiped). It is often a good idea to make this body much larger than the minimum required size: this makes easier to introduce possible future extensions. In some cases, as in our basic example, it is also useful to surround the actual geometry by a region of ideal vacuum, and to have the blackhole region surrounding the vacuum. This can be useful, for instance, in order to start the trajectory of the primary particles outside the physical geometry (a particle may not be started on a boundary).

Both the body input section and the region input section must be ended with an END card (see 8.2.5 and 8.2.7). Optionally, region volumes can be input between the region END card and the GEOEND card (this option can be requested by setting a special flag in the Geometry title card, see 8.2.2). The only effect of specifying region volumes is to normalise per cm^3 the quantities calculated via the SCORE option (see below): for other estimators requiring volume normalisation the volume is input as part of the detector definition (USRTRACK, USRCOLL, USRYIELD), or is calculated directly by the program (USRBIN).

The GEOEND card indicates the end of the geometry description, but can also be used to invoke the geometry debugger.

The geometry output is an expanded echo of the corresponding input, containing information also on memory allocation and on the structure of composite regions made of several sub-regions by means of the OR operator.

A possible realisation of the geometry set up for our basic example can be seen in Fig. 2.1.

Only four bodies are used here: an RPP body (Rectangular Parallelepiped, body no. 3, see 8.2.4.1) to define a volume which will be the Be target region, inside another larger RPP body (no. 2), which will be filled with ideal vacuum, and in turn is contained inside another larger RPP body (no. 1) to define the blackhole region. The fourth body, an XYP half-space (defined by a plane perpendicular to the z axis, see 8.2.4.10), will be used to divide the target into 2 different regions: the upstream half will be defined as the portion of body 3 contained inside the half-space, and the downstream half as the portion outside it. Therefore, region "3" (the upstream half of the target) is the part of body no. 3 which is also inside body 4, while region "4" (downstream half of the target) is the part of body no. 3 which is not inside body 4. Region "2" (the vacuum around the target) is defined as the inside of body no. 2 from which body no. 3 is subtracted. Region "1" is simply the interior of body no. 1 from which body no. 2 is subtracted.

The beam starting point has been chosen so that it is in the vacuum, outside the target region. The geometry part of the input file can then be written as:

```
*...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
GEOBEGIN                                                              COMBINAT
                         A simple Be target inside vacuum
  RPP    1-5000000.0+5000000.0-5000000.0+5000000.0-5000000.0+5000000.0
  RPP    2-1000000.0+1000000.0-1000000.0+1000000.0    -100.0+1000000.0
```

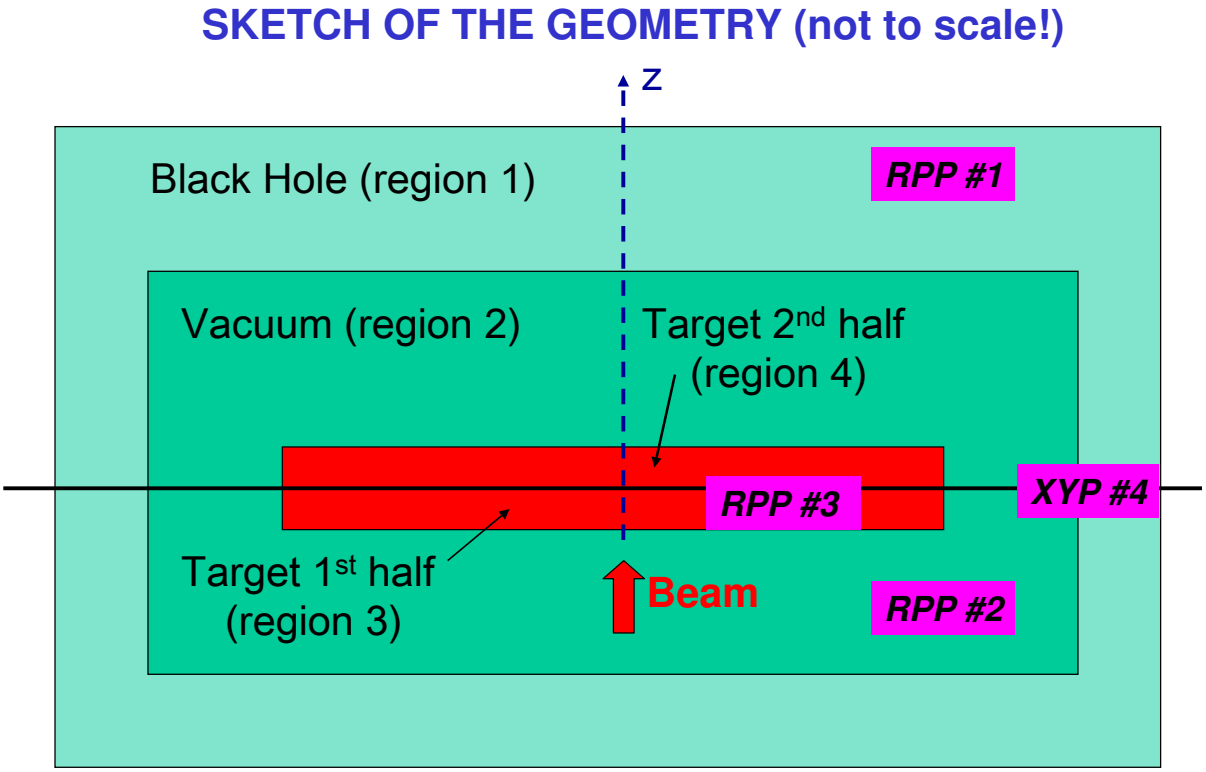

Fig. 2.1: Sketch of the geometry set–up of the example

```
  RPP    3     -10.0     +10.0     -10.0     +10.0       0.0      +5.0
* plane to separate the upstream and downstream part of the target
  XYP    4       2.5
  END
* black hole
  BH1    5     +1      -2
* vacuum around
  VA2    5     +2      -3
* Be target 1st half
  BE3    5     +3      +4
* Be target 2nd half
  BE4    5     +3      -4
  END
GEOEND
```

2.7 Materials

Each geometry region is supposed to be filled with a homogeneous material, or with vacuum, or with "blackhole". The latter is a fictitious material used to terminate particle trajectories: any particle is discarded when reaching a blackhole boundary. Materials can be simple elements or compounds, where an element can have either natural composition or consist of a single nuclide, and compound indicates a chemical compound or a mixture or an alloy (or an isotopic mixture) of known composition.

An element can be either predefined (see list in Table 5.3 on page 44) or defined by a MATERIAL card (p. 142) giving its atomic number, atomic mass, density, name and a material identification number > 2. The material number can be chosen by the user, with the restriction that all lower numbers must also be defined (but not necessarily used).

Number 1 is reserved for blackhole and 2 for ideal vacuum. There are 25 predefined materials; but each of the numbers from 3 to 25 can be redefined freely, overriding the default definition.

Materials can also be defined with higher index numbers, provided no gaps are left in the numbering sequence. For instance a material cannot be defined to have number 28 unless also 26 and 27 have been defined.

A compound is defined by a MATERIAL card plus as many COMPOUND cards (p. 78) as needed to describe its composition. The MATERIAL card used to define a compound carries only the compound name, density and material number (atomic number and atomic mass having no meaning in this case).

Materials predefined or defined in the standard input are referred to as "FLUKA materials", to distinguish them from materials available in the low-energy neutron cross-section library (called "low-energy cross section materials").

When transport of low-energy neutrons (E $<$ 19.6 MeV) is requested (explicitly or by the chosen defaults), a correspondence is needed between each elemental (i.e., not compound) "FLUKA material" and one of the "low-energy cross-section materials" available in the FLUKA low-energy neutron library. The default correspondence is set by the name: and if more than one material with that name exist in the neutron library, the first in the list with that name (see 10.3, p. 295) is assumed by default. The default can be changed using command LOW–MAT.

In the case of our example, only Beryllium is necessary, apart from blackhole and vacuum. In principle, since Beryllium is one of the pre-defined FLUKA materials, this part could even be omitted. However, for pedagogical reasons the following card is proposed, where index 5 is assigned to the target material:

```
*...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
MATERIAL         4.0    9.0122     1.848       5.0                    BERYLLIU
```

Notice that the chosen name is BERYLLIU and not BERYLLIUM, in order to match the name in the list of "MORSE materials" for low-energy neutrons (see Table 10.3), where material names have a maximum length of 8 characters.

The standard output concerning materials is very extended. First a list of multiple scattering parameters (printed by subroutine MULMIX) is reported for each material requested. This is mostly of scarce interest for the normal user, except for a table giving for each requested material the proportion of components, both by number of atoms (normalised to 1) and by weight (normalised to material density). The same information is repeated later on in another table entitled `Material compositions`.

If low-energy neutron transport has been requested (explicitly or by a chosen default), the following section reports the relevant material information: in particular, a table entitled "`Fluka to low en. xsec material correspondence`" specifying which material in the neutron cross-section library has been mapped to each input material. Note that a much more detailed cross-section information can be obtained by setting a printing flag (WHAT(4)) in the LOW–NEUT command.

The Table `Material compositions` contains information about the requested materials, those predefined by default and the elements used to define compounds. In addition to effective atomic and mass number, density and composition, the table shows the value of some typical quantities: inelastic and elastic scattering length for the beam particles (not valid for electron and photon beams), radiation length and inelastic scattering length for 19.6 MeV neutrons.

The next table contains material data related to stopping power (average excitation energy, density effect parameters, and pressure in the case of gases) plus information about the implementation of various physical effects and the corresponding thresholds and tabulations.

The last material table is printed just before the beginning of the history calculations, and concerns the `Correspondence of regions and EMF-FLUKA material numbers and names`.

2.7.1 Assigning materials to regions

A material must be associated to each of the geometry regions, except to those defined as blackhole. This is done in a very straightforward way by command ASSIGNMAt. Assigning explicitly blackhole to a region is

allowed, but is not necessary (except for region 2) because a region is blackhole by default unless another material has been associated to it. (Region 2, if not assigned a material, is COPPER by default).

The table entitled `Regions: materials and fields`, in the standard output, can be consulted to check that material assignment has been done as desired.

For the present example the assignment could be:

```
*...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
*  Be target, 1st and 2nd half
ASSIGNMAT        5.0       3.0       4.0
*  External Black Hole
ASSIGNMAT        1.0       1.0
*  Vacuum
ASSIGNMAT        2.0       2.0
```

2.8 Production Thresholds

The implicit NEW–DEFA default setting, adopted in the example, sets, among other things, the production and transport threshold of heavy particles to 10 MeV. Production thresholds for $e^+e^-$and photons must be explicitly set using the EMFCUT command for all materials in the problem, as described in detail on p. 102. Let us choose also in this case a 10 MeV threshold for the single material of the example (previously marked with material index 5). Following the instructions about the EMFCUT option, the card can be written as:

```
*...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
EMFCUT        -0.010     0.010       1.0       5.0                     PROD-CUT
```

where the first numerical field is the threshold for e^+e^-(the minus sign meaning kinetic energy) and the second is for photons. The material number is given in the fourth numerical field. For details on all other parameters, and for other possibilities (for example how to introduce a transport cut-off different from production threshold) the user must accurately consult the Notes in Sec. 7.21.

Production and transport threshold for all other particles can be overwritten using the PART–THR command (p. 172).

2.9 Estimators and Detectors

Even though, for setting-up purposes, it is conceivable that no detector be requested in a preliminary run, in most cases FLUKA is used to predict the expectation value of one or more quantities, as determined by the radiation field and by the material geometry described by the user: for such a task several different *estimators* are available. The quantities which are most commonly scored are dose and fluence, but others are available. Dose equivalent is generally calculated from differential fluence using conversion coefficients.

The simplest estimator available to the user is a historical vestige, survived from the "ancient" FLUKA (pre-1988) where the only possible output quantities were energy deposition and star density in regions. It is invoked by option SCORE (p. 195), requesting evaluation of one to four different quantities. These can be different forms of energy density (proportional to dose), or of star density (approximately proportional to fluence of selected high-energy hadrons).

In this case, the detectors are pre-determined: the selected quantities are reported for each region of the geometry. The corresponding results, printed in the main output immediately after the last history has been completed, are presented in 6 columns as follows:

```
region    region     first       second      third       fourth
number    volume    quantity    quantity    quantity    quantity
  1       ......    ........    ........    ........    ........
  2       ......    ........    ........    ........    ........
```

etc.
on a line for each geometry region. The region volumes (in cm^3) have the value `1.0`, or values optionally

supplied by the user at the end of the geometry description (see 8.2.2). All other columns are normalised per region volume and per primary particle.

The input could be as follows:

```
*...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
* score in each region energy deposition and stars produced by primaries
SCORE          208.0     210.0
```

Other estimators are more flexible: the corresponding detectors can be requested practically in any number, can be written as unformatted or text files, and in most cases can provide differential distributions with respect to one or more variables. On the other hand, their output is presented in a very compact array form and must generally be post-processed by the user. For this purpose several utility programs are available: but output in text format can even be exported to a spreadsheet for post-processing.

USRBDX (see p. 211) is the command required to define a detector for the boundary-crossing estimator. It calculates fluence or current, mono- or bi-directional, differential in energy and/or angle on any boundary between two selected regions. The area normalisation needed to obtain a current in particles per cm^2 is performed using an area value input by the user: if none is given, the area is assumed to be $=$ `1.0` cm^2 and the option amounts simply to counting the total number of particles crossing the boundary. Similarly if fluence is scored, but in this case each particle is weighted with the secant of the angle of the trajectory of each particle with respect to the normal to the boundary surface at the crossing point.

This is one of the estimators proposed for our example. We will request two detectors, one to estimate fluence and one for current, of particles crossing the boundary which separates the upstream and the downstream half of the target.

The following group of card can be inserted:

```
*...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
* Boundary crossing fluence in the middle of the target (log intervals, one-way)
USRBDX          99.0     209.0     -47.0       3.0       4.0     400. piFluenUD
USRBDX         +50.0               +50.0                 0.0      10.0 &
*
* Boundary crossing current in the middle of the target (log intervals, one-way)
USRBDX          -1.0     209.0     -47.0       3.0       4.0     400. piCurrUD
USRBDX        +50.00               +50.0                 0.0      10.0 &
```

According to the instructions reported in Sec. 7.76, it can be seen that the combined fluence of π^+ and π^- is requested only when particles exit region "3" (the upstream half of the target) to enter into region "4" (the downstream half). There is no interest in the reverse, therefore "one-way scoring" is selected. The scoring of the first detector will be inverse cosine-weighted, in order to define correctly the fluence. Results will be written unformatted on unit 47 for both quantities (so there will be two "Detectors" on the same output unit, but this is not mandatory). The energy distribution is going to be binned in 50 logarithmic intervals, from 0.001 GeV (the default minimum) up to 50 GeV. The angular distribution will be binned into 10 linear solid angle intervals from 0. to 2π (having chosen the one-way estimator). The results will be normalised dividing by the area of the boundary (separation surface between the two regions, in this case the transverse section of the target), and will provide a double-differential fluence or current averaged over that surface (in cm^{-2} GeV^{-1} sr^{-1}).

Other fluence scoring options, based respectively on a track-length and on a collision estimator, are USRTRACK (see p. 231) and USRCOLL (p. 227) which request the estimation of volume-averaged fluence (differential in energy) for any type of particle or family of particles in any selected region. The volume normalisation needed to obtain the fluence as track-length density or collision density is performed using a volume value input by the user: if none is given, the volume is assumed to be $=$ `1.0` cm^3 and the result will be respectively the total track-length in that region, or the total number of collisions (weighted with the mean free path at each collision point).

Note that if additional normalisation factors are desired (e.g., beam power), this can be achieved by

giving in input the "volume" or "area" value multiplied or divided by those factors. Options USRTRACK, USRCOLL and USRBDX can also calculate energy fluence, if the "particle" type is set = 208.0 (energy) or 211.0 (electron and photon energy).

In our example, we are requesting two track-length detectors, to get the average fluence in the upstream half and in the downstream half of the target, respectively.

```
*...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
* Track-length fluence inside the target, Upstream part and Downstream part
* Logarithmic energy intervals
USRTRACK        -1.0     209.0     -48.0       3.0    1000.0      20. piFluenU
USRTRACK        50.0     0.001                                        &
USRTRACK        -1.0     209.0     -49.0       4.0    1000.0      20. piFluenD
USRTRACK        50.0     0.001                                        &
```

The volume input is $20 \times 20 \times 2.5 = 1000$ cm^3. We are requesting an energy spectrum in 20 logarithmic intervals between 0.001 and 50 GeV. In this case, we ask that the corresponding output be printed, unformatted, on two different files.

Option USRBIN provides detailed space distributions of energy deposition, star density or integrated fluence (not energy fluence, unless by writing a special user routine). Using some suitable graphics package, USRBIN output (see p. 218) can be presented in the form of colour maps. Programs for this purpose are available in the `$FLUPRO/flutil` directory (`pawlevbin.f` and various `kumac` files), and on the FLUKA website www.fluka.org.

USRBIN results are normalised to bin volumes calculated automatically by the program (except in the case of region binning and special 3-variable binning which are only seldom used).

The binning structure does not need to coincide with any geometry region. In our example we propose to ask for two Cartesian space distributions, one of charged pion fluence and one of total energy deposited. The first will extend over a volume larger than the target, because fluence can be calculated even in vacuum. Energy deposition, on the other hand, will be limited to the target volume.

```
*...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
* Cartesian binning of the pion fluence inside and around the target
USRBIN          10.0     209.0     -50.0      50.0      50.0      50. piFluBin
USRBIN         -50.0     -50.0     -10.0     100.0     100.0      60.0   &
* Cartesian binning of the deposited energy inside the target
USRBIN          10.0     208.0     -51.0      10.0      10.0       5. Edeposit
USRBIN         -10.0     -10.0       0.0      20.0      20.0       5.0   &
```

Also in this case, the request is for output on two separate files.

Angular yields around a fixed direction of particles exiting a given surface can be calculated using option USRYIELD (see p. 236). The results are double-differential distributions with respect to a pair of variables, one of which is generally energy-like (kinetic energy, momentum, etc.) and the other one angle-like (polar angle, rapidity, Feynman-x, etc.) Distributions in LET (Linear Energy Transfer) can also be requested by this option. An arbitrary normalisation factor can be input.

Another commonly used scoring option is RESNUCLEi (see p. 187), which calculates residual nuclei production in a given region. A normalisation factor (usually the region volume) can be input.

A detailed summary of the requested estimators is printed on standard output. The same information is printed in the same format in estimator ASCII output files, and is available in coded form in estimator unformatted output files.

2.10 Initialisation of the random number sequence

The random number sequence used in a run is initialised by default by the seeds contained in file `random.dat` provided with the code. To calculate the statistical error of the results, it is necessary to perform other

independent runs (at least 4 or 5), each with a different independent initialisation, using the seeds written by the program at the end of each run. The `rfluka` script provided with the code on UNIX and LINUX platforms takes care of this task, provided the following card is issued in the input file:

```
*...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
RANDOMIZE        1.0       0.0
```

The seeds of the random number generator are printed on a special file in hexadecimal form at the end of each group of histories (the size of a group depends on the number of histories requested in the START card).

Instead of getting the seeds from the last run, it is also possible to initialise directly another independent random number sequence by setting the second RANDOMIZE parameter equal to a different number, for instance:

```
*...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
RANDOMIZE        1.0     1198.
```

2.11 Starting signal and number of requested histories

At the end of the input file, a START card (see p. 199) is mandatory in order to actually start the calculation. That card must indicate also the number of particle histories requested. The run, however, may be completed before all the histories have been handled in two cases: if a time limit has been met (on some systems) or if a "stop file" is created by the user (see instructions in Note 2 to option START). The START card is optionally followed by a STOP card. For example, if the user wants to generate 100000 histories, the input file can be closed with the following two cards:

```
*...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
START       100000.0
STOP
```

2.12 The sample input file

In summary, the input file for our basic example (`example.inp`), written in fixed format, is the following:

```
*...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
TITLE
Charged pion fluence inside and around a proton-irradiated Be target
*...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
BEAM         50.E+00                                                 PROTON
BEAMPOS          0.0       0.0     -50.0
*
GEOBEGIN                                                             COMBINAT
                        A simple Be target inside vacuum
  RPP    1-5000000.0+5000000.0-5000000.0+5000000.0-5000000.0+5000000.0
  RPP    2-1000000.0+1000000.0-1000000.0+1000000.0    -100.0+1000000.0
  RPP    3     -10.0     +10.0     -10.0     +10.0       0.0      +5.0
* plane to separate the upstream and downstream part of the target
  XYP    4       2.5
  END
* black hole
  BH1    5     +1     -2
* vacuum around
  VA2    5     +2     -3
* Be target 1st half
  BE3    5     +3     +4
* Be target 2nd half
  BE4    5     +3     -4
  END
```

```
GEOEND
*...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
MATERIAL         4.0    9.0122     1.848       5.0                     BERYLLIU
*...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
*  Be target, 1st and 2nd half
ASSIGNMAT        5.0       3.0       4.0
*  External Black Hole
ASSIGNMAT        1.0       1.0
*  Vacuum
ASSIGNMAT        2.0       2.0
*...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
* e+e- and gamma production threshold set at 10 MeV
EMFCUT        -0.010     0.010       1.0       5.0                     PROD-CUT
* score in each region energy deposition and stars produced by primaries
SCORE          208.0     210.0
* Boundary crossing fluence in the middle of the target (log intervals, one-way)
USRBDX          99.0     209.0     -47.0       3.0       4.0     400. piFluenUD
USRBDX         +50.0               +50.0                 0.0      10.0 &
* Boundary crossing current in the middle of the target (log intervals, one-way)
USRBDX          -1.0     209.0     -47.0       3.0       4.0     400. piCurrUD
USRBDX         +50.0               +50.0                 0.0      10.0 &
* Track-length fluence inside the target, Upstream part and Downstream part
* Logarithmic energy intervals
USRTRACK        -1.0     209.0     -48.0       3.0    1000.0      20. piFluenU
USRTRACK        50.0     0.001                                        &
USRTRACK        -1.0     209.0     -49.0       4.0    1000.0      20. piFluenD
USRTRACK        50.0     0.001                                        &
* Cartesian binning of the pion fluence inside and around the target
USRBIN          10.0     209.0     -50.0      50.0      50.0      50. piFluBin
USRBIN         -50.0     -50.0     -10.0     100.0     100.0      60.0   &
* Cartesian binning of the deposited energy inside the target
USRBIN          10.0     208.0     -51.0      10.0      10.0       5. Edeposit
USRBIN         -10.0     -10.0       0.0      20.0      20.0       5.0   &
*...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
RANDOMIZE        1.0
*...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
START       100000.0
STOP
```

The same input file, using the free format option for the FLUKA commands, but not for the geometry, can instead be written as follows:

```
*...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
TITLE
Charged pion fluence inside and around a proton-irradiated Be target
*...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
GLOBAL                                          2.0
BEAM  50.E+00 0. 0. 0. 0. 0. PROTON
BEAMPOS  0. 0. -50.0 0. 0. 0. ' '
*...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
GEOBEGIN                                                          COMBINAT
                        A simple Be target inside vacuum
  RPP    1-5000000.0+5000000.0-5000000.0+5000000.0-5000000.0+5000000.0
  RPP    2-1000000.0+1000000.0-1000000.0+1000000.0    -100.0+1000000.0
  RPP    3     -10.0     +10.0     -10.0     +10.0       0.0      +5.0
  XYP    4       2.5
  END
* black hole
  BH1    5     +1     -2
* vacuum around
  VA2    5     +2     -3
```

```
* Be target 1st half
  BE3    5     +3     +4
* Be target 2nd half
  BE4    5     +3     -4
  END
GEOEND
MATERIAL  4.0 9.0122 1.848 5.0 0. 0. BERYLLIU
*  Be target, 1st and 2nd half
ASSIGNMAT 5.0 3.0 4.0 0. 0. 0.
*  External Black Hole
ASSIGNMAT 1.0 1.0 0. 0. 0. 0.
*  Vacuum
ASSIGNMAT 2.0 2.0 0. 0. 0. 0.
* e+e- and gamma production threshold set at 10 MeV
EMFCUT -0.010 0.010 1.0 5.0, , , PROD-CUT
* score in each region energy deposition and stars produced by primaries
SCORE 208.0 210. 0. 0. 0. 0.
* Boundary crossing fluence in the middle of the target (log intervals, one-way)
USRBDX  99.0 +209.0 -47.0 3.0 4.0 +400.0 piFluenUD
USRBDX  +50.00 0. +50.0 0. 0. 10.0 &
* Boundary crossing current in the middle of the target (log intervals, one-way)
USRBDX  -1.0 +209.0 -47.0 3.0 4.0 +400.0 piCurrUD
USRBDX  +50.00 0. +50.0 0. 0. 10.0 &
* Track-length fluence inside the target, Upstream part and Downstream part
* Logarithmic energy intervals
USRTRACK -1.0 209.0 -48.0 3.0 1000.0 20. piFluenU
USRTRACK 50.0 0.001  0. 0. 0. 0. &
USRTRACK -1.0 209.0 -49.0 4.0 1000.0 20. piFluenD
USRTRACK 50.0 0.001  0. 0. 0. 0. &
* Cartesian binning of the pion fluence inside and around the target
USRBIN  10.0 209.0 -50.0  50.0  50.0  50. piFluBin
USRBIN  -50.0 -50.0 -10.0 100.0 100.0 60.0 &
* Cartesian binning of the deposited energy inside the target
USRBIN  10.0 208.0 -51.0 10.0 10.0 5. Edeposit
USRBIN  -10.0 -10.0 0.0 20.0 20.0 5.0 &
RANDOMIZE 1.0 0. 0. 0. 0. 0.
START  100000.0 0. 0. 0. 0. 0.
STOP
```

2.13 Running FLUKA

It is advisable, but not mandatory, to keep separate the `$FLUPRO` directory from that or those where calculations are run and input files are kept. For instance, `flukawork`:

```
cd /home/user/flukawork
```

As mentioned above, the `rfluka` script in `$FLUPRO/flutil` should be used to drive the FLUKA run. In the following it is supposed that the user is going to ask for five statistically independent runs, each one made of 100000 histories, of the proposed basic example. The command is:

```
$FLUPRO/flutil/rfluka -N0 -M5 example &
```

(on LINUX/UNIX, the & character allows to run the program in the background without "freezing" the terminal window).

The script creates a temporary subdirectory called `fluka_nnnn` where `nnnn` is tbe number of the subprocess. For instance, when the first run (or "cycle") starts, the user will see on the terminal lines similar to the following ones:

```
$TARGET_MACHINE = Linux
$FLUPRO = /home/user/flupro
```

```
2789: old priority 0, new priority 10

Initial seed already existing

Running fluka in /home/user/flukawork/fluka_2789

================================ Running FLUKA for cycle # 1 ============================
```

At the end of each cycle the output files will be copied onto the running directory, the temporary directory will be erased and a new one will be created where the next run will take place. The names of the output files from each run are built by appending to the input file name the run number and an extension depending on their content: `.out` for the standard output, `.err` for the error file, `.log` for the log file and `_fort.nn` for the estimator files (with `nn` = absolute value of the selected output unit). The file containing the random number seeds will be called `ran`<input file name><run number>.

The error file may contain error messages related to the event generators (for instance when the program does not manage to conserve exactly energy or another quantity) or to the geometry tracking. Most of those are generally only warning messages which can be ignored unless there is a large number of them.

The log file generally contains messages related to fatal errors (input errors, overflow, etc.)

During a multiple run, lines like the following will appear on the user's screen:

```
================================ Running FLUKA for cycle # 1 ===================

Removing links

Removing temporary files

Saving output and random number seed

Saving additional files generated
     Moving fort.47 to /home/fasso/Fluka/test/example01_fort.47
     Moving fort.48 to /home/fasso/Fluka/test/example01_fort.48
     Moving fort.49 to /home/fasso/Fluka/test/example01_fort.49
     Moving fort.50 to /home/fasso/Fluka/test/example01_fort.50
     Moving fort.51 to /home/fasso/Fluka/test/example01_fort.51

================================ Running FLUKA for cyle # 2 ===================

Removing links

Removing temporary files

Saving output and random number seed

Saving additional files generated
     Moving fort.47 to /home/fasso/Fluka/test/example002_fort.47
     Moving fort.48 to /home/fasso/Fluka/test/example002_fort.48
     Moving fort.49 to /home/fasso/Fluka/test/example002_fort.49
     Moving fort.50 to /home/fasso/Fluka/test/example002_fort.50
     Moving fort.51 to /home/fasso/Fluka/test/example002_fort.51

================================ Running FLUKA for cycle # 3 ===================

.....

================================ Running FLUKA for cycle # 4 ===================

.....
```

```
================================ Running FLUKA for cycle # 5 ==================

Removing links

Removing temporary files

Saving output and random number seed

Saving additional files generated
    Moving fort.47 to /home/fasso/Fluka/test/example005_fort.47
    Moving fort.48 to /home/fasso/Fluka/test/example005_fort.48
    Moving fort.49 to /home/fasso/Fluka/test/example005_fort.49
    Moving fort.50 to /home/fasso/Fluka/test/example005_fort.50
    Moving fort.51 to /home/fasso/Fluka/test/example005_fort.51

End of FLUKA run
```

At this time, in the working directory, the following new files exist:

```
example001_fort.47   example002_fort.47   ....   example005_fort.47
example001_fort.48   example002_fort.48   ....   example005_fort.48
example001_fort.49   example002_fort.49   ....   example005_fort.49
example001_fort.50   example002_fort.50   ....   example005_fort.50
example001_fort.51   example002_fort.51   ....   example005_fort.51
example001.out       example002.out       ....   example005.out
example001.err       example002.err       ....   example005.err
```

In Chapter 9 the user can find a comprehensive description of the content of the FLUKA standard output. For the purpose of this beginner's guide, it can just be pointed out that, according to the content of the USRBDX command, the files with extension `fort.47` contain, in binary form, the boundary crossing estimator output for the required pion fluence and current detectors (for more details see 9.5.5). These files must be combined together to produce a table of values with their statistical errors which can be easily interfaced by the user to some analysis codes and/or graphic visualisation tools. Similarly, the files with extension `fort.48` and `fort.49` will contain the track-length estimator output, and those with extension `fort.50` and `fort.51` the output from USRBIN.

2.14 Accessing results

2.14.1 Boundary crossing estimator

Binary files from the USRBDX estimator must be accessed by means of the `usxsuw.f` readout code, which is located in the `$FLUPRO/flutil` directory.

That readout code can be easily compiled by means of the same tool used to compile and and link FLUKA:

```
cd $FLUPRO/flutil
./lfluka usxsuw.f -o usxsuw
```

In order to process the 54 output files produced by the proposed example, the following interactive procedure can be used:

```
cd /home/user/flukawork
$FLUPRO/flutil/usxsuw
```

The readout code will ask for the first FLUKA estimator file names:

```
Type the input file:
```

For each estimator file the program will show the content of the TITLE card of the FLUKA input file, the date and time of the FLUKA run and the number of histories for the given run.

The request will be iterated until a blank line is given. This will be interpreted as the end of the list of files, and then a name for the output file prefix will be requested. Let's use, for example, the prefix pionbdx:

```
 Type the input file: example001_fort.47
 Charged pion fluence inside and around a proton-irradiated Be target
 DATE:  7/15/ 5,  TIME: 16:22:11
  100000.
 100000
 Type the input file: example002_fort.47
 Charged pion fluence inside and around a proton-irradiated Be target
 DATE:  7/15/ 5,  TIME: 16:23: 3
  100000.
 100000
 Type the input file: example003_fort.47
 Charged pion fluence inside and around a proton-irradiated Be target
 DATE:  7/15/ 5,  TIME: 16:23:54
  100000.
 100000
 Type the input file: example004_fort.47
 Charged pion fluence inside and around a proton-irradiated Be target
 DATE:  7/15/ 5,  TIME: 16:24:51
  100000.
 100000
 Type the input file: example005_fort.47
 Charged pion fluence inside and around a proton-irradiated Be target
 DATE:  7/15/ 5,  TIME: 16:25:45
  100000.
 100000
 Type the input file: _
 Type the output file name: pionbdx
```

At this point the following 3 new files are produced:

```
pionbdx
piobdxn_sum.lis
pionbdx_tab.lis
```

The first one (pionbdx) is again a binary file that can be read out at any time by usxsuw. The content of this file is statistically equivalent to that of the sum of the files used to obtain it, and it can replace them to be combined with further output files if desired (the usxsuw program takes care of giving it the appropriate weight).
The other two files are ASCII text files.

Let us first examine pionbdx_sum.lis. This contains many comments which can help the user to understand the results. Since by means of the USRBDX command separate detectors for pion fluence and current have been requested, with their output on the same logical unit, there will be two different sections in the file, identified by the word "Detector": Detector no. 1 is for fluence and Detector no. 2 is for current, because this is the order in which the USRBDX commands have been given.

Let us inspect the output from Detector no. 1:

```
 Charged pion fluence inside and around a proton-irradiated Be target

   Total primaries run: 500000
   Total weight of the primaries run:  500000.
```

```
Detector n:  1( 1)  piFluenUD
   (Area:         400. cmq,
    distr. scored: 209   ,
    from reg. 3 to  4,
    one way scoring,
    fluence scoring)

   Tot. resp. (Part/cmq/pr)  8.6904905E-04  +/-  0.6976866     %
   ( -->      (Part/pr)      0.3476196      +/-  0.6976866     % )
```

The total (summed) number of primaries (histories) is reported at first, then the main features of `USRBDX` request are summarised. The following numbers represent the energy and angle integrated fluence ("total response").
Here and later, the statistical error is always expressed in percentage.

After this heading, the differential fluence tabulation as a function of (pion) energy, and integrated in solid angle, is reported, starting with the boundaries of the energy bins. As a general convention, these values are given from the highest to the lowest value:

```
**** Different. Fluxes as a function of energy ****
****      (integrated over solid angle)        ****


Energy boundaries (GeV):

 49.99992       40.27077       32.43475       26.12349       21.04029
 16.94620       13.64875       10.99293       8.853892       7.131072
 5.743484       4.625898       3.725775       3.000802       2.416896
 1.946608       1.567831       1.262757       1.017045      0.8191454
0.6597533      0.5313764      0.4279793      0.3447017      0.2776285
0.2236066      0.1800965      0.1450527      0.1168279      9.4095118E-02
7.5785778E-02  6.1039131E-02  4.9161937E-02  3.9595842E-02  3.1891152E-02
2.5685664E-02  2.0687662E-02  1.6662188E-02  1.3420003E-02  1.0808695E-02
8.7055033E-03  7.0115575E-03  5.6472253E-03  4.5483690E-03  3.6633324E-03
2.9505091E-03  2.3763892E-03  1.9139835E-03  1.5415541E-03  1.2415934E-03
Lowest boundary   (GeV):  1.0000000E-03

Flux (Part/GeV/cmq/pr):

1.5418744E-09 +/-    99.00000     %   4.8503271E-08 +/-    6.709127     %
2.3456116E-07 +/-    6.506497     %   5.9040013E-07 +/-    3.466331     %
1.2585346E-06 +/-    4.051404     %   2.5295039E-06 +/-    2.039807     %
4.6113087E-06 +/-    2.195296     %   7.6260553E-06 +/-    1.939942     %
1.2214471E-05 +/-   0.8310503     %   1.8394410E-05 +/-   0.6178440     %
2.6636921E-05 +/-    1.128397     %   3.6855919E-05 +/-    1.204921     %
5.1703457E-05 +/-    1.100655     %   6.9101960E-05 +/-   0.7564522     %
9.0419722E-05 +/-    1.799108     %   1.1945122E-04 +/-    1.256268     %
1.5757892E-04 +/-   0.8898824     %   1.9452766E-04 +/-    1.332425     %
2.4165030E-04 +/-    1.521364     %   3.0573772E-04 +/-    2.473622     %
3.6900895E-04 +/-    1.399170     %   4.4734811E-04 +/-   0.9543594     %
5.2953843E-04 +/-    1.964312     %   6.1596523E-04 +/-    1.349476     %
6.4003764E-04 +/-    3.323846     %   6.8828161E-04 +/-   0.9288639     %
6.8151421E-04 +/-    2.018673     %   7.0822553E-04 +/-    4.401796     %
7.4972271E-04 +/-    2.600316     %   6.9859857E-04 +/-    3.693749     %
6.8915845E-04 +/-    4.332464     %   6.6514849E-04 +/-    8.753220     %
6.4636284E-04 +/-    11.30834     %   5.5008888E-04 +/-    7.691558     %
4.3721433E-04 +/-    11.36630     %   3.2056248E-04 +/-    8.380781     %
4.2511927E-04 +/-    12.24571     %   2.2697043E-04 +/-    12.99932     %
2.0069227E-04 +/-    13.10813     %   1.7180138E-04 +/-    16.90801     %
9.9383309E-05 +/-    21.15753     %   2.9268101E-04 +/-    39.29378     %
```

```
 1.5672133E-04 +/-    44.01294     %   2.1093644E-04 +/-    34.72458     %
 7.4201569E-05 +/-    33.68359     %   7.2452240E-05 +/-    33.54827     %
 8.6934262E-05 +/-    62.03180     %   1.0245090E-04 +/-    99.00000     %
 1.6312006E-04 +/-    82.06016     %   1.3002084E-04 +/-    52.15991     %
```

Soon after, the cumulative fluence distribution as a function of energy is also given:

```
**** Cumulative Fluxes as a function of energy ****
****      (integrated over solid angle)        ****


Energy boundaries (GeV):

 49.99992       40.27077       32.43475       26.12349       21.04029
 16.94620       13.64875       10.99293       8.853892       7.131072
 5.743484       4.625898       3.725775       3.000802       2.416896
 1.946608       1.567831       1.262757       1.017045      0.8191454
0.6597533      0.5313764      0.4279793      0.3447017      0.2776285
0.2236066      0.1800965      0.1450527      0.1168279      9.4095118E-02
7.5785778E-02  6.1039131E-02  4.9161937E-02  3.9595842E-02  3.1891152E-02
2.5685664E-02  2.0687662E-02  1.6662188E-02  1.3420003E-02  1.0808695E-02
8.7055033E-03  7.0115575E-03  5.6472253E-03  4.5483690E-03  3.6633324E-03
2.9505091E-03  2.3763892E-03  1.9139835E-03  1.5415541E-03  1.2415934E-03
Lowest boundary   (GeV):  1.0000000E-03

Cumul. Flux (Part/cmq/pr):

1.5001119E-08 +/-    99.00000     %   3.9507350E-07 +/-    7.326498     %
1.8754495E-06 +/-    5.464718     %   4.8765669E-06 +/-    1.819896     %
1.0029117E-05 +/-    1.898280     %   1.8370021E-05 +/-    1.277005     %
3.0616819E-05 +/-   0.6900454     %   4.6929261E-05 +/-   0.9553517     %
6.7972585E-05 +/-   0.7029299     %   9.3496434E-05 +/-   0.6531623     %
1.2326548E-04 +/-   0.5382378     %   1.5644032E-04 +/-   0.6154544     %
1.9392396E-04 +/-   0.6043725     %   2.3427299E-04 +/-   0.5368618     %
2.7679623E-04 +/-   0.5548110     %   3.2204165E-04 +/-   0.6000980     %
3.7011484E-04 +/-   0.6263003     %   4.1791250E-04 +/-   0.6480659     %
4.6573509E-04 +/-   0.7125404     %   5.1446725E-04 +/-   0.7778813     %
5.6183949E-04 +/-   0.8066853     %   6.0809392E-04 +/-   0.7142704     %
6.5219263E-04 +/-   0.7654761     %   6.9350738E-04 +/-   0.7260005     %
7.2808337E-04 +/-   0.8159186     %   7.5803063E-04 +/-   0.7573094     %
7.8191340E-04 +/-   0.7549785     %   8.0190296E-04 +/-   0.7531289     %
8.1894622E-04 +/-   0.7366922     %   8.3173712E-04 +/-   0.6872664     %
8.4189989E-04 +/-   0.6799491     %   8.4980001E-04 +/-   0.6579692     %
8.5598318E-04 +/-   0.6862395     %   8.6022145E-04 +/-   0.6667165     %
8.6293457E-04 +/-   0.6859071     %   8.6453673E-04 +/-   0.6995495     %
8.6624804E-04 +/-   0.6864265     %   8.6698390E-04 +/-   0.6886846     %
8.6750800E-04 +/-   0.6864119     %   8.6786930E-04 +/-   0.6882262     %
8.6803763E-04 +/-   0.6885374     %   8.6843700E-04 +/-   0.6933275     %
8.6860918E-04 +/-   0.6915213     %   8.6879585E-04 +/-   0.6911866     %
8.6884876E-04 +/-   0.6931223     %   8.6889038E-04 +/-   0.6942393     %
8.6893054E-04 +/-   0.6953420     %   8.6896872E-04 +/-   0.6967193     %
8.6901762E-04 +/-   0.6981055     %   8.6904905E-04 +/-   0.6976866     %
```

The numbers for the cumulative distribution have been obtained by multiplying each value of the differential distribution by the corresponding energy bin width (variable if the distribution is logarithmic as in our example). The integral fluence in any given energy interval can be obtained as the difference between the values of the cumulative distribution at the two bounds of that interval.

Since more than one angular interval was requested, at this point the angular distribution *with respect to the normal at the boundary crossing point* is reported, both in steradians and in degrees:

```
**** Double diff. Fluxes as a function of energy ****


Solid angle minimum value (sr):   0.000000

Solid angle upper boundaries (sr):

0.6283185       1.256637       1.884956       2.513274       3.141593
 3.769911       4.398230       5.026548       5.654867       6.283185

Angular minimum value (deg.):   0.000000

Angular upper boundaries (deg.):

 25.84193       36.86990       45.57299       53.13010       60.00000
 66.42182       72.54239       78.46304       84.26083       90.00000
```

Let us take for instance the energy bin from 0.345 GeV to 0.278 GeV:

```
Energy interval (GeV):  0.3447016      0.2776284

Flux (Part/sr/GeV/cmq/pr):

2.2090337E-04 +/-    2.271138     %   1.6099877E-04 +/-    2.023665     %
1.2373505E-04 +/-    3.802638     %   9.4749055E-05 +/-    2.419357     %
7.0389280E-05 +/-    5.640523     %   6.6853667E-05 +/-    9.292711     %
6.8042267E-05 +/-    5.421218     %   6.8482914E-05 +/-    11.91976     %
5.8157104E-05 +/-    2.943847     %   4.8027632E-05 +/-    39.71496     %

Flux (Part/deg/GeV/cmq/pr):

5.3710260E-06 +/-    2.271138     %   9.1729089E-06 +/-    2.023665     %
8.9330297E-06 +/-    3.802638     %   7.8776966E-06 +/-    2.419357     %
6.4377805E-06 +/-    5.640523     %   6.5410413E-06 +/-    9.292711     %
6.9850003E-06 +/-    5.421218     %   7.2676362E-06 +/-    11.91976     %
6.3026077E-06 +/-    2.943847     %   5.2580162E-06 +/-    39.71496     %
```

The same structure is then replicated for Detector no. 2:

```
Detector n:  2( 2)  piCurrUD
   (Area:           400. cmq,
    distr. scored: 209   ,
    from reg. 3 to  4,
    one way scoring,
    current scoring)

   Tot. resp. (Part/cmq/pr)  7.1694393E-04  +/-  0.7243900     %
   ( -->      (Part/pr)      0.2867776      +/-  0.7243900     % )
```

and so on.
Note that in this case the ratio between the calculated fluence (8.690E-04) and the corresponding current (7.169E-04) is about 1.2. The ratio between the numerical values of the two quantities would be 1 if the pions were all crossing the boundary at a right angle, 2 in the case of an isotropic distribution, and could even tend to infinity if the particle direction were mainly parallel to the boundary:

Fluence and current are very different quantities and should not be confused!

Note also that the above output reports also the current value not normalised per unit area. This is equivalent to a simple count of crossing particles, so we see that in our example about 0.287 charged pions per primary proton cross the middle plane of the target.

The previous file has a structure which is not easily interfaceable to other readout codes. This can be easily achieved by means of the other output file, `pionbdx_tab.lis`: there the user can find, for each Detector, a simple 4-column structure for the differential fluence integrated over solid angle. The table starts from the lowest energy, and the four columns represent respectively E_{min}, E_{max}, the differential fluence and the statistical error in percentage:

```
 # Detector n:  1  piFluenUD (integrated over solid angle)
 # N. of energy intervals 50
  1.000E-03  1.242E-03  1.300E-04  5.216E+01
  1.242E-03  1.542E-03  1.631E-04  8.206E+01
  1.542E-03  1.914E-03  1.025E-04  9.900E+01
  1.914E-03  2.376E-03  8.693E-05  6.203E+01
  2.376E-03  2.951E-03  7.245E-05  3.355E+01
  2.951E-03  3.663E-03  7.420E-05  3.368E+01
  3.663E-03  4.548E-03  2.109E-04  3.472E+01
  4.548E-03  5.647E-03  1.567E-04  4.401E+01
  5.647E-03  7.012E-03  2.927E-04  3.929E+01
 .....
```

By convention, when in a given bin the statistics is not sufficient to calculate a standard deviation, the statistical error is printed as 99%. For a null fluence the statistical error is also null.

After this table, the double differential fluence is reported.
First, one or more lines marked by a # sign in column 1 give, from minimum to maximum, the extremes of the solid angle intervals. Then, for each energy interval, the minimum and maximum of the interval followed by as many pairs of values as the number of angular bins: the first value is the calculated double-differential quantity (fluence or current) in cm^{-2} sr^{-1} and the second is the corresponding statistical error in percent. For instance, for our example we obtain the following printout (for the sake of space only the first 3 bins in energy are shown):

```
 # double differential distributions
 # number of solid angle intervals 10
 #   0.000E+00  6.283E-01  6.283E-01  1.257E+00  1.257E+00  1.885E+00  ...
 #
....
  2.069E-02  2.569E-02  4.013E-05  2.472E+01  4.509E-05  2.068E+01  ...
  2.569E-02  3.189E-02  5.408E-05  1.907E+01  4.657E-05  2.200E+01  ...
  3.189E-02  3.960E-02  5.150E-05  7.137E+00  5.355E-05  1.587E+01  ...
....
```

2.14.2 Track length estimator

The program to analyse USRTRACK binary output is called `ustsuw.f` and can also be found in `$FLUPRO/flutil`. Its use is very similar to that of `usxsuw.f` described above. Applying it to the `example00*_fort.48` files (output of the first USRTRACK detector in our example), we obtain for the average fluence of charged pions in the upstream half of the beryllium target:

```
Tot. response (p/cmq/pr)  5.4765277E-04  +/-  0.6965669     %
```

and from the `example00*_fort.49` files (pion fluence in the downstream half):

```
Tot. response (p/cmq/pr)  1.3474772E-03  +/-  0.5352812     %
```

As it was to be expected, the average fluence obtained above by the boundary crossing estimator on the middle surface (`8.69E-04` cm^{-2}) has a value which is intermediate between these two.

2.14.3 Binning estimator

To analyse the binary output from USRBIN, two programs are needed, both available in `$FLUPRO/flutil`. The first, `usbsuw.f`, performs a statistical analysis of the results and produces a new unformatted file, with a name chosen by the user. The second program, `usbrea.f`, reads the latter file and writes on a formatted file two arrays, namely the content of each bin, averaged over the given number of runs, followed by the corresponding errors in percent. The second USRBIN detector defined in `example.inp` gives the following values of energy deposition (in GeV/cm^3):

```
1
  Cartesian binning n.   1  "Edeposit  " , generalized particle n.  208
     X coordinate: from -1.0000E+01 to  1.0000E+01 cm,    20 bins ( 1.0000E+00 cm wide)
     Y coordinate: from -1.0000E+01 to  1.0000E+01 cm,    20 bins ( 1.0000E+00 cm wide)
     Z coordinate: from  0.0000E+00 to  5.0000E+00 cm,     5 bins ( 1.0000E+00 cm wide)
     Data follow in a matrix A(ix,iy,iz), format (1(5x,1p,10(1x,e11.4)))

     accurate deposition along the tracks requested
      1.7164E-07  3.4587E-07  2.1976E-07  3.0997E-07  1.4963E-07  3.5431E-07  .....
      5.6597E-07  7.5792E-07  3.6563E-07  2.7822E-07  2.6084E-07  2.8645E-07  .....
      2.6191E-07  1.6716E-07  3.8680E-07  2.4925E-07  4.2334E-07  3.5025E-07  .....
      ..........................................................................
```

and the following corresponding percentage errors:

```
     Percentage errors follow in a matrix A(ix,iy,iz), format (1(5x,1p,10(1x,e11.4)))

      1.3936E+01  4.3211E+01  3.0601E+01  2.2874E+01  1.7783E+01  2.7942E+01  .....
      1.6548E+01  1.2291E+01  1.4539E+01  2.4576E+01  2.7828E+01  1.7247E+01  .....
      2.2423E+01  1.7258E+01  2.0349E+01  3.7997E+01  2.6855E+01  2.9230E+01  .....
      ..........................................................................
```

2.14.4 Readout of other FLUKA estimators

The `$FLUPRO/flutil` directory contains two other programs similar to `usxsuw.f` and `ustsuw.f` to average the outputs from other FLUKA estimators and binnings:

1. `usrsuw.f`: to read out the RESNUCLEI output (p. 187);
2. `usysuw.f`: to read out the USRYIELD output (p. 236);

2.15 Various settings

Accuracy and computer speed depend on several parameters that can be freely tuned by the user: but in general an increase of either one results in a decrease of the other. The proper balance must be based both on physical and on practical considerations. The present defaults have been designed to give satisfactory results in the most common cases: but other sets of defaults can be implemented using option DEFAULTS (to be placed at the beginning of the input file, just after the title).

Even when one set of defaults is enforced, the user can still override some of them by modifying single parameters. The most used ones concern energy cut-offs (option EMFCUT for electrons and photons, LOW–BIAS for low-energy neutrons, PART–THR for all other particles, described respectively in 7.21, 7.38 and 7.53), thresholds for delta-ray production (option DELTARAY, 7.14), particles to be ignored (option DISCARD, 7.16), switching on or off some physical effect (EMFCUT, IONFLUCT, MUPHOTON, PAIRBREM, PHOTONUC, PHYSICS, POLARIZA: see respectively 7.21, 7.35, 7.47, 7.52, 7.54, 7.55, 7.57), and (more rarely) the size of the step for transporting charged particles (FLUKAFIX, MCSTHRES, MULSOPT, STEPSIZE, see respectively 7.29, 7.44, 7.46, 7.67).

Energy cut-offs for each particle are listed in a table on standard output (`Particle transport thresholds`).

2.16 Biasing

Although FLUKA is able to perform fully analogue particle transport calculations (i.e., to reproduce faithfully actual particle histories), in many cases of very non-uniform radiation fields, such as those encountered in shielding design, only a very small fraction of all the histories contributes to the desired response (dose, fluence) in the regions of interest, for instance behind thick shielding. In these cases, the user's concern is not to simulate exactly what occurs in reality, but to estimate in the most efficient way the desired response. This can be obtained by replacing the actual physical problem with a mathematically equivalent one, i.e., having the same solution but faster statistical convergence.

A rigorous mathematical treatment of such variance-reduction techniques can be found in several textbooks (see for instance those of Lux and Koblinger [99] or Carter and Cashwell [37]). In the present practical introduction we will only issue a few important warnings:

- In the limit of the number of histories tending to infinity, the value of each calculated quantity tends *exactly* to the same average in the analogue and in the corresponding biased calculation. In other words, biasing is *mathematically correct* and implies no approximation. However, an acceleration of convergence in certain regions of phase space (space/energy/angle) will generally be paid for by a slower convergence in other regions.
  Because an actual calculation does not use an infinite number of particles, but is necessarily truncated after a finite number of histories, results must be considered with some judgment. For instance, if the number of histories is too small, it can happen that total energy is not conserved (check the energy budget summary at the very end of main output!)
- A bad choice of biasing parameters may have consequences opposite to what is desired, namely a slower convergence. A long experience, and often some preliminary trial-and-error investigation, are needed in order to fully master these techniques (but some biasing algorithms are "safer" than others).
- Because biasing implies replacing some distributions by others having the same expectation value but different variance (and different higher moments), biasing techniques in general do not conserve correlations and cannot describe correctly fluctuations.

The simplest (and safest) biasing option offered by FLUKA is importance biasing, which can be requested by option BIASING (p. 71). Each geometry region is assigned an "importance", namely a number between 10^{-4} and 10^{4}, proportional to the contribution that particles in that region are expected to give to the desired result. The ratio of importances in any two adjacent regions is used to drive a classical biasing algorithm ("Splitting" and "Russian Roulette"). In a simple, monodimensional attenuation problem, the importance is often set equal to the inverse of the expected fluence attenuation factor for each region.

In electron accelerator shielding, two other biasing options are commonly employed: EMF–BIAS (p. 98) and LAM–BIAS (p. 131). The first one is used to request leading particle biasing, a technique which reduces considerably the computer time required to handle high-energy electromagnetic showers. With this option, CPU time becomes proportional to primary energy rather than increasing exponentially with it. Option LAM–BIAS is necessary in order to sample with acceptable statistics photonuclear reactions which have a much lower probability than competing electromagnetic photon reactions, but are often more important from the radiological point of view.

Other important options are those which set weight window biasing (WW–FACTOr WW–THRESh and WW–PROFIle, described respectively in 7.83, 7.85 and 7.84) but their use requires more experience than assumed here for a beginner.

Particle importances, weight windows and low-energy neutron biasing parameters are reported for each region on standard output. On user's request (expressed as SDUM = PRINT in a BIASING card), Russian Roulette and Splitting counters are printed for each region on standard output before the final summary. Such counters can be used for a better tuning of the biasing options.

Chapter 3

Installation

The FLUKA package for LINUX and UNIX platforms is distributed as a tar file which can be downloaded from the FLUKA website `www.fluka.org`.

The release of the FLUKA source code will be available under the licence reported at the beginning of this volume. For most applications we distribute a package containing a compiled library, user routines in source form, `INCLUDE` files, various unformatted and formatted data files and a number of scripts for compiling, linking and running the program on a given platform. A list of the contents is provided in a `README` file, and information on the current version, possibly overriding parts of the present manual, may be contained in a file `RELEASE-NOTES`.

No external library routines are required. The timing and other necessary service routines are already included.

In principle, FLUKA can be installed on any computer platform where a Fortran compiler is available. However, at present only the following are supported (see the FLUKA website http://www.fluka.org for more information).

- Hewlett-Packard 9000 Series 700/800 running HP-UX
- Sun running SunOS
- DEC computers running Digital UNIX > 4.0
- Intel PCs running LINUX:
  - RedHat 7.3 (compiler gcc-2.96)
  - RedHat 9.0 (compiler gcc-3.2.2)
  - Fedora

It is suggested that the user define an environmental variable `FLUPRO` pointing to the directory where the distribution tar file has been opened. The FLUKA libraries and most data files will be located in `$FLUPRO`, the `INCLUDE` files in `$FLUPRO/flukapro/`, the default user routines in `$FLUPRO/usermvax/`, compilation and linking scripts (as well as several postprocessing programs to analyse user scores) in `$FLUPRO/flutil/`.

If the source code is included in the distribution, it will be contained in `$FLUPRO` files with names of the form `...vax.for`[1].

The basic FLUKA program on UNIX machines consists of 30 files:

```
bamjmvax.for   blockmvax.for  comlatmvax.for decaymvax.for  dedxmvax.for
dumvax.for     elsmvax.for    emfmvax.for    eventqmvax.for eventvmvax.for
evffrmvax.for  fluoxmvax.for  fluxesmvax.for geolatmvax.for kaskadmvax.for
lowneumvax.for mainmvax.for   mathmvax.for   neutrimvax.for noptmvax.for
opphmvax.for   outputmvax.for pemfmvax.for   pgmvax.for     preclmvax.for
preeqmvax.for  pripromvax.for rndmvax.for    usermvax.for   XXXmvax.for (*)
```

`(*) XXX` stands for `hp, ibm, linux, osf` etc. depending on the platform. Most UNIX Fortran compilers require that the extension `.for` be replaced by `.f` (but the Makefile provided with FLUKA takes care of this, see below).

See Chap. 4 for a short description of the content of these files.

[1]The form `...vax.for` has historical reasons. Actually, as seen later, the files are automatically split by a `Makefile` into single routines (with extension `.f`) before compilation

If the source code is present, the `INCLUDE` files needed to compile the program may be grouped into three files `emfadd.add`, `flukaadd.add` and `lowneuadd.add`.

A Makefile and a number of auxiliary programs split these files into individual routines and `INCLUDE` files, which are placed in 30+1 separate directories and compiled. The object files are inserted in a FLUKA library `libflukahp.a`. A shell script `lfluka` links all routines into an executable `flukahp` (the name is the same for all UNIX platforms, the "`hp`" being due to historical reasons).

The DPMJET and RQMD object files are collected in two separate libraries.

The FLUKA distribution `tar` file normally does not contain an executable file. To create the default FLUKA executable, type:

```
$FLUPRO/flutil/lfluka -m fluka
```

(the name of the resulting executable will be `flukahp`)

or, if heavy ion nuclear interactions are needed:

```
$FLUPRO/flutil/ldpmqmd
```

(the name of the resulting executable will be `flukadpm`)

User-written routines (in particular a `SOURCE` subroutine, see list of user interface routines in Chap. 12) can be compiled separately and linked overriding the default routines of the library. The `$FLUPRO/flutil/lfluka` script can take care of them in three different ways:

1. appending the Fortran files (`xxx.f`) as last arguments to the `lfluka` procedure (LINUX only);
2. appending the object files (precompiled using the `$FLUPRO/flutil/fff` procedure supplied with the code) as last arguments to the `lfluka` procedure;
3. inserting the object files into a library and giving the library name to the script with the `-O` or `-l` options.

An online help is available issuing `lfluka -h`.

The program may need up to 13 auxiliary data files, containing cross-sections, nuclear and atomic data and so on. Seven of these files are unformatted and have an extension `.bin` (or `.dat`).

The seven unformatted files and five of the remaining auxiliary files require no modification by the user.

They are generally kept in the main FLUKA directory.

Here is the list:

fluodt.dat Fluorescence emission data, needed for problems involving low-energy electron-photon transport

neuxsc_72.bin Low-energy neutron cross-sections: needed for all problems with neutron transport below 20 MeV.

nuclear.bin Nuclide masses, abundances and other data: needed for all hadronic problems

elasct.bin Elastic cross-sections for hadronic problems

sigmapi.bin Pion cross-sections

brems_fin.bin Bremsstrahlung cross-sections

e6r1nds3.fyi, jef2.fyi, jendl3.fyi, xnloan.dat Fission nuclide yields and neutron multiplicities

3.1 Pre-connected I/O files

FLUKA reads its main input from standard logical unit 5 and writes its main output to logical unit 11. Both are parameterised in the `INCLUDE` file `IOUNIT` as `LUNIN` and `LUNOUT`, and can therefore be redefined

if necessary. Assignment of unit number 5 and log messages to the corresponding files is achieved (on Linux/UNIX) via the redirection symbols < and >. Other input and output files on UNIX can be assigned a I/O unit number by means of symbolic links (but the syntax for Fortran implicit connection is not standard and forms like `fort.xx` or `ftnxx` can both be found on different platforms). An alternative way is offered by the OPEN command of FLUKA(p. 158), which allows to perform explicit connections.

The `$FLUPRO/rfluka` script supplied with the code contains all relevant I/O file definitions, and can be used to run the code interactively or through a batch queue. It allows to submit multiple runs with a single command. Both `rfluka` and `lfluka` (the script used for linking, see above) contain usage instructions.

The `rfluka` script creates a temporary directory where it copies the necessary files and deletes it after the results have been copied back to the parent directory, thus allowing to run more than one job at the same time in the same directory. Appropriate names for the output files are generated by `rfluka`, including a sequential number for each run.

If user routines are linked and a new executable is created, the name of the new executable can be input using the `-e` option. Some on-line help is available issuing `rfluka -h`.

Chapter 4

FLUKA modules (Fortran files)

Since several years, the FLUKA source code has been organised in "modules". This word must not be intended to have the technical meaning which it has been assigned later in Fortran 90. A FLUKA module is simply a collection of routines covering the same physics field or belonging to the same class of functionality.

In the FLUKA2005 version, there are 31 modules:

`BAMJM` : new hadronisation package (strongly improved version of the original BAMJET) [150]
`BLOCKM` : `BLOCK DATA` routines
`COMLATM` : all geometry routines specific for Combinatorial Geometry (CG) and repetition (lattice capability) in the geometry description
`DECAYM` : all routines connected with particle decays during transport, including those managing polarisation or non phase-space-like decays for pions, kaons, muons and taus
`DEDXM` : all routines connected with dE/dx and radiative losses of hadrons and muons, including delta-ray production and cross-sections for muon photonuclear interactions
`DPMM` : interface routines for the DPMJET generator
`DUM` : dummy routines
`ELSM` : all hadron and photon (photonuclear) cross-section routines
`EMFM` : all routines dealing with electron, positron and photon transport and interactions (except for photon photonuclear interactions, fluorescent X-ray and Auger production)
`EVENTQM` : auxiliary routines for the high energy hadronic interaction generators
`EVENTVM` : all routines (besides those in `EVENTQM`) connected with the high energy hadronic inelastic interaction package
`EVFFRM` : separate module with all evaporation, fission and Fermi break-up routines
`FLUOXM` : all routines dealing with fluorescence X-ray and Auger production
`GEOLATM` : geometry navigation and debugging routines
`KASKADM` : general event steering, most of the relevant transport routines for hadron and muon transport, including magnetic field tracking, most of material and region dependent initialisation and source routines
`LOWNEUM` : all routines concerning the multigroup treatment of "low" energy (E $<$ 19.6 MeV) neutrons
`MAINM` : main, input parsing and initialisation
`MATHM` : mathematical auxiliary routines (interpolation, integration, etc.). Many of them adapted from SLATEC (http://www.netlib.org/slatec)
`NEUTRIM` : nuclear interactions of neutrinos
`NOPTM` : all routines connected with new scoring options implemented after FLUKA86, and blank `COMMON` setting for scoring options.
`OPPHM` : optical photon production and transport
`OUTPUTM` : printing routines (apart from output of new options which is performed in `NOPTM`)
`PEMFM` : electromagnetic initialisation
`PGM` : PLOTGEOM geometry drawing package
`PRECLM` : full PEANUT second part
`PREEQM` : full PEANUT first part
`PRIPROM` : initialisation and drivers for PEANUT
`RNDM` : random number generation, including gaussian-distributed random numbers
`RQMDM` : interface routines for the RQMD generator
`USERM` : user oriented routines (see list below)
`IBMM/HPM/LINUXM/OSFM/VAXM` : timing and "environment" routines. These are machine specific.

User oriented routines (see description in Chap. 12):

The "FLUKA User Routines" mentioned at point 3) in the FLUKA User License are those (and only those) contained in the directory `usermvax`, both in the source and binary versions of the code.

`ABSCFF` : absorption coefficient (for optical photons)
`COMSCW` : response functions, user dependent selection for density-like quantities
`DFFCFF` : diffusion coefficient (for optical photons)
`ENDSCP` : energy density distributed — change of positions
`FLDSCP` : fluence distributed — change of positions
`FLUSCW` : response functions, user dependent selection for flux-like quantities
`FORMFU` : nuclear charge form factors
`FRGHNS` : material roughness (for optical photons)
`FUSRBV` : defines a continuous variable for 3-D binnings
`LATTIC` : symmetry transformation for lattice geometry
`LUSRBL` : defines a discrete variable for 3-D binnings
`MAGFLD` : to use a magnetic field map
`MDSTCK` : management of secondary stack
`MGDRAW` : to dump trajectories, etc.
`MUSRBR` : defines a discrete variable for 3-D binnings
`OPHBDX` : boundary crossing properties (for optical photons)
`QUEFFC` : quantum efficiency (for optical photons)
`RFLCTV` : reflectivity (for optical photons)
`RFRNDX` : refraction index (for optical photons)
`SOEVSV` : saving source events
`SOURCE` : to generate any distribution for source particles
`STUPRE` : set user variables (electrons and photons)
`STUPRF` : set user variables (hadrons, muons and neutrinos)
`UBSSET` : to override input biasing parameters
`UDCDRL` : decay direction biasing
`USIMBS` : user-defined importance biasing
`USREIN` : event initialisation
`USREOU` : post-event output
`USRINI` : user initialisation
`USRMED` : to perform user driven biasing and/or particle selections on a material basis
`USROUT` : user output
`USRRNC` : user customisation of residual nuclei scoring

Part II

User Guide

Chapter 5

Particle and material codes

5.1 Particles transported by FLUKA

Each particle which can be transported by FLUKA is identified by an integer number. Negative values of such identifiers are reserved to light and heavy ions, and to optical photons. The value 0 indicates a pseudoparticle RAY, which can be used to scan the geometry. Numbers $>$ 200 designate "families" of particles, grouped according to some common characteristics (all hadrons, or all charged particles, etc.). In FLUKA, they are called Generalised Particles and can be used only for scoring. Various forms of scored energy are also treated as Generalised Particles.

The identifier values are reported in Table 5.1 together with the corresponding particle numbering scheme of the Particle Data Group [118].

Table 5.1: FLUKA particle names and code numbers

FLUKA name	FLUKA number	Symbol	Common name	Standard PDG number (Particle Data Group) [118]
4-HELIUM (1)	-6	α	Alpha	—
3-HELIUM (1)	-5	^{3}He	Helium 3	—
TRITON (1)	-4	^{3}H	Triton	—
DEUTERON (1)	-3	^{2}H	Deuteron	—
HEAVYION (1)	-2	—	Generic Heavy Ion (see command HI–PROPE)	—
OPTIPHOT	-1	—	Optical Photon	—
RAY (2)	0	—	Pseudoparticle	—
PROTON	1	p	Proton	2212
APROTON	2	$\bar{\text{p}}$	Antiproton	-2212
ELECTRON	3	e^-	Electron	11
POSITRON	4	e^+	Positron	-11
NEUTRIE	5	ν_e	Electron Neutrino	12
ANEUTRIE	6	$\bar{\nu}_e$	Electron Antineutrino	-12
PHOTON	7	γ	Photon	22
NEUTRON	8	n	Neutron	2112
ANEUTRON	9	$\bar{\text{n}}$	Antineutron	-2112
MUON+	10	μ^+	Positive Muon	-13
MUON-	11	μ^-	Negative Muon	13
KAONLONG	12	K^0_L	Kaon-zero long	130
PION+	13	π^+	Positive Pion	211
PION-	14	π^-	Negative Pion	-211
KAON+	15	K^+	Positive Kaon	321
KAON-	16	K^-	Negative Kaon	-321
LAMBDA	17	Λ	Lambda	3122
ALAMBDA	18	$\bar{\Lambda}$	Antilambda	-3122
KAONSHRT	19	K^0_S	Kaon-zero short	310
SIGMA-	20	Σ^-	Negative Sigma	3112
SIGMA+	21	Σ^+	Positive Sigma	3222
SIGMAZER	22	Σ^0	Sigma-zero	3212
PIZERO	23	π^0	Pion-zero	111
KAONZERO	24	K^0	Kaon-zero	311
AKAONZER	25	$\bar{\text{K}}^0$	Antikaon-zero	-311
Reserved	26	—	—	—
NEUTRIM	27	ν_μ	Muon Neutrino	14
ANEUTRIM	28	$\bar{\nu}_\mu$	Muon Antineutrino	-14
Blank	29	—	—	—

table continues

FLUKA name	FLUKA number	Symbol	Common name	Standard PDG number (Particle Data Group)
Reserved	30	—	—	—
ASIGMA-	31	$\bar{\Sigma}^-$	Antisigma-minus	-3222
ASIGMAZE	32	$\bar{\Sigma}^0$	Antisigma-zero	-3212
ASIGMA+	33	$\bar{\Sigma}^+$	Antisigma-plus	-3112
XSIZERO	34	Ξ^0	Xi-zero	3322
AXSIZERO	35	$\bar{\Xi}^0$	Antixi-zero	-3322
XSI-	36	$\bar{\Xi}^-$	Negative Xi	3312
AXSI+	37	$\bar{\Xi}^+$	Positive Xi	-3312
OMEGA-	38	Ω^-	Omega-minus	3334
AOMEGA+	39	$\bar{\Omega}^+$	Antiomega	-3334
Reserved	40	—	—	—
TAU+	41	τ^+	Positive Tau [3]	-15
TAU-	42	τ^-	Negative Tau [3]	15
NEUTRIT	43	ν_τ	Tau Neutrino	16
ANEUTRIT	44	$\bar{\nu}_\tau$	Tau Antineutrino	-16
D+	45	D^+	D-plus	411
D-	46	D^-	D-minus	-411
D0	47	D^0	D-zero	421
D0BAR	48	$\bar{\mathrm{D}}^0$	AntiD-zero	-421
DS+	49	D_s^+	D_s-plus	431
DS-	50	D_s^-	D_s-minus	-431
LAMBDAC+	51	Λ_c^+	Lambda$_c$-plus	4122
XSIC+	52	Ξ_c^+	Xi$_c$-plus	4232
XSIC0	53	Ξ_c^0	Xi$_c$-zero	4112
XSIPC+	54	$\Xi_c^{\prime+}$	Xi$_c^\prime$-plus	4322
XSIPC0	55	$\Xi_c^{\prime 0}$	Xi$_c^\prime$-zero	4312
OMEGAC0	56	Ω_c^0	Omega$_c$-zero	4332
ALAMBDC-	57	$\bar{\Lambda}_c^-$	Antilambda$_c$-minus	-4122
AXSIC-	58	$\bar{\Xi}_c^-$	AntiXi$_c$-minus	-4232
AXSIC0	59	$\bar{\Xi}_c^0$	AntiXi$_c$-zero	-4112
AXSIPC-	60	$\bar{\Xi}_c^{\prime-}$	AntiXi$_c^\prime$-minus	-4322
AXSIPC0	61	$\bar{\Xi}_c^{\prime 0}$	AntiXi$_c^\prime$-zero	-4312
AOMEGAC0	62	$\bar{\Omega}_c^0$	AntiOmega$_c$-zero	-4332
Reserved	63	—	—	—
Reserved	64	—	—	—

[1] Heavy fragments produced in evaporation are loaded in a special stack (`COMMON FHEAVY`, contained in the `INCLUDE` file with the same name).
The internal code for heavy evaporation fragments is the following: 3 = deuteron, 4 = ^{3}H, 5 = ^{3}He, 6 = ^{4}He, 7–12 = fission fragments. Transport capabilities (dE/dx, with account of effective charge and effective charge straggling, multiple Coulomb scattering, no interaction yet) are now available for d, t, ^{3}He and ^{4}He. Heavier ions can be transported on demand (see option EVENTYPE), with or without nuclear interactions. Fission fragments, when produced, are also put in `COMMON FHEAVY` with id's ranging from 7 to 12 (usually 7 and 8 for two fragments).

[2] A RAY is not a real particle, but a straight line trajectory through the FLUKA geometry. When a primary particle (defined by options BEAM and BEAMPOS, or by a `SOURCE` subroutine) is found to be a RAY, the program tracks through the geometry in the given direction calculating a number of quantities (distance traversed in each material, number of radiation lengths, number of interaction lengths etc.). See Chap. 14 for instructions about its use.

[3] Only leptonic decays of τ's are implemented for the time being

Table 5.2: FLUKA generalised particles (to be used only for scoring)

FLUKA name	FLUKA number	Description
—	40	Low-energy neutrons (used only in some input options)
ALL-PART	201	All transportable particles
ALL-CHAR	202	All charged particles
ALL-NEUT	203	All neutral particles
ALL-NEGA	204	All negative particles
ALL-POSI	205	All positive particles
NUCLEONS	206	Protons and neutrons
NUC&PI+-	207	Protons, neutrons and charged pions
ENERGY	208	For dose scoring: Deposited energy For energy fluence scoring: Kinetic energy
PIONS+-	209	Charged pions
BEAMPART	210	Primary (source or beam) particles
EM-ENRGY	211	Electromagnetic energy (of electrons, positrons or photons)
MUONS	212	Muons
E+&E-	213	Electrons and positrons
AP&AN	214	Antiprotons and antineutrons
KAONS	215	All kaons
STRANGE	216	All kaons and all hyperons and anti-hyperons (i.e., all strange particles)
KAONS+-	217	Charged kaons
HAD-CHAR	218	Charged hadrons
FISSIONS	219	Fissions
HE-FISS	220	High energy fissions
LE-FISS	221	Low energy fissions
NEU-BALA	222	Neutron balance (algebraic sum of outgoing neutrons minus incoming neutrons for all interactions)
HAD-NEUT	223	Neutral hadrons
KAONS0	224	Neutral kaons
C-MESONS	225	Charmed mesons
C-(A)BAR	226	Charmed (anti)baryons
CHARMED	227	Charmed hadrons
Blank	228	—
UNB-ENER	229	Unbiased deposited energy [4]
UNB-EMEN	230	Unbiased electromagnetic energy (of electrons, positrons or photons) [4]
X-MOMENT	231	X component of momentum transfer
Y-MOMENT	232	Y component of momentum transfer
Z-MOMENT	233	Z component of momentum transfer

[4] "Unbiased energy" means that the energy deposited (or the energy fluence) is scored with weight 1, independent of the actual weight of the particle. Of course, the result will have no physical meaning, but in some circumstances it will provide useful information about the run itself (for instance in order to optimise biasing).

5.2 Pre-defined materials

Materials can be easily defined by option MATERIAL (p. 142), by assigning a density, a name, a code number, and, in the case of single elements, an atomic and a mass number. For compounds, the MATERIAL option card must be accompanied by a COMPOUND definition (p. 78) referred to the same material name. If low-energy neutrons (E $<$ 20 MeV) need to be transported, the chosen name of a single element material must coincide with that of one for which cross-sections are available (see Table 10.3).

However, for user's convenience, 25 common materials are already pre-defined (see Table 5.3): they are assigned a default density, name and code number even if no MATERIAL definition has been given. The user can override any of these if desired (for instance re-assigning code numbers), and can add more material definitions, by means of one or more MATERIAL cards. The only constraint is that the number sequence of the defined materials be *uninterrupted*, i.e., there may not be any gap in the numbering sequence from 25 onwards.

Table 5.3: List of pre-defined FLUKA materials

FLUKA name	FLUKA number	Common name	**A**	**Z**	Density (g/cm^3)
BLCKHOLE	1	Blackhole or External Vacuum	0.	0.	0.
VACUUM	2	Vacuum or Internal Vacuum	0.	0.	0.
HYDROGEN	3	Hydrogen	1.00794	1.	0.0000837
HELIUM	4	Helium	4.002602	2.	0.000166
BERYLLIU	5	Beryllium	9.012182	4.	1.848
CARBON	6	Carbon	12.0107	6.	2.000
NITROGEN	7	Nitrogen	14.0067	7.	0.00117
OXYGEN	8	Oxygen	15.9994	8.	0.00133
MAGNESIU	9	Magnesium	24.3050	12.	1.740
ALUMINUM	10	Aluminium	26.981538	13.	2.699
IRON	11	Iron	55.845	26.	7.874
COPPER	12	Copper	63.546	29.	8.960
SILVER	13	Silver	107.8682	47.	10.500
SILICON	14	Silicon	28.0855	14.	2.329
GOLD	15	Gold	196.96655	79.	19.320
MERCURY	16	Mercury	200.59	80.	13.546
LEAD	17	Lead	207.2	82.	11.350
TANTALUM	18	Tantalum	180.9479	73.	16.654
SODIUM	19	Sodium	22.989770	11.	0.971
ARGON	20	Argon	39.948	18.	0.00166
CALCIUM	21	Calcium	40.078	20.	1.550
TIN	22	Tin	118.710	50.	7.310
TUNGSTEN	23	Tungsten	183.84	74.	19.300
TITANIUM	24	Titanium	47.867	22.	4.540
NICKEL	25	Nickel	58.6934	28.	8.902

Chapter 6

General features of FLUKA input

The input of FLUKA consists of a text file containing a sequence of option lines (often called "cards") which are followed sometimes by data cards specific of the option (or command) requested. Option cards have all the same structure, and can be read in in fixed format or in free format. A description of free format is given in Chap. 7, option GLOBAL (p. 125).

```
CODEWD, (WHAT(I), I = 1, 6), SDUM
```

(the fixed format is `A8, 2X, 6E10.0, A8`)

where:

- CODEWD is the option keyword
- The WHAT-parameters are numerical data (or logical data in numerical form)
- SDUM, if present, contains character data (only in one exceptional case, the STERNHEIme option, SDUM contains numerical information)

Note that even if the values to be assigned to WHAT-parameters were logically integers, because of the format used they must be given with a decimal point. The order of the input cards is almost free, with the following exceptions:

- GLOBAL declarations, if present, must precede any executable option.
- Option DEFAULTS must be issued at the very beginning of input. It can be preceded only by a GLOBAL card and by commands such as COMMENT or TITLE.
- The START command initiates execution. While old versions of FLUKA were allowing multiple re-starts, only the first START command is executed now. Thus any input given after START is ignored, with the exception of STOP.
- The STOP command stops the execution of the program. Thus any input present after STOP is ignored.
- In some cases, the MAT–PROP option must be requested after the corresponding MATERIAL card.
- The PLOTGEOM command must be issued after the geometry input, and, in case the user chooses to plot only boundaries between different materials, it must come also after all the ASSIGNMAt cards. It is also recommended that PLOTGEOM be issued before any biasing and any other option which makes use of permanent and/or temporary storage.
- Some option cards must or can be immediately followed by a variable amount of information, not always in the standard format indicated above. These are:

  COMMENT:
  an optional number of cards following the COMMENT command are considered as comments and are merely reproduced in the output. In case COMMENT is issued with SDUM option = INPUT or MODEL, one single card must follow (see p. 76).

  OPEN is generally followed by the name of the file to be opened (scratch files are an exception). See p. 158.

  DETECT, USRBIN, USRBDX, USRCOLL, USRTRACK, USRYIELD, EVENTBIN, EVENTDAT:
  data concerning user-defined estimators and binnings extend in general over two cards. The second ("continuation card") must come after the first, but doesn't need to follow it immediately.

Input included between GEOBEGIN and GEOEND:
geometry data must be given in a well-defined order and in a special format between a GEOBEGIN and a GEOEND definition (but the LATTICE geometry option and the GEOBEGIN and GEOEND cards themselves follow the normal FLUKA format convention).

PLOTGEOM:
Unless a different logical input unit is specified, the call to the PLOTGEOM program must be followed immediately by the PLOTGEOM input, in special format (see p. 179).

TITLE:
the card following the TITLE command is considered as the title of the run and is reproduced in the output (see p. 207).

Most definitions have some default values. If these are acceptable, it is not compulsory that the corresponding option card appear explicitly in the input sequence. Furthermore for most WHAT and/or SDUM parameters a default value (that may be different from the default value when the definition has not been input) is applied if the corresponding field is left blank (or set = 0.0) in the input card.

Several option cards may appear more than once in the input sequence. In most cases, each of such additional cards obviously adds more definitions to those already given, provided they are different and not contradictory. In case of conflict, the last given generally overrides the previous one(s). This feature may be successfully exploited in the numerous cases where whole arrays are assigned according to the scheme

"From ... to ... in step of ..." (corresponding to a Fortran DO-loop)

making the input more compact. An example can be found in the description of option ASSIGNMAt (7.2), which is used to set a one-to-many correspondence between material numbers and region numbers.

In most cases of such "DO-loop" assignments, especially when the same option card can be used to assign a value to more than one quantity, a blank or zero field does not assign the default value but leaves the previously given value unchanged. To remove any possible ambiguity, resetting the default value needs then to be done explicitly (generally -1. has to be input in such cases). An example can be found in the description of option EMF–BIAS (7.20).

All defaults and exceptions are listed under the description of each FLUKA input option. Different defaults, tuned to the type of application of interest, can be specified using the option DEFAULTS (p. 83).

6.1 Physical units

Physical units consistently used in FLUKA input and output are:

distance	cm (and derived units cm^2, cm^3 for areas and volumes)
energy	GeV
	Exceptions:
	eV is used for average ionisation potential input by option MAT–PROP
	g MeV^{-1} cm^{-2} are used for Birks coefficients input by option TCQUENCH
momentum	GeV/c
temperature	degree Kelvin
solid angle	sr (exception: degrees may be used, on user's request, with option USRYIELD)
magnetic field	T
electric field	kV/cm
time	s (option TCQUENCH) or ns (option TIME–CUT)

6.2 The input preprocessor

FLUKA Version 2005 comes bundled with an internal preprocessor, a simplified version of a C-like preprocessor. The preprocessor can modify the input file before it is executed with the use of conditions. Presently the functionality is limited to 2 types of directives, **definition** and **conditional**, and probably will be expanded

in the future. The preprocessor is a particularly useful way to include or remove blocks of input cards, allowing a more flexible and easy organisation of an input file. One can write the input file around a few directives to allow easier debugging, changing thresholds, biasing and scoring cards for the final production.

6.2.1 Syntax

All preprocessor directives are single lines starting with the `#` character in the first column and can appear anywhere in the input file, either between normal input cards or inside the geometry definition (inline or externally defined). Each identifier can be up to 10 characters in length.

6.2.2 Definition of Constants

With the **definition directives** one can define identifiers to be used later for inclusion or removal parts of the input file:

```
#define [identifier_name]
```

defines `[identifier_name]` without giving it a value. This can be used in conjunction with another set of directives that allow conditional execution.

```
#undef [identifier_name]
```

deletes any previously defined `[identifier_name]`.

6.2.3 Conditional directives

With the **conditional directives** one can include or remove parts of the input file before execution. The `#if, #elif, #else` blocks must be terminated with a closing `#endif`. There is a maximum of 10 nesting levels that can be used.

```
#if [identifier_name]
...
#elif [identifier_name]
...
#else
...
#endif
```

The `#if` and `#elif` (`else-if`) directive is followed by an identifier. If the identifier is defined then the statement evaluates to true, otherwise to false.

Example:

```
#define DEBUG
#define PLOT1
...
#if DEBUG
*...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
GEOEND         100.0     100.0     100.0    -100.0    -100.0    -100.0          DEBUG
GEOEND          50.0      50.0      50.0                                       &
#else
GEOEND
#endif
...
#if PLOT1
*...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
PLOTGEOM               1.0              -2000.0
MBWD6L1
    -100.0       0.0  -21620.0     100.0       0.0  -21250.0
       1.0       0.0       0.0       0.0       0.0       1.0
```

```
    -100.0       0.0  -21200.0     100.0       0.0  -20800.0
STOP
#endif
```

Chapter 7

Description of FLUKA input options

There are more than 70 option keywords available for input in FLUKA. A summary is given in Section 7.1, where the commands will be shortly introduced and grouped by type of functionality. Some of the commands, which can provide several different services, will be mentioned in the context of more than one group.

A complete description of each command will follow in Sections 7.2 to 7.85, in alphabetical order.

7.1 Introduction to the FLUKA input options

7.1.1 Summary of the available options

Here is a list of the options (commands) that are at the disposal of the FLUKA user to prepare an input file. In the rest of this section, the same commands will be presented by grouping them according to the different services they can provide.

ASSIGNMAt	defines the correspondence between region and material indices and defines regions where a magnetic field exists
BEAM	defines most of the beam characteristics (energy, profile, divergence, particle type)
BEAMAXES	defines the axes used for a beam reference frame different from the geometry frame
BEAMPOS	defines the starting point of beam particles and the beam direction
BIASING	sets importance sampling (Russian Roulette/splitting) at boundary crossings and at high-energy hadronic collisions on a region by region basis
BME	defines some I/O parameters relevant to the heavy ion event generator BME
COMMENT	inserts comments in the input or passes to the program information about the computer used and about the input file
COMPOUND	defines a compound or a mixture or a mixture of isotopes
CORRFACT	allows to alter material density for dE/dx and nuclear processes on a region-by-region basis
DCYSCORE	associates selected scoring detectors of given estimator type with user-defined decay times
DCYTIMES	defines decay times for radioactive product scoring
DEFAULTS	sets FLUKA defaults for specified kinds of problems
DELTARAY	activates delta ray production by heavy charged particles and controls energy loss and deposition
DETECT	scores energy deposition in coincidence or anti-coincidence with a trigger, on an event by event basis
DISCARD	defines the particles which must not be transported
DPMJET	defines some I/O parameters relevant to the heavy ion event generator DPMJET
ELCFIELD	sets the tracking conditions for transport in electric fields and possibly defines an homogeneous electric field *(not yet implemented)*
EMF	requests detailed transport of electrons, positrons and photons
EMF–BIAS	defines electron/photon leading particle biasing or biases electron/photon interaction length
EMFCUT	sets energy cut-offs for electrons, positrons and photons, for transport and production, or for switching off some physical interactions
EMFFIX	sets the size of electron steps corresponding to a fixed fraction of the total energy
EMFFLUO	activates production of fluorescence X rays in selected materials
EMFRAY	activates Rayleigh (coherent) scattering in selected regions

EVENTBIN	scores energy or star densities in a binning structure independent from the geometry, and prints the binning output after each "event" (primary history)
EVENTDAT	prints event by event the scored star production and/or energy deposition in each region, and the total energy balance
EVENTYPE	defines the hadron particle production model to be used
EXPTRANS	requests exponential transformation ("path stretching") *(not yet implemented)*
FLUKAFIX	sets the size of the step of muons and charged hadrons to a fixed fraction of the kinetic energy
FREE	switches to free-format input (geometry excluded)
GEOBEGIN	starts the geometry description
GEOEND	ends the geometry description; can also be used to activate the geometry debugger
GLOBAL	issues global declarations about the class of the problem (analogue or weighted) and about the complexity of the geometry. It also allows to use free format input (geometry included)
HI–PROPE	defines the properties of a heavy ion primary
IONFLUCT	calculates ionisation energy losses with fluctuations
IRRPROFI	defines an irradiation profile for radioactive decay calculations
LAM–BIAS	biases decay length and interaction length
LOW–BIAS	requests non-analogue absorption and defines the energy cut-off for low-energy neutron transport on a region by region basis
LOW–DOWN	biases the downscattering probability in low-energy neutron transport on a region by region basis
LOW–MAT	sets the correspondence between FLUKA materials and low-energy neutron cross-section data
LOW–NEUT	requests low-energy neutron transport
MATERIAL	defines a material and its properties
MAT–PROP	supplies extra information about gaseous materials and materials with fictitious or inhomogeneous density and defines other material properties
MCSTHRES	defines energy thresholds for applying the multiple Coulomb scattering algorithm to the transport of muons and charged hadrons
MGNFIELD	sets the tracking conditions for transport in magnetic fields and possibly defines an homogeneous magnetic field
MULSOPT	controls optimisation of multiple Coulomb scattering treatment
MUPHOTON	controls photonuclear interactions of high-energy heavy charged particles (mediated by virtual photons)
MYRQMD	defines some I/O parameters relevant to the new heavy ion event generator RQMD
OPEN	defines input/output files without pre-connecting
OPT–PROP	defines optical properties of materials
OPT–PROD	controls Cherenkov and Transition Radiation photon production
PAIRBREM	controls simulation of pair production and bremsstrahlung by high-energy heavy charged particles
PART–THR	sets different energy cut-offs for selected particles
PHOTONUC	activates photon interactions with nuclei
PHYSICS	controls some physical processes for selected particles
PLOTGEOM	calls the PLOTGEOM package to draw a slice of the geometry
POLARIZA	defines polarised beams (only for photons at present)
RADDECAY	requests simulation of radioactive decays and sets the corresponding biasing and transport conditions
RANDOMIZe	sets the seeds for the random number generator or the number of calls to the generator to be skipped before start
RESNUCLEi	scores residual nuclei after inelastic hadronic interactions

ROT–DEFIni	defines rotations/translations to be applied to user-defined binnings
ROTPRBIN	sets the storage precision (single or double) and assigns possible rotations/translations for a given user-defined binning (USRBIN or EVENTBIN)
RQMD	defines some I/O parameters relevant to the heavy ion event generator RQMD
SCORE	defines the (generalised) particles to be scored by region
SOURCE	tells FLUKA to call a user-written source routine
START	defines the number of primary particles to follow, gets a primary particle from a beam or from a source, starts the transport and repeats until the predetermined number of primaries is reached
STEPSIZE	sets the maximum step size in cm (by region) for transport of charged particles
STERNHEIme	allows users to input their own values of the density effect parameters
STOP	stops input reading
TCQUENCH	sets scoring time cut-offs and/or Birks quenching parameters
THRESHOLd	defines the energy threshold for star density scoring, and sets thresholds for elastic and inelastic hadron reactions
TIME-CUT	sets transport time cut-offs
TITLE	gives the title of the run
USERDUMP	requests a collision file and defines the events to be written
USERWEIG	defines extra weighting to be applied to scored yields, fluences, doses, residual nuclei or star densities (at scoring time)
USRBDX	defines a detector for a boundary crossing fluence or current estimator
USRBIN	scores energy, star density or particle fluence in a binning structure independent from the geometry
USRCOLL	defines a detector for a collision fluence estimator
USRICALL	calls user-dependent initialisation
USROCALL	calls user-dependent output
USRTRACK	defines a detector for a track-length fluence estimator
USRYIELD	defines a detector for scoring particle yield around a given direction
WW–FACTOr	defines weight windows in selected regions
WW–PROFIle	defines energy group-dependent extra factors ("profiles") to modify the basic setting of the low-energy neutron weight windows in selected sets of regions, or the low-energy neutron importances in each region
WW–THRESh	defines the energy limits for a RR/splitting weight window

7.1.2 Basic commands

Most FLUKA commands are optional, and if anyone of them is not used an appropriate set of defaults is provided. A few commands, however, are nearly always needed in order to provide a meaningful definition of the problem to be studied.

In general, for a problem to be fully determined, the following elements need to be defined:

1. the radiation source
2. the geometrical layout
3. the materials
4. the requested results
5. setting of parameters, accuracy, conditions, and in general technical directives to the program on how the calculation shall be performed

Defaults are provided in FLUKA for all the above features, but those for items 1, 2 and 3 are unlikely to be useful: therefore the few commands used to define source, geometry and materials are practically always present in the input file.

For what concerns item 4, the user has a choice of several options to request the estimation of various radiometric quantities. Of course, there is no much point in running the program without requesting any result, but in a phase of input preparation it is quite common to have a few runs without any scoring commands. A typical minimum input containing only specifications for the above items 1, 2 and 3 will still produce some useful information. Looking at the standard FLUKA output, the user can do several consistency checks, and can get some better insight into the problem from the final statistics and energy balance (see 9.1).

The last part of problem definition, element 5 (setting) is important but is supported by very robust defaults. In many cases, the only user concern should consist in choosing the right set of defaults. However, there are some applications which require explicit setting commands, for instance to request photonuclear reactions for electron accelerator shielding.

7.1.2.1 Definition of the radiation source

The simplest particle source is pointlike, monoenergetic and monodirectional, that is, a "particle beam". Option BEAM, fully described in 7.3, is used to define the particle type and momentum (or energy). If desired, this option can also define an energy spread, a beam profile shape and an angular divergence. However, the two latter distributions are restricted to a beam is directed in the positive z direction: to describe divergence and beam profile for an arbitrary beam direction it is necessary to define a beam reference frame by means of option BEAMAXES (p. 67).

The energy declared with BEAM is used by the program to initialise cross-section tables and other energy-dependent arrays: therefore that command must always be present, even when a more complex source is described by means of a user routine.

The particle starting point and direction are declared by means of option BEAMPOS (see 7.5). If BEAMPOS is not present, the beam particles are assumed to start from the origin of the coordinates `0., 0., 0.` and to be directed along the z axis. It is important that the starting point be not on a boundary and not inside a blackhole region. In many cases, starting in vacuum upstream of the actual target can be convenient.

Both BEAM and BEAMPOS commands can be placed anywhere in the input file, before the START command.

Particle sources with more complicated features (with arbitrary distribution in energy, space, angle, time, and even with more than one type of particle) can be described by a user-written subroutine `SOURCE` (see 12.2.19). To call it, a command SOURCE (see 7.65) must be present in input.

7.1.2.2 Description of the geometry

The Combinatorial Geometry used by FLUKA is based on two important concepts: *bodies* and *regions*. The first ones are closed solid bodies (spheres, parallelepipeds, etc.) or semi-infinite portions of space (half-spaces, infinite cylinders) delimited by surfaces of first or second degree. The user must combine *bodies* by boolean operations (addition, intersection and subtraction) to perform a complete partition of the space of interest into *regions*, namely cells of uniform material composition. One important rule to remember is that inside the space of interest, defined by means of an external closed body, **every point must belong to one and only one region**.

Input for the geometry description, which has its own format and rules, explained in Chap. 8, must be contained between a GEOBEGIN and a GEOEND card. These two cards follow the normal FLUKA input syntax. An option offered by the GEOBEGIN command (Sec. 7.31) is to read the geometry input from a separate file. Command GEOEND (7.32) can be used also to invoke the geometry debugger, a check which is always strongly recommended.

Geometry input, sandwiched between a GEOBEGIN and a GEOEND card, can be placed anywhere in the input file (before the START command). It is mandatory in all cases.

An optional command related to geometry is PLOTGEOM. It is used to display sections of the geometry

and needs to read its own input, as explained in 7.56.

7.1.2.3 Materials

Materials in FLUKA are identified by a name (an 8-character string) and by a number, or material index. Both are used to create correspondences, for instance between region number and material number, or between material name and neutron cross-section name.

Some materials are already pre-defined. Table 5.3 on p. 44 lists the 25 available pre-defined materials with their default name, index number, density and atomic and mass number. The user can either refer to any one of them as it is, or override it with a new number, name and other properties, or define a new material. In the latter two cases, the new material definition is done by option MATERIAL (7.42). If the material is not a single element or isotope, but a compound, mixture or alloy, a command COMPOUND (7.9), extended on as many cards as necessary, is needed to specify its atomic composition. The correspondence between the material and the composition is set using the same name in the MATERIAL and in the COMPOUND cards. Note that material names, if low-energy neutron transport is desired, cannot be assigned arbitrarily but must match one of the names available in the FLUKA cross-section library (see Table 10.3 on p. 295).

Once all the materials to be assigned to the various geometry regions have been defined (either explicitly with MATERIAL or implicitly in the pre-defined list), it is necessary to specify of which material each region is made, by setting a correspondence *material index* → *region number*. This is done by command ASSIGNMAt (7.2).

Command ASSIGNMAt is used also to indicate that a magnetic field exists inside one or more given regions: in this case a command MGNFIELD is needed to specify intensity and direction of a constant magnetic field, or a complex one defined by a user routine as explained below in 7.1.7. Note that in practice at least one ASSIGNMAt command must always be present.

A less common kind of correspondence is set by option LOW–MAT (7.40). By default, the correspondence between a material and a low-energy neutron cross-section set is established by name, but in some circumstances this cannot be done, for instance when two different materials share the same cross-section set, or when two cross-section sets have the same name. Option LOW–MAT can be used to set a different correspondence.

Another FLUKA option concerning the definition of materials is MAT–PROP (7.43). It is used for a variety of purposes: to describe porous, inhomogeneous or gas materials, to override the default average ionisation potential, to do a rough temperature re-scaling of thermal neutron cross-sections when no cross-section set is available at a desired temperature, and to request a call to a special user routine when particles are transported in a given material.

7.1.3 Setting options

Many FLUKA input options are not used to describe the radiation transport problem but to issue directives to the program about how to do the calculations. Other options are used just to select a preferred input format. We refer to these options as "setting options".

Thanks to a complete and well-tuned set of defaults, setting options are not always necessary, especially for a beginner or in a preliminary phase of input preparation. However, an experienced user can often improve considerably the code performance by a judicious selection of parameters.

7.1.3.1 Format setting

The default, fixed input format can be replaced by a free format using option FREE (7.30) or better GLOBAL (7.33). The latter allows to choose free format for both the normal input and the geometry input separately, and serves also a few other purposes: it can be used to increase the maximum allowed number of geometry regions, and to force a calculation to fully analogue (i.e.,, simulating physical reality as directly as possible, without any biasing to accelerate statistical convergence). A more esoteric capability of GLOBAL, used mainly for debugging, is to ensure that the random number sequence be exactly reproduced even in cases where the geometry tracking algorithm has the possibility to follow different logical paths to achieve the same result.

7.1.3.2 General setting options

The difficult task of choosing the best settings for a calculation problem is made much easier by existence of several "pre-packaged" sets of defaults, each of which is optimised for a particular type of application. Each set is chosen by option DEFAULTS, which has to be placed at the beginning of the input file, possibly preceded only by TITLE or GLOBAL. Several possibilities include hadrotherapy, calorimetry, pure electromagnetic runs without photonuclear reactions, low-energy neutron runs without gamma production, and others (see 7.13). One set of defaults is tuned for maximum precision (but not necessarily great time efficiency). Reasonable defaults, acceptable for most generic routine calculations, are provided in case DEFAULTS is missing. In most cases, the user has the possibility to use some of the other setting options described below, to override one or more of the defaults provided by the chosen set.

In any case, it is important to check the list of defaults to make sure that nothing important is missing or has been overlooked. For instance, photonuclear reactions, which are critical for electron accelerator shielding, are not provided by any of the available default sets and must be added by the user by means of the PHOTONUC command.

Another setting option, DISCARD, is used to indicate particles which shall not be transported. The energy of those particles is not deposited anywhere but is added up in an accumulator which is printed at the end of the FLUKA standard output. Of course it is the user's responsibility to see that the discarded particles *and their progeny* would not give a significant contribution to the requested results.

7.1.3.3 Multiple Coulomb scattering

The concept of multiple scattering is an approximation to physical reality (condensed history approximation [18]), where charged particles undergo a very large number of single collisions with the atomic electrons, too many to be simulated in detail except in very special cases. All the theoretical treatments which have been developed are valid only within certain limits, and none of them gives rules on how to handle material boundaries and magnetic fields. FLUKA uses an original approach [64], based on Molière's theory [22, 113–115], which gives very good results for all charged particles in all circumstances (even in backscattering problems), preserving various angular and space correlations and freeing the user from the need to control the particle step length.

Although the default treatment is always more than satisfactory, the user has the possibility to request various kinds of optimisation, for both electrons/positrons and heavy charged particles. This can be done by means of option MULSOPT (7.46), which offers also the possibility to switch off completely multiple scattering in selected materials. The latter is a technique used when simulating particle interactions in very rare gases such as are contained in accelerator vacuum chambers (gas bremsstrahlung): the simulation is done for a gas of much larger density and the results are scaled to the actual low density: but scaling is meaningful only if no scattering takes place.

Another very important feature of option MULSOPT is single scattering, which can be requested in various degrees at boundary crossing, when the limits of Molière's theory are not satisfied, and even all the time (but the latter possibility is to be used only for problems of very low energy, because it is very demanding in CPU time).

There is also another option connected with multiple scattering, which however concerns only heavy charged particles such as hadrons and muons: MCSTHRESh (7.44) allows to set a threshold below which multiple Coulomb scattering is not performed. However, the CPU time saved is minimal and the option is not frequently used.

7.1.3.4 Step length

Another aspect of the condensed history approximation is that charged particle transport is performed in *steps*. The finite fraction of the particle energy which is lost and deposited in matter in each step is an approximation for the sum of innumerable tiny amounts of energy lost by the particle in elastic and inelastic collisions.

In early Monte Carlo programs results could depend critically on the size of the step, mainly due to the inaccurate determination of the path length correction (ratio between the length of the actual wiggling path of the particle and that of the straight step connecting the two endpoints). For a more complete discussion, see [6,59]. The multiple scattering algorithm used by FLUKA [64] provides a robust independence of the results from the step size, but for problems where a special accuracy is requested, or when magnetic fields are present, it is possible for the user to override the default step length. Two options control the maximum fractional energy loss per step: EMFFIX for electrons and positrons (7.22), and FLUKAFIX for muons and charged hadrons (7.29). The second one is seldom used, however, except in problems of very large dimensions typical of cosmic ray research. Option STEPSIZE (7.67) is used instead to limit the absolute length of the step, independent of the energy lost. Contrary to EMFFIX and FLUKAFIX, it works also in vacuum. While its use is highly recommended in problems with magnetic fields, to ensure that steps be smaller than the dimensions of the current region and of those that border it, when no magnetic fields are present this option should better be avoided, as it would imply no obvious advantage and could even downgrade performance.

7.1.3.5 Energy cut-offs

Setting energy cut-offs, for both transport and production, is an important responsibility of the user, who is interested in choosing the best compromise between accuracy and time efficiency. Each of the parameter sets available via option DEFAULTS, including the basic defaults set which exists when that option has not been explicitly requested, offers a well-optimised choice for the corresponding class of applications, with only one exception. But even so, it is often convenient to override some of the default cut-offs in order to improve performance. The exception concerns the default particle production cut-offs for electrons, positrons and photons, which are dependent on other settings (see EMFCUT below).

Transport cut-offs, or thresholds, are set with command PART–THRes (7.53) for hadrons and muons, with EMFCUT (7.21) for electrons, positrons and photons, and with LOW–BIAS (7.38) for low-energy neutrons. Despite the similar functionality of the three commands, there are important differences in their syntax and in the way the threshold is implemented for the three families of particles. PART–THRes can assign different transport thresholds to different particles, but the thresholds are the same in all materials and regions. When the hadron or muon energy becomes lower than the relevant threshold, the particle is not stopped but ranged out in a simplified way. Because the electron and photon cut-offs are more critical with respect to calculation accuracy, EMFCUT can assign transport thresholds on a region basis: on the other hand no ranging out is performed, due to the difficulty to clearly define electron ranges. For low-energy neutrons, the transport threshold is set by LOW–BIAS also on a region basis, but as a group number rather than an energy.

Two input commands can set particle production cut-offs, respectively for heavy particles and for electrons, positrons and photons.
Thresholds for delta ray production by charged hadrons and muons are assigned, on a material basis, by means of option DELTARAY. Energy transfers to electrons lower than the threshold are handled in the continuous slowing down approximation.
Production of bremsstrahlung by electrons and of Møller/Bhabha secondary electrons is simulated explicitly above thresholds set *on a material basis* with option EMFCUT. Defaults for electron and photon production cut-offs are dependent on other settings in a complex way. Therefore it is recommended to check the values printed on standard output, or to set EMFCUT production cut-offs explicitly for each material. Note also that the same EMFCUT command is used to set both transport and production cut-offs: but the setting is done by region in the first case and by material in the second.

To complete the list of commands used for cut-off setting, we will mention THRESHOLd (7.71), which is used to set an energy threshold for star scoring. In principle, a "star" is any high energy inelastic hadron interaction (spallation) and star density has always been one the quantities which can be scored by FLUKA. Since a popular technique to estimate induced radioactivity was based originally on the density of stars produced by hadrons with energies higher than 50 MeV, the possibility to set a scoring energy limit is provided.

7.1.3.6 Time cut-offs

For time-dependent calculations, two time cut-off options are available: one for particle transport, TIME–CUT, and one for scoring, TCQUENCH. While option TIME–CUT (7.72) sets a particle-dependent time limit after which the corresponding particle history is terminated, the limits set by TCQUENCH (7.70) are assigned to selected binnings. Scoring contributions to a binning by particles having exceeded the corresponding time limit are ignored, but particle transport continues, possibly contributing to other detector scores.

7.1.3.7 Ionisation energy loss

Transport of charged particles can be done in many ways: without delta ray production and ionisation fluctuations (continuous slowing down approximation), with ionisation fluctuations and no delta rays, with delta ray production above a chosen energy threshold and no ionisation fluctuations below the threshold, and with both: delta rays above the threshold and ionisation fluctuations below it. Depending on the application type chosen with option DEFAULTS, different defaults and thresholds apply, which can be modified by the user by means of options IONFLUCT, DELTARAY and EMFCUT. Option IONFLUCT (7.35) is used to request (restricted) ionisation fluctuations on a material basis. In FLUKA, these fluctuations are not simulated according to Landau or Vavilov theory but according to an original statistical approach [57]. They can be requested separately for electrons and positrons and for muons and charged hadrons. Delta ray production thresholds are instead set for the two particle families by two separate options, which have already been mentioned above in the context of production cut-offs (7.1.3.5): EMFCUT (7.21) and DELTARAY (7.14). DELTARAY can be used also to define (and print) the mesh width of the stopping power tabulations used by the program.

The user has also the possibility to change the default parameters used in the calculation of stopping power. Command STERNHEIme (7.68) allows to change the density effect parameters, and MAT–PROP (7.43) can set, in addition to several other material properties, a user-defined average ionisation potential.

7.1.4 Special radiation components and effects

In FLUKA, an effort has been made to implement a full cross-talk between different radiation components (hadronic, muonic, electromagnetic, low-energy neutrons, heavy ions, optical photons). However, some components are not activated by default, and others are only activated in some of the available default settings. Input options are provided to switch them on and off.

In a similar way, some physical effects may need to be activated, overriding the chosen defaults. On the other hand, in some cases it can be of interest (but possibly dangerous!) to ignore some effects. A number of commands are available for these purposes.

7.1.4.1 Radiation components

High-energy hadrons and muons are always generated and transported, except with defaults settings EM–CASCA and NEUTRONS (however, they cannot be requested overriding these two defaults). To suppress them, one can use DISCARD.

Option EMF (**E**lectro**M**agnetic **F**luka (7.19) can be used to request electron, positron and photon transport, and also to ask for its suppression (the latter could be obtained also by discarding electrons, positrons and photons by means of DISCARD).

Low-energy neutron transport (if not already on by default) can be activated with option LOW–NEUT. Explicit suppression is not possible: but the same effect can be obtained using option LOW–BIAS to set a cut-off at group 1.

Heavy ion transport (only ionisation energy loss, without nuclear interactions) is implicit with some default settings, while with others it is not available. Details can be found in the description of command EVENTYPE (7.27). The same command can be used also to request heavy ion interactions using different event generators: in this case the corresponding libraries must be linked.

A special option, HI–PROPErt, is necessary to define the properties of a heavy ion primary, since the

particle type input via the BEAM command can only be a generic heavy ion.

Generation and transport of optical photons is available only on explicit user request. Activation (and deactivation) are requested via OPT–PROD (for Cherenkov, transition radiation or scintillation photon production) and OPT–PROP (transport).

7.1.4.2 Physics effects

Some physical effects are automatically activated, but only when certain default sets are in force (see option DEFAULTS in 7.13 and Table 7.1 on p. 88), and can be switched on or off with appropriate commands. The command to simulate fluorescence is EMFFLUO (7.23), that for Rayleigh scattering and Compton binding corrections is EMFRAY (7.24), while for multiple scattering there are MULSOPT and MCSTHRES which we have already introduced in 7.1.3.3. High-energy effects such as production of bremsstrahlung and electron pairs by heavy charged particles (in particular muons) are regulated by option PAIRBREM (7.52).

A few physical effects need to be requested explicitly, whatever the defaults. These are photon polarisation (command POLARIZA, 7.57), polarisation of pion, kaon and muon decays (command PHYSICS, 7.55), photonuclear reactions (PHOTONUC, 7.54) and muon hadronic interactions via virtual photons (MUPHOTON, 7.47).

In some cases, it is also possible to switch off some important effects to study the relative importance of different processes. Command THRESHOLd allows to set a lower energy limit for hadron elastic scattering and inelastic reactions, and EMFCUT does the same with various kinds of electron and photon interactions. The user must bear in mind, however, that results obtained suppressing effects which are critical for the development of the electromagnetic or hadronic cascade are unlikely to be physically correct.

7.1.5 Scoring options

Any result in a Monte Carlo calculation is obtained by adding up the contributions to the "score", or "tally" of a detector defined by the user. A detector is the Monte Carlo equivalent of a measurement instrument. Each "estimator" (detector type) is designed to estimate one or more radiometric quantities, and the final score is a statistical estimation of the average value of the corresponding population. As in experimental measurements, it is possible to calculate a standard deviation by running several independent calculations.

No default detector is available: each scoring option must be explicitly requested. There are different input options corresponding to different types of detector. The simplest is SCORE which provides energy deposition (proportional to dose) or star density in every region of the geometry. "Stars" is an old name for inelastic hadron reactions which derives from early experiments with nuclear emulsions.

The same quantities can be scored in a uniform spatial mesh independent of geometry, called a "binning", by means of option USRBIN (7.77). There are several types of binnings: Cartesian, 2D-cylindrical, 3D-cylindrical and even more complex phase space structures. In addition to dose and star density, it is possible to use USRBIN to score particle fluence distributions. USRBIN results are often displayed as colour plots where each colour corresponds to a pre-defined range of values. A post-processing program for this purposes (`PAWLEVBIN`) is available in the directory `$FLUPRO/flutil`, and a GUI interface can be downloaded from the FLUKA website `www.fluka.org`.

Fluence, averaged over the volume of a given geometry region, can be calculated with options USRTRACK (7.81) and — less often — USRCOLL (7.78). The first is a "track-length estimator" (it estimates fluence as volume density of particle trajectory lengths), and the second is a "collision estimator" (fluence is estimated as volume density of collisions weighted with the particle mean free path). Of course, USRCOLL can be used only in a region of matter, while USRTRACK works also in vacuum. Both options provide fluence differential energy spectra.

Another common scoring option is USRBDX (7.76), which also calculates fluence, but averaged over the boundary between two geometry regions. It is a "boundary crossing estimator", which estimates fluence as the surface density of crossing particles weighted with the secant of the angle between trajectory and normal to the boundary at the crossing point. Option USRBDX can also calculate current, i.e., a simple

counter of crossings, not weighted by inverse cosine: but despite a widespread credence, current is only seldom a quantity worth calculating. The results of USRBDX can account on request for particles crossing the boundary from either side or from one side only, and are in the form of double-differential energy and angular spectra. The angle considered is again that with the normal to the boundary at the crossing point.

USRYIELD is a multi-purpose estimator option, which can estimate several different double-differential quantities. The main one is an energy-angle double-differential yield of particles escaping from a target, the angle this time being with respect to a fixed direction. Energy and angle can be replaced by many other variables which are mostly of the same kind, such as momentum and rapidity. But it is possible also to score yields as a function of charge and LET (linear energy transfer).

Production of residual nuclei can be obtained with command RESNUCLEi. The results, which are closely related to induced activity and dose rate from activated components, may include nuclei produced in low-energy neutron interactions, provided the corresponding information is available in the neutron cross-section library for the materials of interest.

7.1.5.1 Event by event scoring options

Typical particle physics applications, in particular calorimetry, require separate scoring event by event (that is, results are printed after each primary particle history). Two commands, EVENTBIN (7.25) and EVENTDAT (7.26), are respectively the event-equivalent of USRBIN and SCORE which have been introduced before. A third command, DETECT, allows to score event by event energy deposition simulating a detector trigger, defining coincidences and anticoincidences. All these options are incompatible with any biasing. It is suggested to use command GLOBAL (7.33) to make sure that the run will be completely analogue.

7.1.5.2 Scoring modifying options

There are a few commands which are used to modify some of the scoring options already described. TCQUENCH (7.70), which has already been shown to define a time cut-off, can be used also to apply a quenching factor (Birks factor) to energy deposition scored with USRBIN or EVENTBIN. ROT–DEFI (7.61) and ROTPRBIN (7.62) allow to define roto-translation transformations for binnings not aligned with the coordinate axes. ROTPRBIN can be used also to set the binning storage precision: a space saving feature, which is useful mainly when scoring event by event with EVENTBIN.

7.1.5.3 Options to handle radioactive decay

It is possible to transport and score in the same run also the beta and gamma radiation emitted in the decay of radioactive nuclei produced in the hadronic or electromagnetic cascade. Several options are available for this purpose: RADDECAY is used to request the simulation of radioactive decays, IRRPROFIle defines a time profile for the intensity of the primary particles, DCYTIMES requests one or more decay times at which the desired scoring shall occur, and DCYSCORE associates selected scoring detectors to the decay times so requested.

7.1.6 Biasing options

When run in fully analogue mode, FLUKA allows the user to study fluctuations and correlations, and to set up a direct simulation of physical reality where all moments of phase space distributions are faithfully reproduced. On the other hand, in the many applications where only quantities averaged over many events are of interest, it is convenient to use calculation techniques converging to the correct expectation values but reducing the variance (or the CPU time, or both) by sampling from biased distributions. This is especially useful in deep penetration calculations, or when the results of interest are driven by rare physical interactions or cover a small domain of phase space.

FLUKA makes available several biasing options. Some are easy to use, but others require experience and judgment, and often a few preliminary preparation runs are needed to optimise the biasing parameters.

7.1.6.1 Simple biasing options

The easiest biasing command is fittingly called BIASING (7.6). It provides two different kinds of variance reduction: *Multiplicity Reduction* and *Importance Biasing*, which is based on the two complementary techniques *Geometry Splitting* and *Russian Roulette* (RR).

Splitting and Russian Roulette are two classical variance reduction techniques, which are described in most textbooks on Monte Carlo [37, 99]. A detailed description of how they are implemented in FLUKA is available on p. 72. Importance biasing consists in assigning an importance value to each geometry region. The number of particles moving from a region to another will increase (by splitting) or decrease (via RR) according to the ratio of importances, and the particle statistical weight will be modified inversely so that the total weight will remain unchanged. In this way, the user can strive to keep the particle population constant, making up for attenuation, or to make it decrease in regions far from the detectors where there is a lower probability to contribute to the score. In FLUKA, importance biasing can be done separately for hadrons/muons, electrons/positrons/photons and low-energy neutrons.

Multiplicity Reduction is a simple technique which was introduced for the first time in FLUKA (now it has been adopted also by other programs), in order to decrease the computer time needed to simulate a very high energy hadron cascade. At energies of several hundred GeV and more, the number of secondaries produced in a hadron-nucleus interaction is very large and the total number can increase geometrically in the following interactions, requiring an unacceptably long computer time. Since many secondaries are particles of the same kind and with a similar angular and energy distribution, the user can decide to follow only a region-dependent fraction of them.

A biasing option performing a similar multiplicity reduction on electromagnetic showers is EMF–BIAS (7.20). In this case the technique is known as *Leading Particle Biasing* and consists in sampling only one of the two secondary particles which are present in the final state of most electromagnetic interactions. The secondary of higher energy is sampled with higher probability. The EMF–BIAS option can be tuned per region below user-defined energy thresholds and is used very often in shielding calculations for high-energy electron accelerators. The same command can be used also to bias the electron and photon mean free path for various types of interaction, e.g., to enhance the probability of interaction in a thin or low-density target.

In a similar way, option LAM–BIAS can be used to increase the probability of hadronic interactions, and in particular photohadron reactions. These are the dominant reactions for high-energy electron accelerator induced activity and shielding design, but because their cross-section is small compared to that of electromagnetic effects, analogue sampling would be very inefficient. The same command can help to get a higher probability of hadron interaction in a thin target. It can also be used to bias a particle decay length (for instance, to enhance muon or neutrino production) and the emission angle of the decay secondaries in a direction indicated by the user.

7.1.6.2 Weight window options

The weight window is a very powerful biasing technique, not based on relative importances, but on the absolute value of particle weight. The user sets an upper and a lower limit for the particle weight in each geometry region, possibly tuned per type of particle and energy. Splitting and RR will be applied so that the weight of all relevant particles will have a value between the two limits. In addition to controlling the particle population, this techniques helps also to "damp" excessive weight fluctuations due to other biasing options.

Its use is not as easy as that of importance biasing, because it is necessary to have at least a rough idea of what are the average weights in different regions. Special splitting and RR counters can be printed on request to help setting the window parameters (see p. 9.1). Weight window setting is done in FLUKA by three input commands: WW–FACTOr, WW–THRESh and WW–PROFIle. The first two commands must be used together: WW–FACTOR (7.83) sets the upper and lower weight limits per region, while WW–THRESh (7.85) defines energy limits within which the weight window must be applied, and the particles to which it is to be applied. The third option is reserved to low-energy neutrons, whose transport characteristics often require a more detailed biasing pattern: WW–PROFIle (7.84) allows indeed to tune the weight window by neutron energy group.

7.1.6.3 Biasing options for low-energy neutrons

The special multigroup transport structure used by FLUKA for low-energy neutrons calls for some biasing options specific to these particles. We have just introduced the weight window command WW–PROFIle. Two more options are LOW–BIAS (7.38), which has already been mentioned before in the context of energy cut-offs, but which is used also to set a user-defined non-analogue absorption probability, and LOW–DOWN (7.39), by which it is possible to bias neutron thermalisation (downscattering). The latter, however, is an option recommended only to users with a good knowledge and experience of neutronics.

7.1.7 Calls to user routines

The purpose of several FLUKA input options is to trigger calls to user routines (user routines are described in Chap. 12). One of the most important ones is SOURCE (7.65), which makes FLUKA get the characteristics its primary particles from subroutine `SOURCE` instead of from options BEAM and BEAMPOS. This option allows to pass to the subroutine several parameters, thus allowing to drive it from input without the need to re-compile it. Note that even when using a user-written source, it is still necessary to have in input a BEAM card indicating the maximum expected energy of a primary particle, so that the program can prepare appropriate cross-section tables. If command SOURCE is present, but no `SOURCE` routine has been linked, the default one in the FLUKA library will be called, which leaves unchanged the particle type, energy, position etc. as defined by BEAM and BEAMPOS.

Command USERWEIG (7.75) can call 5 different user routines used to modify a scored quantity (at the time of scoring). The routines are:

- `FLUSCW` is a function returning a multiplication factor for fluences (12.2.6). A typical application is to convert a fluence to dose equivalent.
- `COMSCW` is a function returning a multiplication factor for star densities and doses (12.2.2). Common application: converting energy deposition to dose.
- `USRRNC` is a subroutine returning a multiplication factor for residual nuclei (12.2.29).
- `ENDSCP` is a subroutine performing a displacement of the energy deposited in a particle step, for instance to account for an instrument drift (12.2.4).
- `FLDSCP` is a subroutine performing a displacement (drift) of the track corresponding to a particle step (12.2.5).

Complex magnetic fields can be defined or read from a map by a user routine `MAGFLD` (12.2.11. Calls to the routine are activated by command MGNFIELD (7.45).

A collision file (also called a collision tape, or a phase space file) is a file on which FLUKA writes on request details of user-selected events: particle trajectories, energy deposition events, source particles, boundary crossings, physical interactions, etc. This task is performed by subroutine `MGDRAW` (12.2.13, which is called if option USERDUMP is given in input (7.74). The default routine present in the FLUKA library can be driven as it is, without re-compilation, by setting some of the USERDUMP parameters, but can also be modified and re-compiled to adjust to specific needs of the user. A typical simple task is to draw particles trajectories. Another frequent application of USERDUMP is to perform a calculation in two steps, where the second step uses the collision file as a source. In principle it is also possible to use subroutine `MGDRAW` for scoring, for instance by interfacing it to some histogramming package, as it is customary in some other Monte Carlo programs. However, in general it is discouraged in FLUKA, unless the desired quantity cannot be scored via the standard FLUKA input commands, which is very rare. The FLUKA scoring options are indeed highly optimised and well checked against possible errors and artifacts. It is very unlikely that a user might be able to achieve in a short time the same level of reliability. In any case, user-written scoring via `MGDRAW` *must* be avoided in all runs where biasing is present, because to handle correctly the particle weights requires other FLUKA tools which are not available to the normal user.

Two more input options activating calls to user routines are USRICALL (7.79) and USROCALL (7.80). They allow the user to issue a call respectively to an initialisation routine `USRINI` (12.2.26) and to an output routine `USROUT` (12.2.28).

7.1.8 Miscellaneous

Option COMMENT (7.8) is not often used. Its main function is to insert a certain number of comment lines in the input file, but these are now most often replaced by lines with an asterisk in column 1. The same command can be used to override the default name (`fluka.stop`) of the "stopping file" which can terminate a FLUKA run before the requested number of histories is reached (see Note 2 on p. 199). In addition, it can communicate to the program information about the computer platform used, so that it will printed in output.

Much more important is command RANDOMIZe, which starts a new independent random number sequence. It can be omitted only in a first run, but it is compulsory if a sequence of independent runs is desired in order to calculate statistical errors.

Command STOP, inserted at any point in the input file, interrupts the reading. Any further input card is ignored. It may be made to follow PLOTGEOM and the corresponding input, so that the PLOTGEOM program is executed, but no FLUKA simulation is started.

Finally, command START is always needed to give the program the signal to begin the calculation. The command includes the number of primary histories to be simulated. A STOP command may follow, but it is not necessary since it is assumed to be present by default.

7.2 ASSIGNMAt

Defines the correspondence between region indices and material indices. It defines also regions with magnetic field.

See also MATERIAL, MGNFIELD

WHAT(1) = material index

WHAT(2) = lower bound of the region indices with material index equal to WHAT(1)
("*From region WHAT(2)...*")
Default = 2.0

WHAT(3) = upper bound of the region indices with material index equal to WHAT(1)
("*... to region WHAT(3)...*")
Default = WHAT(2)

WHAT(4) = step length in assigning indices
("*... in steps of WHAT(4)*")
Default = 1.0

WHAT(5) = 1.0: a magnetic field is present in the region(s) defined by WHAT(2), (3), and (4)
= 0.0: ignored
< 0.0: resets the default (no field) in the region(s) defined by WHAT(2), (3), and (4)
Default = 0.0 (ignored)

WHAT(6), SDUM: not used

Default : (option ASSIGNMAt not given): all regions are assigned material 1.0 (blackhole) except region 2.0 which is assumed to be COPPER (FLUKA pre-defined material 12.0). No magnetic field is the default for all regions.

Notes

1. Several ASSIGNMAt definitions are generally necessary to assign a material to all regions. Standard material names and their numbers are listed in Table 5.3 (p. 44. They may be redefined and others may be added.
2. Overlapping region indices can be given in several ASSIGNMAt definitions, each definition overriding the earlier ones. This makes the assigning of materials very convenient (see Example below).
3. In practice, option ASSIGNMAt must always be present. (If it is not chosen, all regions are considered to be black holes except region 2 which is assumed to be COPPER).
4. Magnetic field tracking is performed only in regions defined as magnetic field regions by WHAT(5) = 1.0. It is strongly recommended to define as such only regions where a magnetic field actually exists, due to the less efficient and less accurate tracking algorithm used in magnetic fields. To define a region as one with magnetic field and to return systematically B = 0.0 in that region via the user subroutine MAGFLD (p. 309) must be absolutely avoided (see MGNFIELD, p. 151).

Example:

```
*...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....
MATERIAL        13.0      27.0       2.7      10.0       0.0      0.0 ALUMINUM
ASSIGNMA        10.0       1.0      15.0       0.0       1.0      0.0
ASSIGNMA         2.0       5.0      17.0       6.0      -1.0      0.0
ASSIGNMA         2.0      16.0      18.0       2.0       0.0      0.0
*       The above definitions mean that all regions from 1 to 15 are
*       aluminium with a magnetic field, except regions 5 and 11 which are
*       vacuum without any magnetic field. Regions 16, 17 and 18 are also
*       vacuum without field.
*       Note that in the above example material 10 has been defined
*       overriding the pre-defined FLUKA aluminium material, but keeping
*       the same material number.
```

7.3 BEAM

Defines several beam characteristics: type of particle, energy, divergence, profile and statistical weight

See also BEAMPOS, SOURCE

WHAT(1) > 0.0: average beam momentum in GeV/c
< 0.0: average beam kinetic energy in GeV
This value can be overridden in user routine SOURCE (p. 314) by assigning a value to variable PBEAM
Default = 200.0 GeV/c (momentum)

WHAT(2) > 0.0: beam momentum spread in GeV/c. The momentum distribution is assumed to be rectangular
< 0.0: |WHAT(2)| is the full width at half maximum (FWHM) of a Gaussian momentum distribution (FWHM = 2.355 σ)
This value can be overridden in user routine SOURCE (p. 314) by assigning a value to variable DPBEAM
Default = 0.0

WHAT(3) specifies the beam divergence (in mrad):
> 0.0: |WHAT(3)| is the width of a rectangular angular distribution
< 0.0: |WHAT(3)| is the FWHM of a Gaussian angular distribution
> 2000π mrad (i.e., 2π rad): an isotropic distribution is assumed (see Note 7).
This value can be overridden in user routine SOURCE (p. 314) by assigning a value to variable DIVBM
Default = 0.0

WHAT(4) > 0.0: If WHAT(6) > 0.0, beam width in x-direction in cm. The beam profile is assumed to be rectangular.
If WHAT(6) < 0.0, WHAT(4) is the maximum radius of an annular beam spot
< 0.0: |WHAT(4)| is the FWHM of a Gaussian profile in x-direction (whatever the value of WHAT(6))
This value can be overridden in user routine SOURCE (p. 314) by assigning a value to variable XSPOT
Default = 0.0

WHAT(5) > 0.0: if WHAT(6) > 0.0, beam width in y-direction in cm. The beam profile is assumed to be rectangular.
If WHAT(6) < 0.0, WHAT(5) is the minimum radius of an annular beam spot
< 0.0: |WHAT(5) | is the FWHM of a gaussian profile in y-direction (whatever the value of WHAT(6))
This value can be overridden in user routine SOURCE (p. 314) by assigning a value to variable YSPOT
Default = WHAT(4)

|WHAT(6)| = weight of the beam particles.

If WHAT(6) $<$ 0.0, WHAT(4) and WHAT(5), if positive, are interpreted as the maximum and minimum radii of an annular beam spot. If negative, they are interpreted as FWHMs of gaussian profiles as explained above, independent of the value of WHAT(6)
This value can be overridden in user routine SOURCE (p. 314) by assigning a new weight to the beam particles (variable BEAWEI) or assigning a new value to the logical variable LBEAMC (LBEAMC = .TRUE. $\Longrightarrow$ annular beam).

Default = 1.0

SDUM = beam particle name. Particle names and numerical codes are listed in the table of FLUKA particle types (see Table 5.1 on p. 41).
For heavy ions, use the name HEAVYION and specify further the ion properties by means of option HI–PROPErt (p. 127). In this case WHAT(1) will mean the energy (or momentum) *per nucleon*, and not the total energy or momentum.
[**Not yet implemented:** For optical photons, use the name OPTIPHOT and specify further the transport properties by material by means of option OPT–PROP.]
This value can be overridden in user routine SOURCE (p. 314) by assigning a value to variable IJBEAM equal to the numerical code of the beam particle.

Default = PROTON

Default (option BEAM not given): all the above defaults apply, unless other values are provided by the user by means of a SOURCE subroutine (see Sec. 12.2.19).

Notes

1. Cases of distributed, non monoenergetic or other more complex sources should be treated by means of a user-written subroutine SOURCE as explained in the description of the SOURCE option (p. 314). In particular, the BEAM definition cannot handle beams of elliptical cross-section and rectangular profile. However, even when using a SOURCE subroutine, the momentum or kinetic energy defined by WHAT(1) of BEAM is meaningful, since it is taken as maximum energy for several cross-section tabulations and scoring facilities.
   Advice: when a user-written SOURCE is used, set WHAT(1) in BEAM equal to the maximum source particle momentum (or energy).
2. A two-dimensional distribution, gaussian with equal variances in x and y, results in a *radial* gaussian distribution with variance $\sigma_r = \sigma_x = \sigma_y$. The distribution has a form:
$$\mathrm{P}(r) = \frac{1}{2\pi\sigma_x\sigma_y}\,\mathrm{e}^{-\frac{1}{2}\left[\left(\frac{x}{\sigma_x}\right)^2+\left(\frac{y}{\sigma_y}\right)^2\right]} = \frac{1}{2\pi\sigma_r^2}\,\mathrm{e}^{-\frac{1}{2}\left(\frac{r}{\sigma_r}\right)^2}$$
3. All FLUKA results are normalised per unit incident particle weight. Thus, setting the starting weight to a fixed value different from 1.0 has no practical effect. A distribution of initial weights may be needed, however, when sampling from a non-monoenergetic spectrum: in this case, a SOURCE subroutine must be written (see Sec. 12.2.19).
4. All options governed by WHAT(3)–WHAT(5) are meaningful only if the beam direction is along the positive z-axis, unless a command BEAMAXES is issued to establish a beam reference frame different from the geometry frame (see p. 7.4). If the beam is not in the z direction and no BEAMAXES command has been given, WHAT(3)–WHAT(5) must be set = 0.0 (unpredictable effects would arise otherwise).
5. The beam momentum value as defined with the BEAM card is available to user routines as a variable PBEAM and so is the beam particle type IJBEAM. These variables, as well as those defining other beam properties, are in COMMON BEAMCM which can be accessed with the INCLUDE file (BEAMCM).
6. It is possible to track pseudoparticles by setting SDUM = RAY. See Chap. 14 for details.
7. When an isotropic source is defined (by setting WHAT(3) $> 2000\pi$), any cosines defined by option BEAMPOS become meaningless, but their values are still reported on standard output.

Example:

```
*         The following BEAM card refers to a 100 keV pencil-like
*         electron beam:
*...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+...
BEAM          -1.E-4       0.0       0.0       0.0       0.0      1.0 ELECTRON
*         The next option card describes a parallel proton beam with a
*         momentum of 10.0 +/- 0.2 GeV/c, with a gaussian profile in
*         the x-direction and in the y-direction described by standard
*         deviations sigma_x = 1. cm (FWHM = 2.36 cm) and sigma_y = 0.5
*         cm (FWHM = 1.18 cm).
*...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+...
BEAM            10.0       0.2       0.0     -2.36     -1.18      1.0 PROTON
*         The next example concerns a negative muon beam of 2 GeV
*         kinetic energy, with a divergence of 3 mrad.
*...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+...
BEAM            -2.0       0.0       3.0       0.0       0.0      1.0 MUON-
*         The next BEAM card describes a 137-Cs isotropic source
BEAM       -661.7E-6       0.0      1.E4       0.0       0.0      1.0 PHOTON
*         The last example illustrates how to define a hollow 14 MeV
*         neutron beam, with an inner radius of 7 mm and an outer radius
*         of 1.2 cm.
*...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+...
BEAM         -14.E-3       0.0       0.0       1.2       0.7     -1.0 NEUTRON
```

7.4 BEAMAXES

Defines the axes used for a beam reference frame different from the geometry frame

See also BEAM, BEAMPOS, POLARIZAti, SOURCE

WHAT(1) = cosine of the angle between the x-axis of the beam reference frame and the x-axis of the geometry frame
Default : no default

WHAT(2) = cosine of the angle between the x-axis of the beam reference frame and the y-axis of the geometry frame
Default : no default

WHAT(3) = cosine of the angle between the x-axis of the beam reference frame and the z-axis of the geometry frame
Default : no default

WHAT(4) = cosine of the angle between the z-axis of the beam reference frame and the x-axis of the geometry frame
Default : no default

WHAT(5) = cosine of the angle between the z-axis of the beam reference frame and the y-axis of the geometry frame
Default : no default

WHAT(6) = cosine of the angle between the z-axis of the beam reference frame and the z-axis of the geometry frame
Default : no default

SDUM : not used

Default (option BEAMAXES not given): the beam frame coincides with the geometry frame

Notes

1. Option BEAM (p. 64) describes a simple pencil beam, or also a beam simply distributed in space (angular divergence and transversal profile), provided the beam axis coincides with the z-axis of the input geometry. Also a possible beam polarisation described by option POLARIZAti (p. 182) refers to a beam with its axis coinciding with the geometry z-axis.
   The purpose of option BEAMAXES is to allow the user to define angular divergence, transversal profile and polarisation for a beam of arbitrary direction, either constant as defined by option BEAMPOS, or not necessarily known in advance as provided by a user `SOURCE` routine (12.2.19). For this purpose, the user can define divergence, profile and polarisation in a beam reference frame. Option BEAMAXES establishes the correspondence between beam and geometry reference frame.
2. The origin of the beam reference frame coincides always with that of the geometry frame
3. The user needs to input only the direction cosines of the x- and of the z-axis of the beam frame. The direction of the y-axis is determined by the program as the vector product $\vec{z} \times \vec{x}$
4. If the the x- and z-axes defined with BEAMAXES are not exactly perpendicular (in double precision!) the program forces perpendicularity by adjusting the cosines of the x-axis

5. The direction cosines of the x- and z-axes do not need to be exactly normalised to 1. The code takes care of properly normalising all cosines

Example:

```
*  The next option cards describe a 10 GeV proton beam with a divergence of
*  50 mrad and a gaussian profile in the "beam x"-direction and in the
*  "beam y"-direction described by standard deviations sigma_x = 1. cm
*  (FWHM = 2.36 cm) and sigma_y = 0.5 cm (FWHM = 1.18 cm). The beam starts
*  from point (0,0,0) and is directed in a direction perpendicular to the
*  "geometry x" axis, at 45 degrees with respect to both "geometry y" and
*  "geometry z". The "beam x" axis has cosines 1,0,0 and the "beam z"
*  axis has cosines 0, cos(pi/4), cos(pi/4)
*...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+...
BEAM           -10.0       0.0      50.0     -2.36     -1.18      1.0 PROTON
BEAMPOS          0.0       0.0       0.0       0.0 0.7071068       0.0
BEAMAXES         1.0       0.0       0.0       0.0 0.7071068 0.7071068
```

7.5 BEAMPOS

Defines the coordinates of the centre of the beam spot (i.e., the point from which transport starts) and the beam direction

See also BEAM, SOURCE

WHAT(1) = x-coordinate of the spot centre.
Can be overridden in user routine `SOURCE` (p. 314) by assigning a value to variable `XBEAM`
Default = `0.0`

WHAT(2) = y-coordinate of the spot centre.
Can be overridden in user routine `SOURCE` (p. 314) by assigning a value to variable `YBEAM`
Default = `0.0`

WHAT(3) = z-coordinate of the spot centre.
Can be overridden in user routine `SOURCE` (p. 314) by assigning a value to variable `ZBEAM`
Default = `0.0`

WHAT(4) = direction cosine of the beam with respect to the x-axis.
Can be overridden in user routine `SOURCE` (p. 314) by assigning a value to variable `UBEAM`
Default = `0.0`

WHAT(5) = direction cosine of the beam with respect to the y-axis.
Can be overridden in user routine `SOURCE` (p. 314) by assigning a value to variable `VBEAM`
Default = `0.0`

WHAT(6) : not used

SDUM = NEGATIVE means that the direction cosine with respect to z-axis is negative.
The value of the direction cosine with respect to the z-axis can be overridden in user routine `SOURCE` (p. 314) by assigning a value to variable `WBEAM` *(make sure that the three cosines are properly normalised so that the sum of their squares is 1.0!)*
Default : beam directed in the positive z-direction

Default (option BEAMPOS not given): all the above defaults apply (the beam starts at point `0.0, 0.0, 0.0` in the z-direction)

Notes

1. To take full advantage of some tracking optimisation features, it is often a good idea to create a buffer vacuum region containing the whole geometry, which must itself be contained within the external (mandatory) black hole region. It is then suggested that the beam impact point be chosen in vacuum, slightly upstream of the actual one on a material boundary. As a general rule, anyway, it is recommended to never select the impact point *exactly* on a boundary.
2. The beam spot coordinates and the beam director cosines as defined with the BEAMPOS card are available to user routines with names `XBEAM, YBEAM, ZBEAM` and `UBEAM, VBEAM, WBEAM` respectively. These variables, as well as those defining other beam properties, are in `COMMON BEAMCM` which can be accessed with the `INCLUDE` file `(BEAMCM)`.

3. Beam divergence and transversal profile defined by option BEAM (p. 7.3), as well as polarisation defined by option POLARIZAti (p. 7.57), are meaningful only if the beam direction is along the positive z-axis, unless a command BEAMAXES is issued to establish a beam reference frame different from the geometry frame (see p. 7.4).

Examples:

```
*         A beam parallel to the x-axis starting at a point of
*         coordinates -0.1, 5.0, 5.0 :
*...+....1....+....2....+....3....+....4....+....5....+....6....+....7...
BEAMPOS         -0.1       5.0       5.0       1.0       0.0       0.0

*         A beam perpendicular to the x-axis, with director cosines
*         0., 1/sqrt(2), -1/sqrt(2) with respect to x, y and z,
*         starting at point 0.0, 0.0, 0.0 :
*...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+...
BEAMPOS          0.0       0.0       0.0       0.0 0.7071068      0.0 NEGATIVE
```

7.6 BIASING

Biases the multiplicity of secondaries (only for hadron or muon/photon photonuclear interactions) on a region by region basis.
Sets importance sampling (Russian Roulette/splitting) at boundary crossing by region and by particle.

See also EMF–BIAS, LOW–BIAS, LAM–BIAS, WW–FACTOr, WW–PROFIle, WW–THRESh

The meaning of WHAT(1)...WHAT(6) and SDUM is different depending on the sign of WHAT(1):

If **WHAT(1)** $\geq$ `0.0`:

WHAT(1) specifies the particles to be biased:
= `0.0`: all particles
= `1.0`: hadrons and muons
= `2.0`: electrons, positrons and photons
= `3.0`: low-energy neutrons

WHAT(2) = RR (or splitting) factor by which the average number of secondaries produced in a collision should be reduced (or increased). Meaningful only for hadron or muon/photon photonuclear interactions.
This value can be overridden in the user routine `UBSSET` (p. 320) by assigning a value to variable `RRHADR`.
Default = `1.0`

WHAT(3) = region importance (allowed values range from `0.0001` to `10000.0`)
This value can be overridden in the user routine `UBSSET` (p. 320) by assigning a value to one or more of the variables `IMPHAD`, `IMPLOW` and `IMPEMF` (depending on the value of WHAT(1))
Default = `1.0`

WHAT(4) = lower bound of the region indices with importance equal to WHAT(3) and/or with multiplicity biasing factor equal to WHAT(2)
("*From region WHAT(4)...*")
Default = `2.0`

WHAT(5) = upper bound of the region indices with importance equal to WHAT(3) and/or with RR factor equal to WHAT(2)
("*...to region WHAT(5)...*")
Default = WHAT(4)

WHAT(6) = step length in assigning indices
("*...in steps of WHAT(6)*")
Default = `1.0`

SDUM = PRINT : importance biasing counters are printed (useful to tune importances and weight windows)
= NOPRINT : counters are not printed (cancels any previous PRINT request)
= USER : importance biasing according to the user defined routine `USIMBS` (p. 322)
= NOUSER : reset to default (cancels any previous USER request)
= RRPRONLY : multiplicity biasing for primary particles only
blank : ignored
Default = NOPRINT, NOUSER, multiplicity biasing for all generations (if requested)

If **WHAT(1)** $<$ 0.0:

WHAT(1) : flag indicating that all region importances shall be modified by a particle-dependent factor, based on a modifying parameter as explained in the Note 3 below

WHAT(2) $\geq$ 0.0: modifying parameter **M** (see Note 3). See also WARNING below.

$<$ 0.0: **M** is reset to the default value 1.0 (i.e., no modification)

WHAT(3) = lower bound of the particle numbers to which the indicated modifying parameter applies ("*From particle WHAT(3)...*")
Default = 1.0

WHAT(4) = upper bound of the particle numbers to which the indicated modifying parameter applies ("*... to particle WHAT(4)...*")
Default = WHAT(3) if WHAT(3) $<$ 0.0, all particles otherwise

WHAT(5) = step length in assigning particle numbers ("*... in steps of WHAT(5)*")
Default = 1.0

WHAT(6) : not used

SDUM = PRIMARY: importance biasing is applied also to primary particles (cancels any previous NOPRIMARy request)
= NOPRIMARy: importance biasing is applied only to secondaries
Default = PRIMARY

WARNING

Even if a BIASING card is issued only to set PRIMARY/NOPRIMARy, remember that a value of 0.0 is meaningful for WHAT(2). Leaving blank WHAT(2) to WHAT(5) has the effect of turning off all importance biasing for all particles!

Default : (option BIASING not given): no multiplicity or RR/splitting biasing

Notes

1. WHAT(2), with WHAT(1) $\geq$ 0.0, governs the application of Russian Roulette (or splitting) at hadronic collisions, in order to achieve a reduction (resp. an increase) of the multiplicity of secondaries.
   The same secondary is loaded onto the particle stack for further transport 0, 1 or any number of times depending on a random choice, such that *on average* the requested multiplicity reduction (or increase) is achieved. The weight of the stacked particles is automatically adjusted in order to account for the bias thus introduced.
   If Russian Roulette has been requested, the reduction will not affect the leading particle, which will always be retained, with unmodified weight. Also, no RR is performed when the number of secondaries is less than 3. On the contrary, there are no such limitations for splitting (multiplicity increase).
   There is some analogy with leading particle biasing as performed for electrons and photons with option EMF–BIAS, and for hadrons in codes like CASIM [172].
2. WHAT(3), with WHAT(1) $\geq$ 0.0, governs RR/splitting at boundary crossing. The number of particles of the selected type crossing a given boundary is reduced/increased on average by a factor equal to the ratio of the importances on either side of the boundary. What is relevant are the relative importances of adjacent regions, not their absolute values. As a guideline, in shielding and, in general, strong attenuation problems, the importance of a region should be about inversely proportional to the corresponding attenuation factor

(absorption plus distance attenuation). This would exactly compensate the dilution of particle density leading to a particle population approximately uniform in space. In some cases, however, when the user is interested in improving statistics only in a limited portion of space, a uniform population density is not desirable, but it is convenient to set importances so as to increase particle densities in a particular direction.

3. Different importances can be given to the same region for different particles, using the particle-dependent modifying factor **M** which can be defined setting WHAT(1) > `0.0`.
   The modifying parameter **M** (WHAT(2), with WHAT(1) > `0.0`) works as follows:
   At a boundary crossing, let us call $\mathbf{I_1}$ the importance of the upstream region, and $\mathbf{I_2}$ that of the downstream region.
   - If $\mathbf{I_2} < \mathbf{I_1}$, Russian Roulette will be played.
     Without any modifying factor, the chance of particle survival is $\mathbf{I_2/I_1}$.
     For $\mathbf{0.0} \leq \mathbf{M} \leq \mathbf{1.0}$, the survival chance is modified to:

     $$\mathbf{1.0 - M \times (1.0 - I_2/I_1)}$$

     It can be seen that a value $\mathbf{M} = $ `0.0` resets the chance of survival to `1.0`, namely *inhibits Russian Roulette* biasing.
     A value $\mathbf{M} = $ `1.0` leaves the survival chance *unmodified*, while any value between `0.0` and `1.0` *increases* the probability of survival with respect to the basic setting.
     For $\mathbf{M} \geq \mathbf{1.0}$, the survival chance is modified to:

     $$\mathbf{I_2/(M \times I_1)}$$

     So, a value larger than `1.0` *decreases* the probability of survival with respect to the basic setting.
   - If $\mathbf{I_2} > \mathbf{I_1}$, there will be splitting.
     Without any modifying factor, the number of particles is increased on average by a factor $\mathbf{I_2/I_1}$.
     With the modifying factor, the number of particles is increased instead by:

     $$\mathbf{1.0 + M \times (I_2/I_1 - 1.0)}$$

     It can be seen that a value $\mathbf{M} = $ `0.0` resets the splitting factor to `1.0`, namely *inhibits splitting*.
     A value $\mathbf{M} = $ `1.0` leaves the number of particles em unmodified; a value between `0.0` and `1.0` *decreases* the amount of splitting with respect to the basic setting; a value > `1.0` *increases* the amount of splitting.

   Hint: One of the most common uses of the modifying factor is to play Russian Roulette/splitting only for some selected particles: one does that by inhibiting biasing for all other particles, i.e., setting = `0.0` the modifying factor **M** (WHAT(2), with WHAT(1) < `0.0`).
4. In the most general case, increasing a region's importance leads to an increased particle "traffic" through that region and consequently to a better scoring statistics in regions "beyond". However, it should be avoided to have relatively large importances in scoring regions compared with those in adjacent ones to avoid correlated tallies. If that happens, the scoring statistics might look only apparently good. It must be avoided also to have too different importances in adjacent zones: the best biasing has to be done gently, without forcing and in a way as continuous as possible.
5. All these biasing techniques are intended to improve statistics in some parts of phase space *at the expenses of the other parts*. Biased runs in particular can neither accelerate convergence in all regions, nor reproduce natural fluctuations and correlations. **Do not bias unless you know what you are doing!**
6. **Advice:** When choosing the multiplicity reduction option of BIASING, or any other biasing option which can introduce weight fluctuations in a given region, it is suggested to set also a weight window (cards WW–FACTOr, p. 244 and WW–THRESh, p. 249) in order to avoid too large fluctuations in weight. The window must be consistent with the other weight-modifying options, i.e. it must be approximately centred on the average value of the weight expected in the region in question. If necessary, set SDUM = PRINT to get such information.
   In case no window has been set, the code still keeps weights under control (but only those of low-energy neutrons) by imposing a maximum deviation from a central value. This reference level is usually equal to the inverse of the neutron importance in the region in question. However, since for technical reasons in FLUKA allowed importance values range only from `0.0001` to `10000.0`, the user can multiply all the importances by a factor, *only for the purpose of calculating the reference weight level*, by means of option WW–PROFIle (p. 247).
   If the only biasing is via region importances set by WHAT(3), only limited fluctuations arise (all particles of a given kind have about the same weight in the same region), and no window is needed.

Example:

```
*...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+...
BIASING          2.0       0.0      10.0       7.0      11.0      2.0
BIASING          2.0       0.0      15.0       8.0       9.0      0.0
BIASING         -1.0       0.0       3.0       4.0       0.0      0.0
BIASING          1.0       0.7       0.4       3.0       8.0      0.0 PRINT
*         In this example, the first two BIASING cards set an importance = 10
*         for electrons, positrons and photons in regions 7, 9 and 11; and
*         an importance = 15 in regions 8 and 9 for the same particles.
*         However, the following card requires a modifying factor = 0.0
*         (no splitting or Russian Roulette) for electrons and positrons.
*         The net result is that biasing at boundary crossing with the above
*         region importances is played only for photons.
*         The fourth card sets a reduction factor = 0.7 for the multiplicity
*         of hadronic events in regions 3, 4, 5, 6, 7 and 8; the importance
*         of these same regions is set = 0.4; and it is required that biasing
*         counters be printed.
```

7.7 BME

Not yet implemented. Prepared for the BME *event generator.*

7.8 COMMENT

Inserts comments in the input. It can also be used to inform the program about the input filename, the computer system or the computer model

See also TITLE

WHAT(1) = number of comment cards following this control card
Default (WHAT(1) and WHAT(2) $\leq$ 0.0 and SDUM = blank): 1 comment card will follow

WHAT(2) = computer system (see Note 2)
= 1: VAX–VMS
= 2: *IBM–VM (CMS) (not supported any longer)*
= 3: *IBM–MVS (not supported)*
= 4: *CRAY (not supported)*
= 5: *UNIX (IBM–AIX) (not supported any longer)*
= 6: UNIX (HP–UX)
= 7: UNIX (SUN)
= 8: Digital UNIX
= 9: LINUX (PC IBM compatible)

WHAT(3) – **WHAT(6)**: not used

SDUM = INPUT: next card must give the input filename, without extension/filetype. An extension .inp is assumed for most systems (but for VAX this option is useless)
= MODEL: next card must give the computer model (not used on VAX, where the model is established directly by FLUKA)
Default = blank (next card is just an ordinary comment).
If SDUM = INPUT is not given, the input file name is assumed to be fluka.
If SDUM = MODEL is not given, the name of the model is left blank.

Default (option COMMENT not given): no comment

Notes

1. An alternative way to insert comments, valid also for Combinatorial Geometry input (see Chap. 8) is to start input cards with a * character. Each line having a * in column 1 is treated as a comment and ignored (but printed on output). Short comments can also be given at the end of any option card, preceded by an exclamation mark. Any character after the character ! will be ignored (this facility is mainly used with free-form input).
2. The information about the computer system, needed by the code in order to carry out a few platform-dependent tasks, is generally hard-wired by initialisation of variable KOMPUT in block data BDINPT. However, WHAT(2) allows to override such information in case of need, but the corresponding COMMENT card must be issued early in input, in particular before any OPEN command. At present, only WHAT(2) = 1,6,7,8 and 9 are supported.
3. Information about the computer model, if available, is printed in the main output together with the CPU time used, and may be useful in order to compare similar runs carried out on different machines.
4. Information about the name of the input file is used to generate the name of a file which stops the program (see note to option START). On UNIX, if the input file is called xxx.inp, the stopping file must be called xxx.stop (fluka.stop if the information is not provided). On VAX, however, the name of the stopping file is always FLUKA$STOP.
5. COMMENT cards with SDUM = INPUT or MODEL, generated by a script (command file), may be used for those systems where the corresponding information can be accessed by a script but not directly by a Fortran program.

Examples:

```
*...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+...
COMMENT          3.0
This comment is made of three lines. This is the first line
and this is the second one
and here is the third.
COMMENT
This is a comment consisting of a single line, equivalent to:
* This is a comment consisting of a single line

* The following cards indicate that the run, based on input file myshower.inp,
* is executed on a Pentium-4 computer at 2.4 GHz
COMMENT           0.        5.                                        MODEL
Pentium4 2.4 GHz
COMMENT           0.                                                  INPUT
myshower
```

7.9 COMPOUND

Defines a compound or mixture, made of several materials, or even a mixture of different isotopes

See also LOW–MAT, MATERIAL, MAT–PROP

If **WHAT(1)** > 0.0 and **WHAT(2)** > 0.0:

WHAT(1) = atom relative content of first material in the compound
WHAT(2) = index of first material

If **WHAT(1)** < 0.0 and **WHAT(2)** > 0.0:

|**WHAT(1)**| = mass fraction of first material in the compound
WHAT(2) = index of first material

If **WHAT(1)** < 0.0 and **WHAT(2)** < 0.0:

|**WHAT(1)**| = volume fraction of first material in the compound
|**WHAT(2)**| = index of first material

No default

In a similar way, WHAT(3) and WHAT(4) refer to the second material in the compound, WHAT(5) and WHAT(6) to the third one.

SDUM : name of the compound

Default (option COMPOUND not given): no compound is defined

For more than three materials in the same compound, add as many COMPOUND cards with the same SDUM name as needed (but the maximum number of components per compound is 80, and the maximum total number of components is 700).

Notes

1. Option COMPOUND must always be used in conjunction with a MATERIAL card having the same SDUM name (see MATERIAL, p. 142). MATERIAL cards used for this purpose provide the density of the compound, its material number and name (WHAT(1) and WHAT(2) of the MATERIAL option, namely atomic and mass number, are ignored).
2. The order of MATERIAL and COMPOUND cards is irrelevant.
3. The atom (or molecule) content, mass fraction or volume fraction need only to be given on a relative basis (normalisation is done automatically by the program).
4. Partial pressures of an (ideal) gas are equivalent to molecule fractions and also to volume fractions.
5. If a compound is defined by volume fractions of the components (either elements or compounds themselves — see Note 8 for recursive definitions), FLUKA internally calculates the atomic densities of each component using the densities indicated in the respective MATERIAL cards: in this case, therefore, (and only in this case) it is important that these correspond to the actual densities.
6. Isotopic compositions other than natural can be defined by the COMPOUND option too.
7. When using the LOW–NEUT option (explicitly or by default set by the DEFAULTS option), a special data set containing low-energy neutron cross-sections for each material used must be available. The data sets are combined in a single file, delivered with the FLUKA program (logical input unit 9, see Chap. 3). Each low-energy

neutron data set is identified either by name (if equal to a FLUKA name and unique or first with that name), or/and by one or more identifiers given with a card LOW–MAT (p. 138) when necessary to remove ambiguity. In the case of a composite material defined by a COMPOUND option, two possibilities are allowed (see LOW–MAT):

(a) to associate the FLUKA material with a pre-mixed neutron data set. In this case interactions take place with individual nuclei at high energy, while average cross-sections are used for low-energy neutrons. Note that no pre-mixed neutron data set is yet available (at the moment the standard sets contain pure elements or isotopes only).

(b) to associate the FLUKA material with several elemental neutron data sets (one per component element). In this case both high-energy and low-energy neutron interactions take place with individual nuclei. This is the only possibility at present but it may change in future.

8. Recursion is allowed, i.e., the components of a composite material can be composite materials. The depth of recursion is only limited by the size of the internal arrays (in case of overflow a message is issued and the job is terminated). Different levels of recursion can use different units in the definition of the component fractions (atoms, mass or volume fractions). Note, however, that if a compound is put together from different composite molecules, the atomic and molecular fractions have to be given without normalisation (use the chemical formulae directly).
What follows is an example for a simple compound BOOZE containing 50% in weight of water and 50% of ethanol.

```
*...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+...
MATERIAL         1.0       1.0  .0000899       3.0       0.0       0.0 HYDROGEN
MATERIAL         6.0      12.0       2.0       4.0       0.0       0.0 CARBON
MATERIAL         8.0      16.0   0.00143       5.0       0.0       0.0 OXYGEN
MATERIAL         0.0       0.0       1.0      20.0       0.0       0.0 WATER
MATERIAL         0.0       0.0    0.7907       7.0       0.0       0.0 ETHANOL
MATERIAL         0.0       0.0    0.9155       8.0       0.0       0.0 BOOZE
COMPOUND         2.0       3.0       1.0       5.0       0.0       0.0 WATER
COMPOUND         2.0       4.0       6.0       3.0       1.0       5.0 ETHANOL
COMPOUND       -50.0      20.0     -50.0       7.0       0.0       0.0 BOOZE
*   Note that in the above example materials 4, 5, 7 and 8 have been defined
*   overriding the default FLUKA material numbers.
```

Example of how COMPOUND is commonly used to define a mixture (concrete):

```
*...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+...
* definition of material 27 (concrete) as compound:  H (1%), C(0.1%),
* O(52.9107%), Na(1.6%), Mg(0.2%), Al(3.3872%), Si(33.7021%), K(1.3%),
* Ca(4.4%), Fe(1.4%)
MATERIAL        19.0   39.0983     0.862      26.0       0.0       0.0 POTASSIU
MATERIAL         0.0       0.0      2.35      27.0       0.0        0. CONCRETE
COMPOUND       -0.01       3.0    -0.001       6.0 -0.529107        8. CONCRETE
COMPOUND      -0.016      19.0    -0.002       9.0 -0.033872       10. CONCRETE
COMPOUND   -0.337021      14.0    -0.013      26.0    -0.044       21. CONCRETE
COMPOUND      -0.014      11.0
*   In the above example, elements 3 (hydrogen), 6 (carbon), 8 (oxygen),
*   9 (magnesium), 10 (aluminium), 11 (iron), 14 (silicon), 19 (sodium) and
*   21 (calcium) are not defined because the corresponding pre-defined FLUKA
*   materials are used. Potassium is not pre-defined, therefore it is
*   assigned a new numbers 26 (that keeps the numbering sequence continuous,
*   since the last FLUKA pre-defined material has number 25). The name is
*   chosen to correspond with the potassium neutron cross-section data set.
```

More complex uses of COMPOUND in connection with MATERIAL and LOW–MAT are illustrated in examples 15.1.

7.10 CORRFACT

allows to alter material density for dE/dx and nuclear processes on a region-by-region basis

(see also ASSIGNMAt, COMPOUND, LOW–MAT, MATERIAL, MAT–PROP)

WHAT(1) $\geq$ `0.0`: density scaling factor for charged particle ionization processes (dE/dx, delta ray production, Möller and Bhabha scattering)

$=$ `0.0`: ignored

$<$ `0.0`: reset to default

Default : `1.0`

WHAT(2) $\geq$ `0.0`: density scaling factor for all other processes

$=$ `0.0`: ignored

$<$ `0.0`: reset to default

Default : `1.0`

WHAT(3) : not used

WHAT(4) : lower index bound of regions where the scaling factors shall apply
("*From region WHAT(4)...*")
Default : `2.0`

WHAT(5) : upper index bound of regions where the scaling factors shall apply
("*... to region WHAT(5)...*")
Default = WHAT(4)

WHAT(6) = step length in assigning region numbers
("*... in steps of WHAT(6)*")
Default = `1.0`

SDUM : not used

Default (option CORRFACT not given): no density scaling factors are applied

Notes

1. Option CORRFACT is mainly used in connection with voxel geometries derived from a CT scan, where particle transport is done often in an equivalent material (e.g., water), but accounting for the density variations provided by scan at the voxel level. While this approach is reliable for what concerns ionisation, other reactions, which do no scale with density, must be simulated for the actual material composition.

   Example:

```
*...+....1....+....2....+....3....+....4....+....5....+....6....+....7...
* Multiply density by  a 0.85 factor for what concerns atomic processes
* in regions 7, 8, 9, 10, 11, 12
CORRFACT       0.85       0.0       0.0       7.0       12.
```

7.11 DCYSCORE

Associates selected scoring detectors of given estimator type with user-defined decay times. See **warning** *in Note 1 below*

See also DCYTIMES, IRRPROFIle, RADDECAY, RESNUCLEi

WHAT(1) : cooling time index to be associated with the detector(s) of estimator type defined by SDUM and with number corresponding to WHAT(4)–WHAT(6) (see Note 2 below)
Default = 0.0: no scoring!

WHAT(2) – **WHAT(3)**: not used

WHAT(4) : lower index bound of detectors associated with the specified cooling time ("*From detector of estimator type SDUM no. WHAT(4)...*")
Default = 1.0

WHAT(5) : upper index bound of detectors associated with the specified cooling time ("*...to detector of estimator type SDUM no. WHAT(5)...*")
Default = WHAT(4)

WHAT(6) = step length in assigning indices ("*...in steps of WHAT(6)*")
Default = 1.0

SDUM : identifies the kind of estimator under consideration: EVENTBIN, RESNUCLEi, USRBDX, USRBIN, USRCOLL, USRTRACK, USRYIELD
Default : no default!

Default (option DCYSCORE not given): no detector is associated with any cooling time index

Notes

1. **Warning:** when the DCYSCORE option is applied to a detector, all quantities are expressed **per unit time**. For instance, the RESNUCLEi estimator will output **Bq**, dose estimators will provide dose rate etc..
2. The cooling time index indicated by WHAT(1) must be one of those defined by means of option DCYTIMES (p. 82)
3. USRBIN and EVENTBIN detectors are counted together (they belong to the same index sequence), and so are USRTRACK and USRCOLL

Example:

```
*...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+...
DCYTIMES      86400.   172800.
DCYSCORE          1.        0.        0.        4.        8.        2.USRBIN
DCYSCORE          2.        0.        0.        1.        5.          USRTRACK
* Two cooling times have been defined. Binnings no. 4, 6 and 8 will be
* associated to the first cooling time (one day), and track-length detectors
* no. 1, 2, 3, 4 and 5 will be associated to the second cooling time (2 days)
```

7.12 DCYTIMES

defines decay times for radioactive product scoring

See also DCYSCORE, IRRPROFIle, RADDECAY, RESNUCLEi

Option DCYTIMES defines decay times after irradiations at which selected quantities (for instance residual dose) are scored.

WHAT(1) : cooling time (in s) after the irradiation end, to be associated to a scoring detector (see Note 1 below)

`>=<  0.0` : a new decay time is added with delay WHAT(1)

Default :scoring at end of irradiation is associated with index `1`

WHAT(2) : the same as WHAT(1) (one more cooling time)

WHAT(3) : the same as WHAT(1) (one more cooling time)

WHAT(4) : the same as WHAT(1) (one more cooling time)

WHAT(5) : the same as WHAT(1) (one more cooling time)

WHAT(6) : the same as WHAT(1) (one more cooling time)

SDUM : not used

Default (option DCYTIMES not given): no decay times are defined

Notes

1. Each cooling time is assigned an index, following the order in which it has been input. This index can be used in option DCYSCORE to assign that particular cooling time to one or more scoring detectors.
2. Multiple cards can be given, up to a maximum of 20 decay times. All decay times are counted from the end of the last irradiation period as defined by the IRRPROFI command. A null decay time activates scoring exactly at the end of irradiation. A negative decay time is admitted: scoring is performed at the chosen time "during irradiation"

Example:

```
*...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
DCYTIMES         10.       30.     3600.    43200.    86400.   172800.
DCYTIMES    2592000. 31557600.
* Eight different cooling times have been defined, each with an index
* corresponding to its input order: cooling time no. 1 is 10 s, no. 2 is 30 s,
* and those from no. 3 to 8 are respectively 1 h, 1/2 d, 1 d, 2 d, 30 d, 1 y
```

7.13 DEFAULTS

Sets FLUKA *defaults suitable for a specified kind of problems. Starting from* FLUKA99.5 *(June 2000) the standard defaults are those described under* NEW–DEFAults *below. That is, if no* DEFAULTS *card is issued the code behaves as if a card with* NEW–DEFAults *was given. be given.*

See also GLOBAL

Important: DEFAULTS declarations, if present, must precede any executable option

WHAT(1) – **WHAT(6)**: not used

SDUM = CALORIMEtry : defaults for calorimeter simulations
EET/TRANsmut : defaults for Energy Transformer or transmutation calculations
EM–CASCAde : defaults for pure EM cascades
ICARUS : defaults for studies related to the ICARUS experiment
HADROTHErapy : defaults for hadrotherapy calculations
NEUTRONS : defaults for pure low-energy neutron runs
NEW–DEFAults : reasonable minimal set of new defaults — not needed
PRECISIOn : defaults for precision simulations
SHIELDINg : defaults for shielding calculations

Default : it is not allowed to leave SDUM blank. If the DEFAULT card is missing, the standard defaults are unchanged (equivalent to setting SDUM = NEW–DEFAults)

Defaults changed by the various options

CALORIMEtry

- EMF on (no need for an EMF card)
- Rayleigh scattering and inelastic form factor corrections to Compton scattering activated (no EMFRAY needed)
- Detailed photoelectric edge treatment and fluorescence photons activated (no EMFFLUO needed)
- Low-energy neutron transport on (no LOW–NEUT needed) (high energy neutron threshold at 19.6 MeV)
- Fully analogue absorption for low-energy neutrons
- Particle transport threshold set at $1 \times m_{part}/m_{prot}$ MeV, except neutrons (19.6 MeV) and (anti)neutrinos (0, but they are discarded by default anyway)
- Multiple scattering threshold at minimum allowed energy both for primary and secondary charged particles
- Delta ray production on with threshold 100 keV (see option DELTARAY)
- Restricted ionisation fluctuations on, for both hadrons/muons and EM particles (see option IONFLUCT)
- Fraction of the kinetic energy to be lost in a step set at 0.08, number of dp/dx tabulation points set at 80 (see options DELTARAY, FLUKAFIX)
- Heavy particle e^+e^- pair production activated with full explicit production (with the minimum threshold $= 2\ m_e$)

- Heavy particle bremsstrahlung activated with explicit photon production above 300 keV
- Muon photonuclear interactions activated with explicit generation of secondaries
- Heavy fragment transport activated

EET/TRANsmut

- Low-energy neutron transport on (high energy neutron threshold at 19.6 MeV)
- Non analogue absorption for low-energy neutrons with probability 0.95 for the last (thermal) group
- Particle transport threshold set at 1 MeV, except neutrons (19.6 MeV) and (anti)neutrinos (0, but they are discarded by default anyway)
- Multiple scattering threshold for primary and secondary charged particles lowered to 10 and 20 MeV respectively
- Unrestricted ionisation fluctuations on both for hadrons/muons and EM particles (if requested) (see option IONFLUCT)
- Both explicit and continuous heavy particle bremsstrahlung and pair production inhibited

EM–CASCAde

- Electromagnetic interactions on (no need for explicit option EMF)
- Rayleigh scattering and inelastic form factor corrections to Compton scattering activated (no EMFRAY needed)
- Detailed photoelectric edge treatment and fluorescence photons activated (no EMFFLUO needed)
- Restricted ionisation fluctuations for EM particles (see option IONFLUCT)
- Both explicit and continuous heavy particle bremsstrahlung and pair production inhibited

ICARUS

- EMF on
- Rayleigh scattering and inelastic form factor corrections to Compton scattering activated (no EMFRAY needed)
- Detailed photoelectric edge treatment and fluorescence photons activated (no EMFFLUO needed)
- Low-energy neutron transport on (high energy neutron threshold at 19.6 MeV)
- Fully analogue absorption for low-energy neutrons
- Particle transport threshold set at 100 keV, except neutrons (19.6 MeV) and (anti)neutrinos (0, but they are discarded by default anyway)
- Multiple scattering threshold at minimum allowed energy both for primary and secondary charged particles
- Delta ray production on with threshold 100 keV (see option DELTARAY)
- Restricted ionisation fluctuations on both for hadrons/muons and EM particles (see option IONFLUCT)
- Tabulation ratio for hadron/muon dp/dx set at 1.04, fraction of the kinetic energy to be lost in a step set at 0.05, number of dp/dx tabulation points set at 80 (see options DELTARAY, FLUKAFIX)
- Heavy particle e^+e^- pair production activated with full explicit production (with the minimum threshold $= 2\ m_e$)
- Heavy particle bremsstrahlung activated with explicit photon production above 300 keV

- Muon photonuclear interactions activated with explicit generation of secondaries
- Heavy fragment transport activated

HADROTHErapy

- EMF on
- Inelastic form factor corrections to Compton scattering activated
- Low-energy neutron transport on, no need for option LOW–NEUT (high energy neutron threshold at 19.6 MeV)
- Fully analogue absorption for low-energy neutrons
- Particle transport threshold set at 100 keV, except for neutrons (19.6 MeV) and (anti)neutrinos (0, but they are discarded by default anyway)
- Multiple scattering threshold at minimum allowed energy both for primary and secondary charged particles
- Delta ray production on with threshold 100 keV (see option DELTARAY)
- Restricted ionisation fluctuations on, for both hadrons/muons and EM particles (see option IONFLUCT)
- Tabulation ratio for hadron/muon dp/dx set at 1.03, fraction of the kinetic energy to be lost in a step set at 0.02 (see options DELTARAY, FLUKAFIX)

NEUTRONS

- Low-energy neutron transport on, no need for LOW–NEUT (high energy neutron threshold at 19.6 MeV)
- Non analogue absorption for low-energy neutrons with probability 0.95 for the last (thermal) group
- Both explicit and continuous heavy particle bremsstrahlung and pair production inhibited

NEW–DEFAults (*standard defaults active even if the DEFAULT card is not present)*

- EMF on, with electron and photon transport thresholds to be set using EMFCUT command
- Inelastic form factor corrections to Compton scattering activated (no need for EMFRAY)
- Low-energy neutron transport on (no need for LOW–NEUT). The neutron high energy threshold is set at 19.6 MeV.
- Non analogue absorption for low-energy neutrons with probability 0.95 for the last (thermal) group
- Particle transport threshold set at 10 MeV, except for neutrons (19.6 MeV) (50 MeV), and (anti)neutrinos (0, but they are discarded by default anyway)
- Multiple scattering threshold for secondary charged particles lowered to 20 MeV (equal to that of the primary ones)
- Delta ray production on with threshold 1 MeV (see option DELTARAY)
- Restricted ionisation fluctuations on, for both hadrons/muons and EM particles (see option IONFLUCT)
- Heavy particle e^+e^- pair production activated with full explicit production (with the minimum threshold $= 2\ m_e$)
- Heavy particle bremsstrahlung activated with explicit photon production above 1 MeV
- Muon photonuclear interactions activated with explicit generation of secondaries

PRECISIOn

- EMF on
- Rayleigh scattering and inelastic form factor corrections to Compton scattering activated
- Detailed photoelectric edge treatment and fluorescence photons activated
- Low-energy neutron transport on (high energy neutron threshold at 19.6 MeV)
- Fully analogue absorption for low-energy neutrons
- Particle transport threshold set at 100 keV, except neutrons (19.6 MeV) and (anti)neutrinos (0, but they are discarded by default anyway)
- Multiple scattering threshold at minimum allowed energy both for primary and secondary charged particles
- Delta ray production on with threshold 100 keV (see option DELTARAY)
- Restricted ionisation fluctuations on, for both hadrons/muons and EM particles (see option IONFLUCT)
- Tabulation ratio for hadron/muon dE/dx set at 1.04, fraction of the kinetic energy to be lost in a step set at 0.05, number of dp/dx tabulation points set at 80 (see options DELTARAY, FLUKAFIX)
- Heavy particle e^+e^- pair production activated with full explicit production (with the minimum threshold $= 2\ m_e$)
- Heavy particle bremsstrahlung activated with explicit photon production above 300 keV
- Muon photonuclear interactions activated with explicit generation of secondaries
- Heavy fragment transport activated

SHIELDINg

- Low-energy neutron transport on (the neutron high energy threshold is set at 19.6 MeV)
- Non analogue absorption for low-energy neutrons with probability 0.95 for the last (thermal) group
- Particle transport threshold set at 10 MeV, except neutrons (19.6 MeV) and (anti)neutrinos (0, but they are discarded by default anyway)
- Multiple scattering threshold for secondary charged particles lowered to 20 MeV (= primary ones)
- Both explicit and continuous heavy particle bremsstrahlung and pair production inhibited
- EMF off!!! This default is meant for simple hadron shielding only!

Notes

1. If an option does not appear in input, FLUKA provides default parameter values in most cases. Standard defaults are also applied when the option is present but not all its WHAT and SDUM parameters have been defined explicitly by the user. However, some types of problems are better handled using different defaults. Option DEFAULTS allows to override the standard ones with others, tuned to a specific class of transport problems.
   The present set of defaults (valid if no DEFAULTS card is issued) is equivalent to that set by SDUM = NEW–DEFAults.
2. **Important!** Option DEFAULTS must be issued at the very beginning of input. It can be preceded only by a GLOBAL card and by commands such as COMMENT or TITLE. This is one of the rare cases, like GLOBAL, MAT–PROP and PLOTGEOM, where sequential order of input cards is of importance in FLUKA (see Chap. 6).
3. The name of the SHIELDINg default refers to simple calculations for proton accelerators, where the electromagnetic component can be neglected. It is not applicable to electron accelerator shielding or any other shielding problem where the gamma component is important.

4. The responsibility of choosing reasonable defaults, compatible with the rest of input, is left to the user. In particular, choosing the defaults corresponding to pure EM cascade or to pure low-energy neutron problems has the effect of turning off all initialisations related to the hadronic generators. This will save a considerable time at the beginning of the run, but will lead to a crash if a hadron generator is called because of some other input option. In particular, SDUM = EM–CASCA is incompatible with option PHOTONUC and with beam/source particles different from PHOTON, ELECTRON and POSITRON; and SDUM = NEUTRONS is incompatible with option EMF, with any beam particle different from NEUTRON and with energies higher than 19.6 MeV. On the other hand, it is possible to override some of the defaults, in particular the various thresholds, by issuing the corresponding command after DEFAULTS (PART–THR, EMFCUT, DELTARAY, etc.)

Example:

```
*...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....
DEFAULTS         0.0       0.0       0.0       0.0       0.0      0.0 EM-CASCA
*  The above declaration refers to a problem where only electrons, positrons
*  and photons are transported.
```

Table 7.1: Defaults defined by different SDUM values of the DEFAULTS command

Command		DEFAULTS card missing	SDUM values								
			CALORIME	EET/TRAN	EM-CASCA	HADROTHE	ICARUS	NEUTRONS	NEW-DEFA	PRECISIO	SHIELDIN
MCSTHRES	WHAT(1)	-0.02	1.	-0.01	-0.02	1.	1.	-0.02	**-0.02**	1.	-0.02
	WHAT(2)	-0.02	1.	-0.02	-1.	1.	1.	-1.	**-0.02**	1.	-0.02
DELTARAY	WHAT(1)	0.001	0.0001	-1.	-1.	0.0001	0.0001	-1.	**0.001**	0.0001	-1.
	WHAT(2)	50.	80.	50.	50.	50.	80.	50.	**50.**	80.	50.
	WHAT(3)	1.15	1.15	1.15	1.15	1.03	1.04	1.15	**1.15**	1.04	1.15
EMF		ON	ON	OFF	ON	ON	ON	OFF	**ON**	ON	OFF
EMFFLUO	WHAT(1)	0.	1.	0.	1.	0.	1.	0.	**0.**	1.	0.
EMFRAY	WHAT(1)	3.	1.	0.	1.	3.	1.	0.	**3.**	1.	0.
EVENTYPE	WHAT(3)	1.	1.	1.	-2.	1.	1.	-2.	**1.**	1.	1.
FLUKAFIX	WHAT(1)	0.1	0.08	0.1	0.1	0.05	0.02	0.1	**0.1**	0.05	0.1
IONFLUCT	WHAT(1)	1.	1.	1.	-1.	1.	1.	-1.	**1.**	1.	-1.
	WHAT(2)	1.	1.	-1.	1.	1.	1.	-1.	**1.**	1.	-1.
LOW-BIAS	WHAT(2)	72.	73.	72.	72.	72.	73.	73.	**72.**	73.	72.
	WHAT(3)	0.95	0.85	0.95	0.85	0.95	0.85	0.85	**0.95**	0.85	0.95
LOW-NEUT		ON	ON	ON	OFF	ON	ON	ON	**ON**	ON	ON
	WHAT(6)	0.	1.	0.	0.	0.	1.	1.	**0.**	1.	0.
PAIRBREM	WHAT(1)	3.	3.	-3.	-3.	3.	3.	-3.	**3.**	3.	-3.
	WHAT(2)	0.	0.	N.A.	N.A.	-1.	0.	N.A.	**0.**	0.	N.A.
	WHAT(3)	0.001	0.0003	N.A.	N.A.	-1.	0.0003	N.A.	**0.001**	0.0003	N.A.
PART-THR	WHAT(1)	-0.010	$-0.001\frac{m}{m_p}$	-0.001	-0.050	-0.0001	-0.0001	-0.050	**-0.010**	-0.0001	-0.010

7.14 DELTARAY

Activates delta ray production by muons and charged hadrons and controls the accuracy of the dp/dx tabulations

See also IONFLUCT

WHAT(1) $>$ 0.0: kinetic energy threshold (GeV) for delta ray production (discrete energy transfer). Energy transfers lower than this energy are assumed to take place as continuous energy losses

$=$ 0.0: ignored

$<$ 0.0: resets the default to infinite threshold, i.e., no delta ray production

Default $=$ 0.001 if option DEFAULTS (p. 83) is not used, or if it is used with SDUM $=$ NEW–DEFAults.
If DEFAULTS is used with SDUM $=$ CALORIMEtry, HADROTHErapy, ICARUS or PRECISIOn, the default is 0.0001.
If it is used with any other SDUM value, the default is -1.0 (continuous slowing down approximation without production of delta rays)

WHAT(2) $>$ 0.0: number of logarithmic intervals for dp/dx momentum loss tabulation

$=$ 0.0: ignored

$<$ 0.0: resets the default to 50.0

Default $=$ 50.0 (this is the default if option DEFAULTS is not used, or if it is used with anything but SDUM $=$ CALORIMEtry, ICARUS or PRECISIOn).
With the latter, the default is 80.
See Note 1 for more details

WHAT(3) $>$ 1.0: logarithmic width of dp/dx momentum loss tabulation intervals (ratio between upper and lower interval limits)

0.0 $\leq$ WHAT(3) $\leq$ 1.0: ignored

$\leq$ 0.0: resets the default to 1.15

Default $=$ 1.15 (this is the default if option DEFAULTS is not used, or if it is used with any SDUM value but HADROTHErapy, ICARUS or PRECISIOn).
If DEFAULTS is used with SDUM $=$ ICARUS or PRECISIOn, the default is 1.04
With SUM $=$ HADROTHErapy the default is 1.03
See Note 1 for more details

WHAT(4) $=$ lower index bound of materials where delta ray production or specified tabulation accuracy are requested
("*From material WHAT(4)...*")

Default $=$ 3.0

WHAT(5) $=$ upper index bound of materials where delta ray production or specified tabulation accuracy are requested
("*... to material WHAT(5)...*")

Default $=$ WHAT(4)

WHAT(6) $=$ step length in assigning indices
("*... in steps of WHAT(6)*")

Default $=$ 1.0

SDUM $=$ PRINT : prints the dp/dx tabulations for the given materials on standard output
$=$ NOPRINT : resets to no printing a possible previous request for these materials
blank : ignored

Default $=$ NOPRINT

Default (option DELTARAY not given): the defaults depend on option DEFAULTS as explained above. See also Note 8 and Table 7.1 at p. 88.

Notes

1. The upper and lower limit of the dp/dx tabulations are determined by the options BEAM (p. 64) and PART–THR (p. 172), or by the corresponding defaults. Therefore, either the number *or* the width of the intervals are sufficient to define the tabulations completely. If both WHAT(2) and WHAT(3) are specified, or if the value of both is defined implicitly by the chosen default, the most accurate of the two resulting tabulations is chosen.
2. The lower tabulation limit is the momentum of the charged particle which has the lowest transport threshold. The upper limit corresponds to the maximum primary energy (as set by BEAM plus an additional amount which is supposed to account for possible exoenergetic reactions, Fermi momentum and so on.
3. This option concerns only charged hadrons and muons. Delta rays produced by electrons and positrons are always generated, provided their energy is larger than the production threshold defined by option EMFCUT.
4. Request of delta ray production is not alternative to that of ionisation fluctuation (see IONFLUCT, p. 128). The two options, if not used at the same time, give similar results as far as transport and energy loss are concerned, but their effect is very different concerning energy deposition: with the IONFLUCT option the energy lost is sampled from a distribution but is deposited along the particle track, while DELTARAY, although leading to similar fluctuations in energy loss, will deposit the energy along the delta electron tracks, sometimes rather far from the primary trajectory. IONFLUCT can be used even without requesting the EMF option, while when requesting DELTARAY the EMF card must also be present (or implicitly activated by the selected default — see option DEFAULTS, p. 83) if transport of the generated electrons is desired.
5. Normally, the energy threshold for delta ray production should be higher than the electron energy transport cut-off specified by option EMFCUT (p. 102) for discrete electron interactions. If it is not, the energy of the delta electron produced is deposited on the spot. As explained above, this will result in correct energy loss fluctuations but with the energy deposited along the particle track, a situation similar to that obtained with IONFLUCT alone.
6. Note that FLUKA makes sure that the threshold for delta ray production is not set much smaller than the average ionisation potential.
7. Presently, DELTARAY can be used together with the IONFLUCT option with a threshold for delta rays chosen by the user. As a result, energy losses larger than the threshold result in the production and transport of delta electrons, while those smaller than the threshold will be sampled according to the correct fluctuation distribution.
8. Here are the settings for delta ray production and dp/dx tabulations corresponding to available DEFAULTS options:

   ICARUS, PRECISIOn: threshold for delta ray production 100 keV; momentum loss tabulation with 80 logarithmic intervals or 1.04 logarithmic width (whichever is more accurate)

   CALORIMEtry: threshold for delta ray production 100 keV; momentum loss tabulation with 80 logarithmic intervals or 1.15 logarithmic width

   HADROTHErapy: threshold for delta ray production 100 keV; momentum loss tabulation with 50 logarithmic intervals or 1.03 logarithmic width

   NEW–DEFAults, or DEFAULTS missing: threshold for delta ray production 1 MeV; momentum loss tabulation with 50 logarithmic intervals or 1.15 logarithmic width

   Any other SDUM value: no delta ray production; momentum loss tabulation with 50 logarithmic intervals or 1.15 logarithmic width

Example:

```
*...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+...
DELTARAY        0.01       30.        0.0       3.0      18.0          PRINT
DELTARAY        0.02        0.0      1.05       4.0      12.0      8.0 NOPRINT
*    In this example, delta-rays with energies higher than 20 MeV (0.02 GeV)
*    will be produced in materials 4 and 12; for the same materials,
*    logarithmic intervals with a ratio of 1.05 between the the upper and the
*    lower limit of each interval are requested for the dp/dx tabulation. For
*    all other materials with number between 3 and 18, delta-rays are
*    produced above 10 MeV and 30 intervals are used in the dp/dx tabulation.
*    Tabulations are printed for all materials except 4 and 12.
```

7.15 DETECT

Scores energy deposition on an event by event basis (detector), providing coincidence and anti-coincidence capabilities such as those of a trigger.

In the following, an "event" means energy deposited in one or more *detector regions* by one primary particle and its descendants, i.e., between two successive calls to the FEEDER subroutine (case of an incident beam) or to a user-written SOURCE subroutine (case of an extended source, or of a source read from a collision file or sampled from a distribution).
A "signal" means energy deposited in one or more *trigger regions* by the same primary particle and descendants (i.e., between the same calls).
The output of DETECT is a distribution of energy deposited per event in the region(s) making up the detector in (anti)coincidence with a signal larger than a given cut-off or threshold in the trigger region(s).
It is also possible to define *detector combinations* (**NOT YET IMPLEMENTED!!!)**

This option can extend over several sequential cards. The meaning of the parameters on the first card are:

WHAT(1) $\leq$ 0.0 for a detector, $>$ 0.0 for a combination of detectors.

If **WHAT(1)** $\leq$ 0.0:

WHAT(2) = minimum total energy to be scored in the detector regions in one event, i.e., lower limit of the event distribution
Default = 0.0

WHAT(3) = maximum total energy to be scored in the detector regions in one event, i.e., upper limit of the event distribution
No default

WHAT(4) = cut-off energy for the signal (coincidence/anticoincidence threshold). The energy deposition event is scored only if a total of more than WHAT(4) GeV are/aren't correspondingly deposited in the trigger regions
Default = 10^{-9} (= 1 eV)

WHAT(5) $\geq$ 0.0: the detector regions, taken together, must be considered in *coincidence* with the trigger regions (also taken together)
$<$ 0.0: the detector regions must be considered in *anti-coincidence* with the trigger regions
Default: anti-coincidence

WHAT(6) $\geq$ 0.0: first region of the *detector*
$<$ 0.0: |WHAT(6)| is the first region of the *trigger* (the other regions will be given with continuation cards, see below)
No default

SDUM = detector name (max. 10 characters) unless the character "&" is present

Continuation card (if present):

WHAT(1) = same as WHAT(1) for the first card

WHAT(2) – **WHAT(6)** = following regions (with sign). If $>$ 0.0, they are considered detector regions, otherwise trigger regions

SDUM = "&" in any position in column 71 to 78 (or in the last field if free format is used)

Note: if no trigger region is given (i.e., no region with negative sign), a simple event-by-event scoring takes place

If **WHAT(1)** > 0.0: **NOT YET IMPLEMENTED!!!**

|WHAT(2)|: first detector to be considered for this combination, in coincidence if WHAT(2) > 0.0, in anticoincidence otherwise
Default : ignored

|WHAT(3)|: second detector to be considered for this combination, in coincidence if WHAT(3) > 0.0, in anticoincidence otherwise
Default : ignored

|WHAT(4)|: third detector to be considered for this combination, in coincidence if WHAT(4) > 0.0, in anticoincidence otherwise
Default : ignored

|WHAT(5)|: fourth detector to be considered for this combination, in coincidence if WHAT(5) > 0.0, in anticoincidence otherwise
Default : ignored

|WHAT(6)|: fourth detector to be considered for this combination, in coincidence if WHAT(6) > 0.0, in anticoincidence otherwise
Default : ignored

SDUM = combination name (max. 10 characters) unless the character "&" is present

Continuation card (if present):

WHAT(1) = same as WHAT(1) for the first card

WHAT(2) – **WHAT(6)** = following detectors (with sign). If > 0.0, they are considered in coincidence, otherwise in anticoincidence
SDUM = "&" in any position (or in the last field if free format is used)

Default (option DETECT not given): no (anti)coincidence scoring

Notes

1. Output from DETECT is written unformatted on logical unit 17. To recover the output, it is necessary to run a Fortran program containing the following lines:

```
.....................................
CHARACTER*80 RUNTIT, RUNTIM*32, CHNAME*10
INTEGER*4 NCASE, NDET, NBIN, IV(1024)
REAL EMIN, EBIN, ECUT
.....................................
.....................................
READ(17) RUNTIT, RUNTIM, WEIPRI, NCASE
READ(17) NDET, CHNAME, NBIN, EMIN, EBIN, ECUT
READ(17) (IV(I), I = 1, NBIN)
.....................................
```

This is the meaning of variables read:

– RUNTIT: title of the job (as given by input option TITLE)

- `RUNTIM`: time of the job (printed also at the beginning of the main output)
- `WEIPRI`: total weight of the primary particles
- `NCASE`: number of primary particles
- `NDET`: detector number
- `CHNAME`: detector name (= SDUM of the first DETECT card)
- `NBIN`: number of energy bins (presently fixed = 1024)
- `EMIN`: minimum total energy (= WHAT(2) of the first DETECT card)
- `EBIN`: width of each energy bin
- `ECUT`: cut-off energy for the signal (= WHAT(4) of the first DETECT card)
- `NBIN` values `IV(I)`: they are the spectrum channels, or energy bins

2. **Important:** option DETECT will give meaningful results *only* when FLUKA is used in a completely analogue mode, since correlations are destroyed by biasing. Thus, DETECT cannot be used together with any biasing option or weight-changing facility. It is recommended for this purpose to issue a GLOBAL command with WHAT(2) $<$ `0.0` at the beginning of input (see GLOBAL, Sec 7.33).
   A list of incompatible options is: BIASING, EMF–BIAS, LOW–BIAS, LAM–BIAS, WW–FACTOr, EMFCUT with WHAT(3) $<$ `0.0`, EMF with WHAT(6) $\neq$ `1.0`, EXPTRANS, LOW–DOWN.

Example:

```
*...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....
DETECT           0.0     1.E-4     1.E-2     5.E-5       1.0      7.0 Coincid_1
DETECT           0.0      -9.0     -12.0      10.0      11.0           &
*   The meaning of the above lines is the following:
*   a "signal" equal to energy deposition in regions 7 + 10 + 11 will be
*   scored if:
*   1) that signal is larger than 1.E-4 GeV and smaller than 0.01 GeV
*   2) at the same time (in coincidence) an energy deposition of at least
*   5.0E-5 GeV occurs in regions 9 + 12
*   The output will give a signal (event) spectrum between the energy
*   limits 1.0E-4 and 0.010 GeV, distributed over a fixed number of channels
*   (1024 in the standard version).
```

7.16 DISCARD

Defines the particle types to be discarded (i.e., not to be transported)

WHAT(3) – **WHAT(6)** : id-number of particles to be discarded (see particle numbering in Table 5.1, p. 41).
Setting one of the WHATs to a negative value will cancel a previous corresponding DISCARD command (explicit or by default).
When full heavy particle transport is activated (see EVENTYPE, p. 116), discarding of heavies can be performed setting the WHATs = (1000 + K_{heavy}), with K_{heavy} = 3...6 (heavy ion particle code):

3 = ^{2}H, 4 = ^{3}H, 5 = ^{3}He, 6 = ^{4}He, 7-12 = fission fragments

Undiscarding heavies is obtained by setting WHATs = (1000 - K_{heavy}).
The whole scheme is shown in the following table:

	Discard	**Undiscard**
^{2}H	1003	997
^{3}H	1004	996
^{3}He	1005	995
^{4}He	1006	994
fission fragm.	1007-1012	993-988

No default

SDUM : not used

Default (option DISCARD not given): only neutrinos and antineutrinos are discarded by default. Set the WHATs = -5.0, -6.0, -27.0, -28.0, -43.0, -44.0 in order to have them transported.

Notes

1. There is no limit to the number of DISCARD definitions given.
2. Discarding a particle means that that type of particle will possibly be produced but not transported.
3. The user may want to process some particle types with other programs providing only the production by the FLUKA code. These particles can be discarded. The results will then not contain the contribution of the discarded particle types and of their descendants.
4. Neutrinos are always discarded by default, to avoid useless tracking. To force neutrinos (or other particles) to be *not* discarded, set their particle number to negative. In that case, however, remember that:
   - no neutrino cross-sections are available for transport in FLUKA: these particles are just tracked through the geometry until they escape. Boundary crossing and track-length density can however be scored if requested
   - if neutrinos are not discarded, their transport threshold is set = 0.0 by default. This value can be changed (by option PART–THR, p. 172), but it must be kept in mind that the energy of any neutrino produced below the threshold *will be deposited locally*, generating a likely bias in dose calculations.
5. **Warning:** discarding the particles which propagate the hadronic cascade (neutrons, protons, pions) will lead in general to unpredictable results.

Example:

```
*...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+...
DISCARD          3.0       4.0       7.0      10.0      11.0      23.0
*   This example illustrates a typical situation where the use of DISCARD
*   can considerably reduce the computing time: for instance in pure
*   hadronic or neutron problems (hadron fluence calculations without interest
*   in energy deposition). In this case electrons, positrons, photons, muons
*   and pi0 do not contribute to the result and can be discarded
```

7.17 DPMJET

Defines some I/O parameters relevant to the heavy ion event generator DPMJET.

See also BME, EVENTYPE, MYRQMD, PHYSICS, RQMD

Option DPMJET is useful only for development work: generally the user does not need this option — to request activation of DPMJET use the command EVENTYPE.

WHAT(1) : format of the required pre-processed Glauber parameters. A complete set of projectile-target combinations for the entire FLUKA energy range is utilised. To revert to the original DPMJET format set this value to 1. See also Note 2 below.
Default = `0.0`

WHAT(2) : allows to change the logical unit number which is assigned to the DPMJET output
Default = `19.`

WHAT(3) : selects the level of output verbosity for DPMJET. The default corresponds to minimal output. If this value is set to `1.0`, initialisation and minor error messages are printed. In addition, one can request information about the DPMJET common block (the internal list of objects considered by DPMJET). If set equal to `2.0` all the available information is output, if set equal to `3.0` only information about final residual nuclei is output, and if set to `4.0` information about all final state objects is written.
Default = `0.0`

WHAT(4) – **WHAT(6)**: not used

SDUM : not used

Default (option DPMJET not given): all the above defaults apply

Notes

1. FLUKA utilises DPMJET and a heavily modified version of a RQMD–2.4 implementation (H. Sorge [160–162] in order to simulate heavy ion interactions.
2. Glauber parameter sets in the original ASCII format can be generated by running the stand-alone DPMJET program. Only up to 10 different projectile/target combinations can be initialised per run. Parameter sets in excess of the first 10 combinations will be provided by the default mechanism.

Example:

```
*...+....1....+....2....+....3....+....4....+....5....+....6....+....7...
DPMJET          0.0       71.        4.
* running with the pre-processed, full matrix of Glauber parameter sets
* but requesting more verbose output for DPMJET: initialisation related
* messages, all error messages and information about final state objects
* in the DPMJET common block.
* DPMJET is told not to read any specific input options.
* All output will be found in file connected to logical unit 71 instead
* of 19 (under Linux this file would be named fort.71)
```

7.18 ELCFIELD

Defines an homogenous electric field

See also MGNFIELD

NOT YET IMPLEMENTED!!!

WHAT(1) = largest angle (in degrees) that a particle is allowed to travel in a single step
Default : 20°

WHAT(2) = error of the boundary iteration (minimum accuracy accepted in determining a boundary intersection)
Default : 0.01 cm

WHAT(3) = minimum step if the step is forced to be smaller due to a too large angle
Default : 0.1 cm

WHAT(4) = E_x (**x**-component of the electric field, in kV/cm)

WHAT(5) = E_y (**y**-component of the electric field, in kV/cm)

WHAT(6) = E_z (**z**-component of the electric field, in kV/cm)

Note

1. This option is being implemented, but it is not yet operational.

7.19 EMF

Activates **E***lectro***M***agnetic* **F**LUKA*: transport of electrons, positrons and photons*

See also DEFAULTS, DELTARAY, EMF–BIAS, EMFCUT, EMFFIX, EMFFLUO, EMFRAY, MULSOPT, PHOTONUC

WHAT(1) – **WHAT(6):** not used

SDUM = EMF–OFF to switch off electron and photon transport. Useful with the new defaults where EMF is on by default

Default : EMF on

Default (option EMF not given): if option DEFAULTS is not used, or if it is used with SDUM = NEW–DEFAults, CALORIMEtry, EM–CASCAde, HADROTHErapy, ICARUS or PRECISIOn, electrons, positrons and photons are transported.
If DEFAULTS is used with SDUM = OLD–DEFAults, EET/TRANsmut, NEUTRONS, SHIELDINg or anything else, electrons, positrons and photons are not transported (see Note 2). To avoid their energy to be deposited at the point of production, it is generally recommended to discard those particles (see Note 5).
See also Table 7.1 at p. 88.

Notes

1. Option EMF is used to request a detailed transport of electrons, positrons and photons. Even if the primary particles are not photons or electrons, photons are created in high-energy hadron cascades, mainly as a product of π° decay, but also during evaporation and fission of excited nuclei; and capture gamma-rays are generated during low-energy neutron transport. Electrons can arise from muon decay or can be set in motion in knock-on collisions by charged particles (delta-rays)
2. If EMF has been turned off by overriding the default (by setting SDUM = EMF–OFF or by a DEFAULT option which switches off electron-photon transport, such as OLD–DEFAults, EET/TRANsmut, NEUTRONS, SHIELDING, not accompanied by an explicit EMF request), such electrons, positrons and photons are not transported and their energy is deposited on the spot at the point of creation unless those particles are DISCARDed (see Note 5 below).
3. Of course, it is also mandatory to request option EMF (either explicitly or implicitly via option DEFAULTS, p. 83) in any pure electron, positron or photon problem (i.e., with electrons, positrons or photons as primary particles).
4. Using EMF without any biasing can lead to very large computing times, especially in problems of high primary energy or with low-energy cut-offs. See in particular leading-particle biasing with EMF–BIAS (p. 98).
5. In case of a pure hadron or neutron problem (e.g., neutron activation calculation) it is recommended to DISCARD electrons, positrons and photons (id-number 3, 4 and 7, see p. 94). In this case it is irrelevant whether the EMF card is present or not. Discarding only electrons and positrons, but not photons, may also be useful in some cases (for instance when calculating photon streaming in a duct).
6. An alternative is to set very large energy cut-offs for electrons and positrons (see EMFCUT, p. 102). That will result in the electron energy being deposited at the point of photon interaction (*kerma approximation*, often sufficient for transport of photons having an energy lower than a few MeV).
7. Hadron photoproduction is dealt with by option PHOTONUC (p. 174).

Example:

```
*...+....1....+....2....+....3....+....4....+....5....+....6....+....7...+...8
EMF                                                                 EMF-OFF
*   This command must be issued without any WHAT parameter.
```

7.20 EMF–BIAS

Sets electron and photon special biasing parameters, including leading particle biasing region by region and mean free path biasing material by material

See also EMF, EMFCUT

For **SDUM** = LPBEMF (default):

WHAT(1) > 0.0: Leading Particle Biasing (LPB) is activated. Which combination of leading particle biasing is actually set up depends on the bit pattern of WHAT(1).

Let WHAT(1) be represented as:

$$2^0 b_0 + 2^1 b_1 + 2^2 b_2 + 2^3 b_3 + 2^4 b_4 + 2^5 b_5 + 2^6 b_6 + 2^7 b_7 + 2^8 b_8 + 2^9 b_9$$

where the meaning of the ten bits is the following:

$b_0 = 1$: LPB activated for bremsstrahlung and pair production (old default)
$b_1 = 1$: LPB activated for bremsstrahlung
$b_2 = 1$: LPB activated for pair production
$b_3 = 1$: LPB activated for positron annihilation at rest
$b_4 = 1$: LPB activated for Compton scattering
$b_5 = 1$: LPB activated for Bhabha & Møller scattering
$b_6 = 1$: LPB activated for photoelectric effect
$b_7 = 1$: LPB activated for positron annihilation in flight
$b_8 = 1$: not used
$b_9 = 1$: not used

Note that WHAT(1) = 1022 activates LPB for all physical effects (values larger than 1022 are converted to 1022)

< 0.0: leading particle biasing is switched off
= 0.0: ignored

WHAT(2) > 0.0: energy threshold below which leading particle biasing is played for electrons and positrons (for electrons, such threshold refers to kinetic energy; for positrons, to total energy plus rest mass energy)
< 0.0: resets any previously defined threshold to infinity (i.e., leading particle biasing is played at all energies)
= 0.0: ignored
This value can be overridden in the user routine UBSSET (p. 320) by assigning a value to variable ELPEMF
Default : leading particle biasing is played at all energies for electrons and positrons

WHAT(3) > 0.0: energy threshold below which leading particle biasing is played for photons
< 0.0: resets any previously defined threshold to infinity (i.e., leading particle biasing is played at all energies)
= 0.0: ignored
This value can be overridden in the user routine UBSSET (p. 320) by assigning a value to variable PLPEMF.
Default : leading particle biasing is played at all energies for photons

WHAT(4) = lower bound of the region indices where the selected leading particle biasing has to be played ("*From region WHAT(4)...*")
Default = 2.0

WHAT(5) = upper bound of the region indices where the selected leading particle biasing has to be played ("*...to region WHAT(5)...*")
Default = WHAT(4)

WHAT(6) = step length in assigning indices ("*...in steps of WHAT(6)*")
Default = 1.0

SDUM = LPBEMF (Leading Particle Biasing for EMF). This is the default, for other values of SDUM see below.
This value can be overridden in the user routine UBSSET (p. 320) by assigning a value to variable LPEMF.

For **SDUM** = LAMBEMF, LAMBCOMP, LAMBBREM, LBRREMF, LBRRCOMP, LBRRBREM:

WHAT(1) $>$ 0.0 and $<$ 1.0: the interaction mean free paths for all electron and positron electromagnetic interactions (SDUM = LAMBEMF), or for electron/positron bremsstrahlung only (SDUM = LAMBBREM) are reduced by a multiplying factor = WHAT(1)
= 0.0: ignored
$<$ 0.0 or $\geq$ 1.0: resets to default (no mean free path biasing for electrons and positrons)

WHAT(2) $>$ 0.0 and $<$ 1.0: the interaction mean free paths for all photon electromagnetic interactions (SDUM = LAMBEMF), or for Compton scattering only (SDUM = LAMBCOMP) are reduced by a multiplying factor = WHAT(2)
= 0.0: ignored
$<$ 0.0 or $\geq$ 1.0: resets to default (no mean free path biasing for photons)

WHAT(3) = generation up to which the biasing has to be applied
Default : biasing is applied only the first generation (i.e., the primary BEAM or SOURCE particles)

WHAT(4) = lower bound of the indices of materials in which the indicated mean free path biasing has to be applied ("*From material WHAT(4)...*")
Default = 3.0

WHAT(5) = upper bound of the indices of materials in which the indicated mean free path biasing has to be applied ("*...to material WHAT(5)...*")
Default = WHAT(4)

WHAT(6) = step length in assigning indices ("*...in steps of WHAT(6)*")
Default = 1.0

SDUM = LAMBEMF (LAMbda Biasing for ElectroMagnetic FLUKA): mean free path biasing is applied to all electron, positron and photon interactions, and both the incident and the secondary particle are assigned a reduced weight
= LAMBCOMP (LAMbda Biasing for Compton interactions): mean free path biasing is applied only to photon Compton effect, and both the incident photon and the secondary electron are assigned a reduced weight
= LAMBBREM (LAMbda Biasing for bremsstrahlung interactions): mean free path biasing is applied only to electron and positron bremsstrahlung, and both the incident electron/positron and the secondary photon are assigned a reduced weight

= LBRREMF (LAMbda Biasing with Russian Roulette for ElectroMagnetic FLUKA): mean free path biasing is applied to all electron, positron and photon interactions, and the incident particle either is suppressed or survives with the same weight it had before collision, depending on a random choice

= LBRRCOMP (LAMbda Biasing with Russian Roulette for Compton interactions): mean free path biasing is applied only to photon Compton effect, and the incident photon either is suppressed or survives with the same weight it had before collision, depending on a random choice

= LBRRBREM (LAMbda Biasing with Russian Roulette for bremsstrahlung interactions): mean free path biasing is applied only to electron and positron bremsstrahlung, and the incident electron/positron either is suppressed or survives with the same weight it had before collision, depending on a random choice

Default : LPBEMF (see above)

Default : (option EMF–BIAS not given): none of the above biasings apply. Note, however, that leading particle biasing can also be requested by option EMFCUT, p. 102 (not recommended).

Notes

1. Depending on the SDUM value, different kinds of biasing are applied to the secondary particles issued from the reaction.
2. If SDUM = LPBEMF, the interaction point of electrons, positrons and photons is sampled analogically and Leading Particle Biasing is applied to the secondary particles, in a manner similar to that provided by option EMFCUT (p. 102).
   However, Leading Particle Biasing with EMFCUT applies to all electromagnetic effects, while EMF–BIAS can be tuned in detail for each type of electron and photon interactions.
3. With all other values of SDUM, the interaction point is sampled from an imposed (biased) exponential distribution, in a manner similar to that provided by option LAM–BIAS for hadrons and muons (p. 131). Further differences in SDUM values allow to restrict biasing to one specific type of interaction and/or to select different treatments of the incident particle.
4. If SDUM = LAMBEMF, LAMBCOM, LAMBBREM, the weights of both the incident and the secondary particle are adjusted according to the ratio between the biased and the physical interaction probability at the sampled distance.
5. If SDUM = LBRREMF, LBRRCOM, LBRRBREM, the suppression or survival of the incident particle (with unchanged weight) is decided by Russian Roulette with a probability equal to the ratio between the biased and the physical interaction probability at the sampled distance. The weight of the secondary particle is adjusted by the same ratio.
6. When using option EMF–BIAS, and in particular when choosing the Russian Roulette alternative, it is suggested to set also a weight window (cards WW–FACTOR, p. 244 and WW–THRESh, p. 249) in order to avoid too large weight fluctuations.
7. LAMBCOMP (LBRRCOMP) and LAMBBREM (LBRRBREM) are synonyms: i.e., input concerning photon interaction biasing given with SDUM = LAMBBREM (LBRRBREM) is accepted and treated in the same way as with SDUM = LAMBCOMP (LBRRCOMP); and input concerning electron/positron interaction biasing with SDUM = LAMBCOMP (LBRRCOMP) is the same as with LAMBBREM (LBRRBREM). This allows to issue just a single EMF–BIAS card requesting both electron and photon interaction biasing at the same time.
8. Option EMF–BIAS concerns only electromagnetic interactions; photonuclear interaction biasing is provided by option LAM–BIAS (p. 131).
9. **Leading Particle Biasing (LPB):**
   Leading particle biasing (available only for electrons, positrons and photons) is generally used to avoid the geometrical increase with energy of the number of particles in an electromagnetic shower. It is characteristic of all electromagnetic interactions that two particles are present in the final state: when this option is selected, only one of them (with a probability proportional to its energy) is randomly retained and its weight is adjusted accordingly. Derived from the EGS4 implementation [116], it has been modified to account for the indirectly enhanced penetration potential of positrons due to the emission of annihilation photons. The probability of each of the two particles to be selected is therefore not proportional to their kinetic energy but rather to their "useful" energy (kinetic plus — in the case of positrons only — twice the mass energy).

The weight of the particle selected is adjusted multiplying it by the inverse of the selection probability. This kind of biasing is aimed at reducing the mean computing time per history rather than the variance of the scored quantities (computer cost is defined as the product of variance times the computing time per primary particle). It is mainly used to estimate shower punchthrough (but comparable and even better efficiency can be obtained with importance splitting, see BIASING, p. 71), or to reduce the time spent in simulating secondary electromagnetic showers produced by π° in hadronic cascades. As any other kind of biasing, leading particle must be used with judgement, since it may lead to a slower convergence of dose estimation in some regions of phase space (see Note 5 to option BIASING).
In particular, the fact that the particle of highest energy is selected preferentially can have the following effects:

(a) the radial profile of the electromagnetic shower might be reproduced less efficiently. This is in general not very important for showers generated inside hadronic cascades, since the overall lateral spread is governed essentially by hadrons.

(b) a few low-energy particles might result with a very large weight giving rise to strong energy deposition fluctuations (this inconvenience can be limited to some extent by the use of a weight window). Therefore, biasing should be avoided in scoring regions and in adjacent ones, especially when using energy deposition bins of very small volume.

When applied in energy deposition calculations, the use of weight windows is recommended in order to avoid large local dose fluctuations (see WW–FACTOR, p. 244 and WW–THRESh, p. 249).

10. Option EMFCUT (p. 102) provides an alternative way to request LPB, but without the possibility to set an energy threshold or to limit biasing to a specified number of generations.

Example 1:

```
*...+....1....+....2....+....3....+....4....+....5....+....6....+....7...+...8
EMF-BIAS        152.        0.     5.E-4       16.       20.        2.LPBEMF
*   LPB is applied in regions 16, 18 and 20 as regards Compton scattering
*   below 0.5 MeV and positron annihilation in flight and at rest.
*   Code 152 = 2^3 (annihilation at rest) + 2^4 (Compton) + 2^7
*   (annihilation in flight).
```

Example 2:

```
*...+....1....+....2....+....3....+....4....+....5....+....6....+....7...+...8
EMF-BIAS       1022.       0.0       0.0       3.0       8.0
*   LPB is applied in regions 3, 4, 5, 6, 7 and 8 for all electron and photon
*   interactions at all energies
```

Example 3:

```
*...+....1....+....2....+....3....+....4....+....5....+....6....+....7...+...8
EMF-BIAS       1022.       0.0       0.0       1.0      15.0
EMF-BIAS         -1.       0.0       0.0       7.0      11.0       2.0
WW-FACTOR        0.5       5.0       1.0       1.0      15.0
WW-FACTOR        0.5       5.0       0.2       7.0      11.0       2.0
WW-THRESH        1.0     0.001      20.0       3.0       4.0
WW-THRESH        1.0     1.E-4      20.0       7.0
*   The above example illustrates the combined use of leading particle biasing
*   and a region-dependent weight-window. Leading particle biasing is requested
*   in all regions from 1 to 15, except 7, 9 and 11. To avoid too large weight
*   fluctuations, a weight window is defined such that at the lowest energies
*   (=< 20 keV for photons and =< 200 keV for electrons in regions 7, 9, 11;
*   =< 100 keV for photons and <= 1 MeV for electrons in the other regions),
*   Russian Roulette will be played for particles with a weight =< 0.5 and
*   those with weight larger than 5.0 will be splitted. The size of this window
*   (a factor 10) is progressively increased up to 20 at the higher threshold
*   (200 MeV for both electrons and photons in regions 7, 9 and 11, 1 GeV in
*   the other regions).
```

7.21 EMFCUT

Sets the energy thresholds for electron and photon production in different materials, and electron and photon transport cut-offs in selected regions. This command can also request leading particle biasing, but EMF–BIAS must be preferred.
It also allows to set an arbitrary energy threshold for all electron and photon interactions managed by EMF on a material basis. This is of course non-physical and it is provided primarily for particular studies where the user wants to switch off selectively a physical process.
Only meaningful when the EMF option is chosen (explicitly or implicitly via option DEFAULTS).

See also EMF, EMF–BIAS, BIASING, PART–THR, MATERIAL

For SDUM = PROD–CUT:

|WHAT(1)| = Energy threshold for electron and positron production in GeV:
- $>$ `0.0`: Energy threshold for electron and positron production is expressed as total energy (kinetic plus rest mass)
- $<$ `0.0`: Energy threshold for electron and positron production is expressed as kinetic energy
- $=$ `0.0`: ignored
- **Default** : equal to the lowest electron transport cut-off in all regions made of this material (see Note 1)

WHAT(2)
- $>$ `0.0`: Energy threshold for γ production in GeV
- $=$ `0.0`: ignored
- **Default** : equal to the lowest electron transport cut-off in all regions made of this material (see Note 1)

WHAT(3) : `FUDGEM` parameter. This parameter takes into account the contribution of atomic electrons to multiple scattering. For production and transport cut-offs larger than 100 keV it must be set = `1.0`, while in the keV region it must be set = `0.0`

Default = `0.0`

WHAT(4) = lower bound of the FLUKA material number where $e^{\pm}$ and photon production and transport threshold respectively equal to WHAT(1) and WHAT(2) apply. The material numbers are those pre-defined or assigned using a MATERIAL card.
("*From material WHAT(4)...*")
Default = `3.0`

WHAT(5) = upper bound of the FLUKA material number where $e^{\pm}$ and photon production and transport threshold respectively equal to WHAT(1) and WHAT(2) apply. The material numbers are those pre-defined or assigned using a MATERIAL card.
("*...to material WHAT(5)...*")
Default = WHAT(4)

WHAT(6) = step length in assigning the material number.
("*...in steps of WHAT(6)*")
Default = `1.0`

For SDUM = *blank*:

|WHAT(1)| = electron and positron transport energy cut-off in GeV:
- $>$ 0.0: electron and positron cut-off is expressed as total energy (kinetic plus rest mass)
- $<$ 0.0: electron and positron cut-off is expressed as kinetic energy
- = 0.0: ignored.

This value can be overridden in the user routine UBSSET (p. 320) by assigning a value to variable ELECUT

Default : the $e^{\pm}$ transport cut-off is set equal to the production threshold for discrete electron interactions

WHAT(2) $>$ 0.0: photon transport energy cut-off (GeV)
= 0.0: ignored.

This value can be overridden in the user routine UBSSET (p. 320) by assigning a value to variable GAMCUT

Default : the photon transport cut-off is set equal to threshold for photon production by electron bremsstrahlung

WHAT(3) $>$ 0.0: leading particle biasing is activated for electrons, positrons and photons. Which combination of leading particle biasing is actually set up depends on the bit pattern of WHAT(3).

Let WHAT(3) be represented as:

$$2^0 b_0 + 2^1 b_1 + 2^2 b_2 + 2^3 b_3 + 2^4 b_4 + 2^5 b_5 + 2^6 b_6 + 2^7 b_7 + 2^8 b_8 + 2^9 b_9$$

where the meaning of the ten bits is the following:

$b_0 = 1$: LPB activated for bremsstrahlung and pair production (old default)
$b_1 = 1$: LPB activated for bremsstrahlung
$b_2 = 1$: LPB activated for pair production
$b_3 = 1$: LPB activated for positron annihilation at rest
$b_4 = 1$: LPB activated for Compton scattering
$b_5 = 1$: LPB activated for Bhabha & Møller scattering
$b_6 = 1$: LPB activated for photoelectric effect
$b_7 = 1$: LPB activated for positron annihilation in flight
$b_8 = 1$: not used
$b_9 = 1$: not used

Note that WHAT(1) = 1022 activates LPB for all physical effects (values larger than 1022 are converted to 1022)

$<$ 0.0: no leading particle biasing for electrons, positrons and photons
= 0.0: ignored (previous definitions hold, if any; otherwise default, i.e., no leading particle biasing).

This value can be overridden in user routine UBSSET (p. 320) by assigning a value to variable LPEMF

WHAT(4) = lower bound of the region indices with electron cut-off equal to |WHAT(1)| and/or photon cut-off equal to WHAT(2) and/or leading particle biasing.
("*From region WHAT(4)...*")

Default = 2.0

WHAT(5) = upper bound of the region indices with electron cut-off |WHAT(1)| and/or photon cut-off equal to WHAT(2) and/or leading particle biasing.
("*... to region WHAT(5)...*")

Default = WHAT(4)

WHAT(6) = step length in assigning indices.
("*... in steps of WHAT(6)*")
Default = `1.0`

For SDUM = ELPO–THR:

WHAT(1) > `0.0`: kinetic energy threshold (GeV) for $e^{\pm}$ bremsstrahlung
= `0.0`: ignored
< `0.0`: resets to default
Default = `1.0`

WHAT(2) > `0.0`: kinetic energy threshold (GeV) for Bhabha/Møller scattering
= `0.0`: ignored
< `0.0`: resets to default
Default = `1.0`

WHAT(3) > `0.0`: kinetic energy threshold (GeV) for $e^{\pm}$ photonuclear interactions
= `0.0`: ignored
< `0.0`: resets to default
Default = `0.0`

WHAT(4) – **WHAT(6)**: see below

For SDUM = ANNH–THR:

WHAT(1) > `0.0`: kinetic energy threshold (GeV) for e^{+} annihilation
= `0.0`: ignored
< `0.0`: resets to default
Default = `0.0`

WHAT(2) – **WHAT(3)**: not used

WHAT(4) – **WHAT(6)**: see below

For SDUM = PHOT–THR:

WHAT(1) > `0.0`: energy threshold (GeV) for Compton scattering
= `0.0`: ignored
< `0.0`: resets to default
Default = `0.0`

WHAT(2) > `0.0`: energy threshold (GeV) for photoelectric effect
= `0.0`: ignored
< `0.0`: resets to default
Default = `0.0`

WHAT(3) > `0.0`: energy threshold (GeV) for gamma pair production
= `0.0`: ignored
< `0.0`: resets to default
Default = `0.0`

WHAT(4) – **WHAT(6)**: see below

For SDUM = PHO2–THR:

WHAT(1) $>$ 0.0: energy threshold (GeV) for Rayleigh scattering
$=$ 0.0: ignored
$<$ 0.0: resets to default
Default $=$ 0.0

WHAT(2) $>$ 0.0: energy threshold (GeV) for photonuclear interactions
$=$ 0.0: ignored
$<$ 0.0: resets to default
Default $=$ 0.0

WHAT(3) : not used

WHAT(4) – **WHAT(6)**: see below

For SDUM = ELPO–THR, ANNH–THR, PHOT–THR, PHO2–THR:

WHAT(4) = lower bound of the material indices in which the respective thresholds apply.
("*From material WHAT(4)...*")
Default $=$ 3.0

WHAT(5) = upper bound of the material indices in which the respective thresholds apply.
("*...to material WHAT(5)...*")
Default $=$ WHAT(4)

WHAT(6) = step length in assigning indices.
("*...in steps of WHAT(6)*")
Default $=$ 1.0

Default (option EMFCUT not given): no leading particle biasing, unless requested by the alternative option EMF–BIAS (p. 98). Cut-offs are set equal to production thresholds for discrete electron interactions and for discrete photon production.

Notes

1. Default values are available for the electron and photon production thresholds in electromagnetic interactions, but they are generated by a complex logic based on possible other settings (transport cut-offs, delta-ray production thresholds, DEFAULTS card, other defaults). It is not always guaranteed that the resulting values be appropriate to the problem of interest. Therefore, in order to have a good control of the physics, the user is recommended to provide explicit threshold values, or at least to check them on the main output.
2. Transport cut-offs set by EMFCUT override the production thresholds (set with SDUM = PROD–CUT) for discrete interactions, but only if higher than them. If lower, the production thresholds apply, of course. The production thresholds are overridden only for transport purposes, that is, particles with energy higher than production thresholds and lower than transport cut-offs are not transported but are still generated (their energy is deposited at the point of production). It is suggested to avoid such a situation unless it is really necessary, as particle generation demands a considerable computer time and partially offsets the gain due to a higher transport cut-off.
3. If WHAT(3) is set $>$ 0.0, Leading Particle Biasing (LPB) is applied at photon and/or $e^{\pm}$ interactions, in a manner similar to that provided by option EMF–BIAS (see Note 9 to that option for a description of the technique and guidance to its use). However, EMF–BIAS offers more flexibility, since it allows to set also an energy threshold for LPB, and to select a number of generations to which it can be applied.

Example 1:

```
*...+....1....+....2....+....3....+....4....+....5....+....6....+....7...
EMFCUT       -1.0E-5    1.0E-5       0.0       4.0       8.0          PROD-CUT
*   A production threshold of 10 keV is set for electrons, positrons and
*   photons in all materials from 4 to 8.
```

Example 2:

```
EMFCUT        -0.002     2.E-4       1.0       1.0      15.0
EMFCUT       -1.0E-4     1.E-5      -1.0       7.0      11.0       2.0
*   A kinetic energy transport cut-off for electrons and positrons is set
*   at 100 keV in regions 7, 9 and 11 and at 2 MeV in all other regions from 1
*   to 15. Photon transport cut-off is set equal to 10 keV in regions 7, 9, 11
*   and to 200 keV in the other regions.
```

7.22 EMFFIX

Sets the size of electron steps corresponding to a fixed fraction of the total energy. The setting is done by material, giving as many EMFFIX definitions as needed. Only meaningful when the EMF option has been requested (explicitly or implicitly via option DEFAULTS).

See also EMF, FLUKAFIX, MULSOPT, STEPSIZE

WHAT(1) = index of the material concerned

WHAT(2) = maximum fraction of the total energy to be lost in a step

Default : 20 % (it is strongly recommended not to set higher than this value!)

WHAT(3) = same as WHAT(1)

WHAT(4) = same as WHAT(2)

WHAT(5) = same as WHAT(1)

WHAT(6) = same as WHAT(2)

SDUM = PRINT : electron and positron dE/dx and maximum allowed step tabulations for this material are printed
= NOPRINT : tabulations are not printed (cancels any previous PRINT request for the given materials)
blank : ignored

Default = NOPRINT

Default (option EMFFIX not given): the energy lost per step is 20 % for all materials

Notes

1. The default provided (step length such that 20 % of the energy is lost) is acceptable for most routine problems. In dosimetry problems and in thin-slab geometries it is recommended not to exceed 5–10 %.
For a detailed discussion of the step length problem, see [64].
2. Related options are STEPSIZE (p. 200), MCSTHRES (p. 149), FLUKAFIX (p. 119) and MULSOPT (p. 153). MCSTHRES and FLUKAFIX concern only heavy charged particles (hadrons and muons), while STEPSIZE applies to *all* charged particles (hadrons, muons, electrons and positrons). However, STEPSIZE defines the steplength in cm and by region, while EMFFIX relates the step length to the maximum allowed energy loss and is based on materials. STEPSIZE works also in vacuum and is adapted to problems with magnetic fields; if both options are used, the smallest of the two steps is always chosen. Note however that if a step required by STEPSIZE is too small for the Molière algorithm, multiple scattering *is* turned off (contrary to what happens with EMFFIX). MULSOPT is very CPU-time consuming; however, it gives the highest accuracy compatible with the Molière theory. It is used rarely, mostly in low-energy and in backscattering problems.

Example:

```
*...+....1....+....2....+....3....+....4....+....5....+....6....+....7...
MATERIAL         13.    26.98     2.6989        3.        0.       0. ALUMINUM
MATERIAL         82.   207.20      11.35        4.        0.       0. LEAD
MATERIAL         29.    63.546      8.96       12.        0.       0. COPPER
MATERIAL          6.    12.000      2.00       26.        0.       0. CARBON
MATERIAL          7.    14.000    0.0012       27.        0.       0. NITROGEN
MATERIAL          8.    16.000    0.0014       28.        0.       0. OXYGEN
MATERIAL          1.     1.000    0.0001       29.        0.       1. HYDROGEN
MATERIAL          0.       0.0    1.0000       30.        0.       0. TISSUE
COMPOUND     5.57E-3      26.0  1.118E-3       27.  2.868E-2      28. TISSUE
COMPOUND    6.082E-2      29.0        0.        0.        0.       0. TISSUE
EMFFIX            3.      0.15        4.      0.15       12.     0.15
EMFFIX           30.      0.05        0.        0.        0.       0. PRINT
*    In this example, a maximum energy loss per step of 15% is requested
*    for aluminium, copper and lead, while a more accurate 5% is requested
*    for tissue
```

7.23 EMFFLUO

Activates a detailed treatment of photoelectric interactions and of the following atomic deexcitation, with production of fluorescence X-rays (and a rough treatment of Auger electrons)

See also EMF

This option, meaningful only if the EMF option has been requested (explicitly or implicitly via option DEFAULTS), requires a special unformatted file, pre-connected as logical input unit 13 (see Chap. 3). This file is delivered with the FLUKA code.

WHAT(1) $\geq$ 0.0: set fluorescence on
$\leq$ 0.0: set fluorescence off
= 0.0: ignored
Default = -1.0 (no fluorescence) if option DEFAULTS is not used, or if it is used with anything but CALORIMEtry, EM–CASCA, ICARUS or PRECISIOn.
If DEFAULTS is used with with a SDUM value equal to one of the latter, the default is 1.0 (fluorescence on).

WHAT(2) = lower bound of the material indices in which fluorescence is activated ("*From material WHAT(2)...*")
Default = 3.0

WHAT(3) = upper bound of the material indices in which fluorescence is activated ("*... to material WHAT(3)...*")
Default = WHAT(2)

WHAT(4) = step length in assigning indices ("*... in steps of WHAT(4)*")
Default = 1.0

WHAT(5), WHAT(6), SDUM: not used

Default (option EMFFLUO not given): fluorescence is not simulated unless option DEFAULTS is chosen with SDUM = CALORIMEtry, EM–CASCA, ICARUS or PRECISIOn.
See also Table 7.1 at p. 88.

Notes

1. Selection of EMFFLUO option is only meaningful for a material defined with electron and photon cut-offs lower than the highest K-edge in the elements constituents of that material.
2. When EMFFLUO is activated for a compound material, if the incident photon energy is lower than the highest K-edge for any constituent element, FLUKA uses separate parameterised representations of the photoelectric cross-section between different edges of each constituent element (all levels having a binding energy $>$ 1 keV are tabulated).
   If the photon energy is higher than the highest K-edge, average cross-sections for the compound are used, but FLUKA still samples a single element on which to interact, in order to generate the correct fluorescence photon or Auger electron.

Example:

```
*...+....1....+....2....+....3....+....4....+....5....+....6....+....7...
EMF
EMFCUT         1.E-5     1.E-5       0.0       17.       18.
EMFFLUO          1.0       17.       18.
*   In the above example, the user activates fluorescence production in
*   Lead and Uranium (standard FLUKA material numbers 17 and 18). The
*   photon and electron cut-off has been set at 10 keV (the K-edge for
*   Pb and U is of the order of 100 keV).
```

7.24 EMFRAY

Activates Rayleigh (coherent) scattering and Compton binding corrections in selected regions. Only meaningful when the EMF option has been requested, explicitly or implicitly via option DEFAULTS.

See also EMF

WHAT(1) $\geq$ 1.0: both Rayleigh scattering and Compton binding corrections are activated
= 2.0: only Rayleigh scattering is activated
= 3.0: only Compton binding corrections are activated
= 0.0: ignored
< 0.0: no Rayleigh scattering and no binding corrections for Compton
Default : if option DEFAULTS is used with SDUM = NEW–DEFAults or HADROTHErapy, the default is 3.0.
If it is used with SDUM = CALORIMEtry, EM–CASCAde, ICARUS or PRECISIOn, the default is 1.0.
With any other value of SDUM, the default is 0.0.

WHAT(2) = lower bound of the region indices where Rayleigh scattering and/or Compton binding corrections are requested
("*From region WHAT(2)...*")
Default = 2.0

WHAT(3) = upper bound of the region indices where Rayleigh scattering and/or Compton binding corrections are requested
("*... to region WHAT(3)...*")
Default = WHAT(2)

WHAT(4) = step length in assigning indices
("*... in steps of WHAT(4)*")
Default = 1.0

WHAT(5), WHAT(6), SDUM: not used

Default (option EMFRAY not given): if option DEFAULTS is used with SDUM = NEW–DEFAults or HADROTHErapy, binding corrections in Compton scattering are activated, but not Rayleigh scattering.
If DEFAULTS is not used, or is used with SDUM = CALORIMEtry, EM–CASCAde, ICARUS or PRECISIOn, both are activated.
With any other value of SDUM, binding corrections and Rayleigh scattering are not activated.
See also Table 7.1 at p. 88.

Notes

1. The treatment of Rayleigh scattering is rather poor for non monoatomic materials (it assumes additivity and does not take into account important molecular effects). However, Rayleigh scattering, in general, has little effect on energy deposition and on particle transport.

Example:

```
*...+....1....+....2....+....3....+....4....+....5....+....6....+....7...
EMF
EMFRAY          1.0       8.0      12.0       4.0
EMFRAY          3.0       9.0      15.0       2.0
*   In the above example, Compton binding corrections are requested in
*   regions 8 to 15, excluding regions 10 and 14.
*   Rayleigh scattering is requested only in regions 8 and 12
```

7.25 EVENTBIN

For calorimetry only.
Superimposes a binning structure to the geometry and prints the result after each "event"

See also USRBIN, EVENTDAT

For a description of input for this option, refer to USRBIN (7.77). Meaning of WHATs and SDUM is practically identical for the two options. The only difference here is that if WHAT(1) is given with a negative sign, only non-zero data ("hit cells") are printed.
For Cartesian binning, WHAT(1) = `0.0` prints all cells, and a negative number > `-0.5` must be used to print only the "hit cells".
See Note 2 below and Note 3 to option ROTPRBIN (p. 192).

This card is similar to USRBIN (p. 218), but the binning data are printed at the end of each event (primary history).
Information about the binning structure is printed at the beginning, then binning data are printed at the end of each event *without any normalisation* (i.e., energy per bin and not energy density per unit incident particle weight).

If the sign of WHAT(1) in the first card defining the binning is negative, only those elements of the binning which are non zero are printed at the end of each event, together with their index.

Default (option EVENTBIN not given): no event-by-event binning.

To read EVENTBIN unformatted output, see instructions for USRBIN, with the following differences:

- first all binning definitions are written
- then, for each event all binnings are dumped, two records for each binning:
  First record: `NB, NCASE, WEIGHT` (resp. binning number, number and weight of the event)
  Second record: binning energy deposition data (see USRBIN)
- if the `LNTZER` flag (only non-zero cells) is activated, the energy deposition can be read as
  ```
  READ (...)  NHITS, (IHELP(J), GMHELP(J), J = 1, NHITS)
  ```
  where:
  `IHELP` = cell index = `IX + (IY-1)*NX + (IZ-1)*NX*NY`
  `GMHELP` = cell content

Notes

1. Normally, this option is meaningful only in fully analogue runs. Any biasing option should be avoided, and a GLOBAL declaration with WHAT(2) < `0.0` is recommended (p. 125). Also, it is recommended to request an option DEFAULTS with SDUM = CALORIMEtry, ICARUS or PRECISIOn (p. 83).
2. In many cases, binnings defined by EVENTBIN result in a number of sparse "hit" cells, while all other bins are empty (scoring zero). In such cases, it is convenient to print to file only the content of non-empty bins. In these circumstances, it may also be convenient to allocate a reduced amount of storage (see option ROTPRBIN, p. 192), and in particular the Note 3 to that option.

Example 1:

```
*...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....
EVENTBIN        10.0     208.0      25.0     150.0     200.0     180. Firstscore
EVENTBIN      -150.0     100.0     -20.0      75.0      50.0     20.0  &
*   In the above example, the user requests an event-by-event scoring of
*   energy deposition (generalised particle 208), in a Cartesian
*   three-dimensional array. The bins are 4 cm wide in x (75 bins between
*   x = -150 and x = 150), 2 cm wide in y (50 bins between y = 100 and
*   y = 200), and 10 cm wide in z (20 bins between z = -20 and z = 180).
*   The results are written, formatted, on logical unit 25. The name given
*   to the binning is "Firstscore".
```

Example 2:

```
* Event-by-event scoring of photon fluence in a cylindrical mesh of
* 1200x3800 bins 1 mm wide in R and Z. Results are written unformatted on
* logical unit 27. The users requests not to print bins with zero content.
* The binning name is "Bigcylindr".
*...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....
EVENTBIN       -11.0       7.0     -27.0     600.0       0.0   1900.0 Bigcylindr
EVENTBIN         0.0       0.0       0.0    1200.0       0.0   3800.0  &
```

7.26 EVENTDAT

For calorimetry only.
Prints event by event the scored star production and/or energy deposition in each region, and the total energy balance. energy deposition star production

See also EVENTBIN, SCORE

EVENTDAT requests separate scoring by region of energy and/or star density for each event (primary history).
The quantities to be scored are defined via a SCORE command (see SCORE for details, p. 195).
As for SCORE, a maximum per run of 4 different energy or star densities is allowed.
The EVENTDAT output includes also a detailed energy balance event by event.

WHAT(1) = output unit. If WHAT(1) $<$ 0.0, the output is unformatted. Values of |WHAT(1)| $<$ 21 should be avoided (with the exception of +11).
Default = 11 (standard output)

WHAT(2) – **WHAT(6)**: not used

SDUM = output file name (no default!)

Default (option EVENTDAT not given): no event by event scoring

Notes

1. Unformatted data are written as follows.
   Once, at the beginning of the run:
   `RUNTIT, RUNTIM, NREGS, NSCO, (ISCORE(IS), IS = 1, NSCO)`
   Then, for each primary particle:
   - `NCASE, WEIPRU, ENETOT`
   - `(ENDIST(IE), IE = 1, 12)`
   - Then, `NSCO` times:
     - `(ISC, ISCORE(ISC)`
     - `(REGSCO(IR,ISC), IR = 1, NREGS)`
   - one dummy record (for historical reasons):
     `NDUM, DUM1, DUM2`
   - `ISEED1, ISEED2, SEED1, SEED2, SOPP1, SOPP2`

   where:
   `RUNTIT` = title of the run (`CHARACTER*80` variable), which appears also at the beginning of the standard output
   `RUNTIM` = time of the run (`CHARACTER*32` variable), which appears also at the beginning of the standard output
   `NREGS` = number of regions
   `NSCO` = number of scoring distributions requested by SCORE
   `ISCORE(I)` = Ith requested (generalised) particle distribution (see 5.1)
   `NCASE` = number of primaries handled so far (current one included)
   `WEIPRU` = primary weight

ENETOT = primary particle total energy (GeV)

ENDIST(I) = 12 energy contributions to the total energy balance, some of which appear also at the end of the standard output. Here they are given separately for each primary history (in GeV) and *not* normalised to the weight of the primary. Note that some of the contributions are meaningful only in specific contexts (e.g., if low-energy neutron transport has been requested).

- ENDIST(1) = energy deposited by ionisation
- ENDIST(2) = energy deposited by π^0, electrons, positrons and photons
- ENDIST(3) = energy deposited by nuclear recoils and heavy fragments
- ENDIST(4) = energy deposited by particles below threshold
- ENDIST(5) = energy leaving the system
- ENDIST(6) = energy carried by discarded particles
- ENDIST(7) = residual excitation energy after evaporation
- ENDIST(8) = energy deposited by low-energy neutrons (kerma, proton recoil energy not included)
- ENDIST(9) = energy of particles out of the time limit
- ENDIST(10) = energy lost in endothermic nuclear reactions (gained in exothermic reactions if < 0.0) above 20 MeV (not implemented yet)
- ENDIST(11) = energy lost in endothermic low-energy neutron reactions (gained in exothermic reactions if < 0.0) (not implemented yet)
- ENDIST(12) = missing energy

REGSCO(IR,ISC) = energy or stars (corresponding to the ISCth generalised particle distribution) deposited or produced in the IRth region during the current primary history. *not* normalised, neither to the the primary weight nor to the region volume.

NDUM, DUM1, DUM2 = three dummy variables, with no meaning

ISEED1, ISEED2, SEED1, SEED2, SOPP1, SOPP2 = random number generator information to be read in order to reproduce the current sequence (skipping calls, see option RANDOMIZe, p. 186).

2. All the above quantities are written in single precision (REAL*4), except RUNTIT and RUNTIM (which are of type CHARACTER) and those with a name beginning with I,J,K,L,M,N (which are integer).
3. The different items appearing in the EVENTDAT energy balance may sometimes give overlapping information and are not all meaningful in every circumstance (for instance residual excitation energy is meaningful only if gamma deexcitation has not been requested). Unlike the balance which is printed at the end of standard output, these terms are not additive.
4. An example on how to read EVENTDAT unformatted output is given below.

```
      PROGRAM RDEVDT
      CHARACTER*80 RUNTIT, FILNAM
      CHARACTER*32 RUNTIM
      DIMENSION ISCORE(4), ENDIST(12), REGSCO(5000,4)

      WRITE(*,*) 'Name of the EVENTDAT binary file?'
      READ(*,'(A)') FILNAM
      IB = INDEX(FILNAM,' ')
      OPEN(UNIT = 7, FORM = 'UNFORMATTED', FILE = FILNAM(1:IB-1),
     &     STATUS = 'OLD')
      OPEN(UNIT = 8, FORM = 'FORMATTED', FILE = FILNAM(1:IB-1)//'.txt',
     &     STATUS = 'NEW')

*     Once, at the beginning of the run:
      READ(7)  RUNTIT, RUNTIM, NREGS, NSCO, (ISCORE(IS), IS = 1, NSCO)
```

```
            WRITE(8,'(A80)') RUNTIT
            WRITE(8,'(A32)') RUNTIM
            WRITE(8,'(A,I6,5X,A,I4)') 'Number of regions: ', NREGS,
           &           ' Number of scored quantities: ', NSCO
            WRITE(8,'(A,4I6)') 'The scored quantities are: ',
           &           (ISCORE(IS), IS = 1, NSCO)

      *     Loop on each primary particle:
       100  CONTINUE
            WRITE(8,*)
            READ(7,END=300) NCASE, WEIPRU, ENETOT
            WRITE(8,'(A,I10,1P,2G12.4)') 'NCASE, WEIPRU, ENETOT: ',
           &               NCASE, WEIPRU, ENETOT
            READ(7) (ENDIST(IE), IE = 1, 12)
            WRITE(8,'(A)') 'ENDIST: '
            DO 400 IE = 1, 12, 2
               WRITE(8,'(2(I5,5X,1P,G12.4))') IE,ENDIST(IE),IE+1,ENDIST(IE+1)
       400  CONTINUE
            DO 200 ISC = 1, NSCO
               READ(7) IISC, ISCORE(ISC)
      *        IISC is redundant, must be equal to ISC
               IF(IISC .NE. ISC) STOP 'Wrong sequence'
               WRITE(8,'(A,I2,A,I3,A)')
           &        'Quantity n. ',ISC, ' (',ISCORE(ISC),'):'

               READ(7) (REGSCO(IR,ISC), IR = 1, NREGS)
               WRITE(8,*) 'Scoring per region:'
               DO 500 IR = 1, NREGS
                  WRITE(8,'(I7,3X,1P,G12.4)') IR, REGSCO(IR,ISC)
       500     CONTINUE
       200  CONTINUE

            READ(7) NDUM, DUM1, DUM2
            IF (DUM1 .LT. 0.) THEN
      *        DUM1 < 0 is used to signal that seeds follow
               READ(7) ISEED1, ISEED2, SEED1, SEED2, SOPP1, SOPP2
               WRITE(8,*) ISEED1, ISEED2, SEED1, SEED2, SOPP1, SOPP2
            ELSE
               BACKSPACE 7
            END IF
      *     This event is finished, start again with the next one
            GO TO 100

       300  CONTINUE
            WRITE(8,*) "End of a run of ", NCASE, " particles"
            CLOSE (UNIT = 7)
            CLOSE (UNIT = 8)
            END
```

Example:

```
*...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+...
SCORE           208.      211.      201.        8.        0.       0.
EVENTDAT        -23.        0.        0.        0.        0.       0. EVT.SCOR
*  In this example, the user requests (with option SCORE) scoring of
*  total and electromagnetic energy deposition, total stars and
*  neutron-produced stars. The average scores for each region will be
*  printed on standard output (as an effect of SCORE command), and
*  corresponding scores, as well as the energy balance, will be written
*  separately for each primary particle on an unformatted file EVT.SCOR
```

7.27 EVENTYPE

Defines the hadronic particle production model to be used

FLUKA is designed to work with different particle production models. The SDUM parameter of option EVENTYPE is used to indicate the desired hadronic event generator. However, only one generator (EVAP) is available at present in the standard FLUKA version.
This option is intended only for code development and should not be requested by the normal user, except to activate heavy recoil transport (see WHAT(3) and/or ion-ion interactions (see SDUM = DPMJET)).

The meaning of **WHAT(1) ... WHAT(6)** depends on the model chosen.

WHAT(1), WHAT(2): reserved for program development

WHAT(3) < -1.0: resets the default (no ion transport at all)
-1.0 < WHAT(3) < 1.0: ignored
= 1.0: approximated transport of ions and recoils (dE/dx only)
= 2.0: all heavy recoils and ions are transported with energy loss and multiple scattering, without nuclear interactions if SDUM = blank or = EVAP, with nuclear interactions if SDUM = DPMJET
3.0 ≤ WHAT(3) ≤ 6.0: heavy recoils up to |particle id| = WHAT(3) are transported with energy loss and multiple scattering, but no nuclear interactions (3 = d, 4 = t, 5 = ^{3}He, 6 = ^{4}He)

Default = 1.0 if option DEFAULTS is not used, or is used with SDUM = CALORIMEtry, EET/TRANsmut, HADROTHErapy, ICARUS, NEW–DEFAults, PRECISIOn or SHIELDINg. With any other value of SDUM in option DEFAULTS, they are not transported.

WHAT(4) – **WHAT(6)**: reserved for program development

SDUM = EVAP : DPM model + PEANUT + evaporation/fission + gamma deexcitation
= DPMJET : same as EVAP, plus heavy ion nuclear interactions (provided the DPMJET library has been linked). The value of WHAT(3) defaults to 2.0

Default :EVAP

Default (option EVENTYPE not given): defaults are as explained for WHAT(3) above.
See also Table 7.1 at p. 88.

Note

1. Option EVENTYPE is mainly for development purposes and should not be used for routine work except (temporarily) to request transport of heavy recoils. See option DEFAULTS in order to set up suitable defaults for different classes of problems.

Example 1:

```
*...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....
EVENTYPE          0.        0.        2.        0.        0.        0.EVAP
*  In this example, the user requests transport of heavy ions without
*  nuclear interactions
```

Example 2:

```
EVENTYPE          0.        0.        2.        0.        0.        0.DPMJET
*  The user requests transport of heavy ions with nuclear interactions
```

Example 3:

```
EVENTYPE          0.        0.        5.        0.        0.        0.EVAP
*  The user requests transport of deuterons, tritons and 3He, but not
*  of alphas and heavier ions.
```

7.28 EXPTRANS

Requests an exponential transformation ("path stretching")

NOT YET IMPLEMENTED!!!

WHAT(1) $\geq$ `0.0`: WHAT(2)...WHAT(5) are used to input a parameter defining the exponential transformation in various regions

$<$ `0.0`: WHAT(2)...WHAT(5) are used to define the particles to which the exponential transformation must be applied

Default = `0.0`

For WHAT(1) $\geq$ `0.0`:

WHAT(2) = exponential transformation parameter η (`0.0` $\leq |\eta| <$ `1.0`).
This value can be overridden in user routine `UBSSET` (p. 320) by assigning a value to variable `EXPTR`

Default = `0.0`

WHAT(3) = lower bound of the region indices with exponential transformation parameter η = WHAT(2)
("*From region WHAT(3)...*")

Default = `2.0`

WHAT(4) = upper bound of the region indices with exponential transformation parameter equal to WHAT(2)
("*...to region WHAT(4)...*")

Default = WHAT(3)

WHAT(5) = step length in assigning indices.
("*...in steps of WHAT(5)*")

Default = `1.0`

WHAT(6) and **SDUM**: not used

For WHAT(1) $<$ `0.0`:

WHAT(2) = lower bound of the particle indices to which exponential transformation is to be applied
("*From particle WHAT(2)...*")

Default = `1.0`

WHAT(3) = upper bound of the particle indices to which exponential transformation is to be applied
("*...to particle WHAT(3)...*")

Default = WHAT(2)if WHAT(2) $>$ `0.0`, otherwise = `40.0` (low-energy neutrons)

WHAT(4) = step length in assigning indices.
("*...in steps of WHAT(4)*")

Default = `1.0`

WHAT(5), WHAT(6) and **SDUM**: not used

Default (option EXPTRANS not given): no exponential transformation is applied

Example:

```
*...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....
EXPTRANS          1.       0.8       10.       18.        8.       0.
EXPTRANS         -1.        7.        8.        0.        0.       0.
*  Exponential transformation is requested for photons (particle no. 7)
*  and neutrons (particle 8), in regions 10 and 18 with an ETA parameter
*  equal to 0.8
```

7.29 FLUKAFIX

Sets the size of the step of muons and charged hadrons to a fixed fraction of the kinetic energy in different materials

See also EMFFIX, MULSOPT, STEPSIZE

WHAT(1) = fraction of the kinetic energy to be lost in a step (must not be > 0.2)
Default : if option DEFAULTS is used with SDUM = ICARUS, the default is 0.02.
With SDUM = HADROTHErapy or PRECISIOn, the default is 0.05.
If SDUM = CALORIMEtry, the default is 0.08.
With any other SDUM value, or if DEFAULTS is missing, the default is 0.1.

WHAT(2) = "ε" parameter used to check the finite size of the nucleus when the nuclear form factor is not invoked by the multiple scattering algorithm (see Note 3 below)
For code development only, do not change the default!
Default = 0.15

WHAT(3) = high-energy limit for the fraction of energy to be lost in a step (the fraction is given by WHAT(3) × WHAT(1))
For code development only, do not change the default!
Default = 0.012

WHAT(4) = lower index bound of materials where the specified energy loss per step is to be applied. ("*From material WHAT(4)...*")
Default = 3.0

WHAT(5) = upper index bound of materials where the specified energy loss per step is to be applied. ("*...to material WHAT(5)...*")
Default = WHAT(4)

WHAT(6) = step length in assigning indices. ("*...in steps of WHAT(6)*")
Default = 1.0

SDUM : not used

Default (option FLUKAFIX not given): the defaults listed above apply. See also Table 7.1 at p. 88.

Notes

1. Usually there is no need for changing the default value of 10 % (0.1) for WHAT(1).
2. The input value is actually applied as such only at intermediate energies (between about a few tens of MeV and 1 GeV): at low energies it is slowly increased to twice the requested value, while at high energies it decreases to a limit controlled by WHAT(3), usually about $\frac{1}{100}$ of the input value.
3. The "ε" parameter controls the accuracy of the multiple scattering algorithm by limiting the step. In most cases the length of the step is practically limited anyway by the hadron interaction length, so the "ε" default value is of little importance and does not need to be changed.
However, in some problems of large dimensions, especially when transporting muons, a small value of ε can slow down the calculations without necessity. In such cases, WHAT(2) can be safely set = 1000.0.

Example:

```
*...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....
FLUKAFIX        0.03        0.          0.        21.         0.         0.
*  The maximum fractional energy loss for hadrons and muons is set to
*  3 percent in material 21.
```

7.30 FREE

Activates free-format input

See also GLOBAL

WHAT(1) – **WHAT(6)** and **SDUM** are not used.

Default (option FREE not given): input must follow the standard FLUKA format `(A8,2X,6E10.0,A8)` (see Chap. 6), unless free format has been chosen via a GLOBAL command (7.33)

Notes

1. FREE can appear at any point of FLUKA input except between GEOBEGIN and GEOEND (not included) and in input for PLOTGEOM. But in general it should be given as the first command or just after a TITLE card. From then on, all successive input (except the geometry) will be read in free format. That means that keywords, WHATs and SDUMs do not need to be aligned in columns but can be written anywhere on the input line, alternating with "separators" in a manner similar to that of list-oriented format in Fortran (but character strings — keywords and SDUMs — must *not* be put between quotes!). A separator is any one of the following:
   - One of the five characters , (comma), ; (semicolon), / (slash), \ (backslash), : (colon), preceded or not by one or more blanks, and followed or not by one or more blanks
   - One or more successive blanks without any non-blank separator
2. Different separators may be used on the same line.
3. If a non-blank separator is immediately followed by another one (with or without blanks in between), a value of `0.0` is assumed to be given between them.
4. Zero values must be given explicitly or as empty places between two non-blank separators as explained above.
5. Geometry input (i.e., input between GEOBEGIN and GEOEND cards not included, see Chap. 8) must still respect the column format described in Chapter 8.
6. `PLOTGEOM` input, whether in a separate file `PLG.GFXINDAT` or directly after a PLOTGEOM card in standard input, must still be formatted as shown in the description of that option (see p. 179).
7. If FREE has been issued, from then on all constants must be written without any blank imbedded (e.g., `5.3 E-5` is not allowed, but must be written `5.3E-5` or `5.30E-5`)
8. Free format, if requested, applies to option cards of the form
   KEYWORD WHAT(1) WHAT(2) ..... WHAT(6) SDUM
   but not to any data card specific to certain options (for instance cards following COMMENT or TITLE)
9. Free format can be requested also by option GLOBAL (p. 125), but extended to the whole input and not only from the point where the command is issued. GLOBAL can also be used to request free format geometry input.

Example:

```
The following fixed-format input line:
*...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+...
BEAM       20.                      0.0   -1.0  E-2     -0.02    1.0   PION+
* can be given in free format in any of the following equivalent ways:
*...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+...
FREE
BEAM  2. 0.0      0.0   -1.0E-2  -0.02 1.0  PION+
* ...equivalent to:
*...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+...
FREE
BEAM, 2.,0.0   ,  0.0,  -1.0E-2; -0.02 1.0 /PION+   ! 2 GeV/c momentum?
* (note the possibility to insert comments at the end of the line)
* ...and also to:
*...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+...
FREE
BEAM   : 20. ,,  , -1.0E-2\   -0.02 1.0               PION+
* etc...
```

7.31 GEOBEGIN

Starts the geometry description

See also GEOEND, PLOTGEOM

WHAT(1) $\neq$ 0.0: only a global summary of tracking errors is printed
Default = 0.0 (all geometry error messages are printed)

WHAT(2) : used to set the accuracy parameter — reserved for program development

WHAT(3) $>$ 0.0: logical unit for geometry input. The name of the corresponding file must be input on the next card if WHAT(3) is different from 5.0. Note that values of WHAT(3) $\neq$ 5.0 and $<$ 21.0 must be avoided because of possible conflicts with FLUKA pre-defined units.
Default = 5.0 (i.e., geometry input follows)

WHAT(4) $>$ 0.0: logical unit for geometry output. If different from 11, the name of the corresponding file must be input on the next card if WHAT(3) = 0.0 or 5.0, otherwise on the card following the next one. Values of WHAT(3) $\neq$ 11.0 and $<$ 21.0 must be avoided because of possible conflicts with FLUKA pre-defined units.
Default = 11.0 (i.e., geometry output is printed on the standard output)

WHAT(5), WHAT(6): not used

SDUM = COMBINAT: Combinatorial Geometry is used. See Chapter 8 on Combinatorial Geometry for input description. Other geometries, available in older versions of FLUKA, are no longer supported.
Default : COMBINAT

Default (option GEOBEGIN not given): **not allowed!** GEOBEGIN and GEOEND must always be present.

Notes

1. Geometry input and output:
   i) If WHAT(3) and WHAT(4) are both $\leq$ 0.0, the geometry input and output are part of the standard I/O streams: the GEOBEGIN card must be immediately followed by the Combinatorial Geometry input, and then by GEOEND. (See Example 1 below).
   ii) If WHAT(3) is $>$ 0.0 (and not = 5.0) and WHAT(4) is $\leq$ 0.0, GEOBEGIN must be followed by one card with the name of the CG input file, and then by GEOEND. (See Example 2 below).
   iii) If WHAT(4) is $>$ 0.0 (and not = 11.0) and WHAT(3) is $\leq$ 0.0, GEOBEGIN must be followed by one card with the name of the CG output file, then by the CG input, and then by GEOEND. (See Example 3 below).
   iv) Otherwise, if WHAT(3) is $>$ 0.0 (and not = 5.0), and WHAT(4) is $>$ 0.0 (and not = 11.0), GEOBEGIN must be followed, in the order, by a line with the name of the CG input file, another line with the name of the CG output file and then the GEOEND card. (See Example 4 below).

   The first part of the CG output (p. 273) is essentially an echo of the input, and gives in addition some information on storage allocation of body and region data. Then follow an "Interpreted body echo" and an "Interpreted region echo"', useful in the case where free geometry input format has been requested. It is recommended to check on this output that the geometry description has been correctly reproduced, especially if fixed input format has been chosen, since column alignment errors are critical due to the CG strict input format. In case of severe error ("next region not found", or similar) the CG output will also contain all error

and debugging messages. Minor tracking problems, however, are reported on the error message file (logical unit 15, see Chap. 3 and Sec. 9.4), unless reporting has been de-activated by setting WHAT(1) $\neq$ 0.0.

2. Special algorithms have been implemented in the Combinatorial Geometry package of FLUKA in order to minimise tracking errors due to rounding, to improve tracking speed and to handle charged particle transport near boundaries and in magnetic fields.
3. A voxel geometry is also available in FLUKA (8.2.11). It is fully integrated into the FLUKA CG and it is even possible to describe a geometry partly as made of voxels and partly of conventional CG regions.
4. Information on how to set up Combinatorial Geometry input is given in Chap. 8.
5. End of geometry information must end with a GEOEND card (p. 123). The same card can also be used to activate a geometry debugger.
6. Option PLOTGEOM (p. 179) allows to draw sections of the geometry on planes specified by the user.
7. Only one GEOBEGIN card is allowed.

Example 1:

```
* CG Input follows, output is printed as part om Main Output
*...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+...
GEOBEGIN          0.        0.        0.        0.        0.        0.COMBINAT
```

Example 2:

```
* CG Input read from file BigHall.geo, output printed as part om Main Output
*...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+...
GEOBEGIN          0.        0.       25.        0.        0.        0.COMBINAT
BigHall.geo
```

Example 3:

```
* CG Input follows, output is printed on file geo2.out
*...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+...
GEOBEGIN          0.        0.        0.       26.        0.        0.COMBINAT
geo2.out
```

Example 4:

```
* CG Input read from file BigHall.geo,, output printed on file geo2.out
*...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+...
GEOBEGIN          0.        0.       25.       26.        0.        0.COMBINAT
BigHall.geo
geo2.out
```

7.32 GEOEND

Ends the geometry definitions.
This option can also be used to debug the geometry.

See also GEOBEGIN, PLOTGEOM

Normally, only one GEOEND card is issued, at the end of the geometry description, with all the WHATs and the SDUM equal to zero or blank. However, GEOEND can also be used to activate the FLUKA geometry debugger: in this case one or two GEOEND cards are needed, providing the information described below.

First GEOEND card:

WHAT(1) = $\mathbf{X}_{\max}$ of the geometry region to be debugged (no default)

WHAT(2) = $\mathbf{Y}_{\max}$ of the geometry region to be debugged (no default)

WHAT(3) = $\mathbf{Z}_{\max}$ of the geometry region to be debugged (no default)

WHAT(4) = $\mathbf{X}_{\min}$ of the geometry region to be debugged (no default)

WHAT(5) = $\mathbf{Y}_{\min}$ of the geometry region to be debugged (no default)

WHAT(6) = $\mathbf{Z}_{\min}$ of the geometry region to be debugged (no default)

SDUM = DEBUG to activate the debugger, otherwise must be left blank

Second (optional) GEOEND card:

WHAT(1) = Number of mesh intervals in the **X**-direction between $\mathbf{X}_{\min}$ and $\mathbf{X}_{\max}$
Default = 20.0

WHAT(2) = Number of mesh intervals in the **Y**-direction between $\mathbf{Y}_{\min}$ and $\mathbf{Y}_{\max}$
Default = 20.0

WHAT(3) = Number of mesh intervals in the **Z**-direction between $\mathbf{Z}_{\min}$ and $\mathbf{Z}_{\max}$
Default = 20.0

WHAT(4) – **WHAT(6)**: not used

SDUM = "&" in any position in column 71 to 78 (or in the last field if free format is used)

Default (option GEOEND not given): **not allowed!** GEOBEGIN and GEOEND must always be present.

Notes

1. The geometry debugger can detect both undefined points (points which are not included in any defined region) and multiple defined points (points which are included in more than one region (i.e., there are overlapping regions) in the selected X,Y,Z mesh. The first kind of error is likely to cause a run-time error every time a particle is passing through the undefined zone, the second one is more subtle and it is not usually detected at run-time. It is impossible to predict to which actual region such multiple defined points will be assigned.
2. The geometry debugger cannot assure that a bug-free geometry input is what the user would like to describe. Another useful tool is available for this purpose: the PLOTGEOM program, which is activated by means of the PLOTGEOM command (7.56).
3. It must be stressed that only the points of the defined X,Y,Z mesh are checked, therefore mesh dimensions and pitches should be chosen according to the present geometry, taking into account region thicknesses etc.

4. It must be stressed too that the geometry debugger can be very time consuming, so don't ask for 100 μm pitches in X,Y,Z over a 10 m distance, or the code will run forever! Make use as much as possible of geometry symmetries (for example for a cylindrical geometry there is no need for a 3-D scan) and possibly "zoom" with fine meshes only around points where you suspect possible errors. Note that as many areas as wished can be scanned with different meshes of the *same* geometry, simply changing the mesh parameters each time.
5. **Warning**: the program does not stop if an error is detected but a message is issued on the output units, and checking goes on. If the code is "stepping" into an area erroneously defined, it is likely that plenty of such error messages will be printed. If your operating system allows inspection of output files before they are closed, check the size of your output from time to time. If it is growing too much, stop the code and correct the geometry for the printed errors before restarting the debugger.

Example of a normal GEOEND card without debugging:

```
*...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+...
GEOEND           0.        0.        0.        0.        0.        0.
*   The next is an example of geometry debugging:
GEOEND         150.       75.      220.       30.        0.     -220.DEBUG
GEOEND         120.        1.      110.        0.        0.        0. &
*   The above cards request a scan of the geometry portion delimited
*   by planes x = 30, x = 150, y = 0, y = 75, z = -220, z = 220,
*   with 120 steps 1-cm wide along x, one single step along y and
*   110 4-cm wide steps along z. A single step in one direction (here y)
*   is typical of searches through cylindrically symmetric geometries.
```

7.33 GLOBAL

Makes a global declaration about the present run, setting some important parameters that must be defined before array memory allocation

See also DEFAULTS, FREE

Important: GLOBAL declarations, if present, must precede any executable option

WHAT(1) = maximum number of regions (must be $\leq$ 10000)
Default = 1000

WHAT(2) = declaration of "how analogue" this run must be: fully analogue, as biased as possible, or automatically chosen by the program?

$<$ 0.0: as analogue as possible (provided the input is consistent with this choice)

$>$ 1.0: as biased as possible (allowed also for a run in which no explicit biasing option is requested: in this case it simply means "do not try to be analogue")

0.0 $\leq$ WHAT(2) $<$ 1.0: as analogue as decided by the program according to the selected biasing options

Default = 0.0(input decides the amount of biasing)

WHAT(3) = declaration about the use of the DNEAR variable (see Note 4) when computing physical steps:

$<$ 0.0: always use DNEAR when computing the tentative length of particle steps (it can cause non reproducibility of the random number sequence when starting from different histories, but it does not affect physics)

$>$ 0.0: do not use DNEAR when computing the tentative length of particle steps (full reproducibility of the random number sequence starting from different histories, some penalty in CPU time)

= 0.0: use DNEAR when computing the tentative length of particle steps only when the random number sequence reproducibility is assured (full reproducibility of random number sequence within the same geometry package, possible non reproducibilities among different geometry packages describing the same geometry)

Default = 0.0 (random number sequence reproducible within the same geometry package)

WHAT(4) flag to request various types of input

$<$ 0.0: resets the default

= 0.0: ignored

= 1.0: **(not yet implemented)**
requests the use of names (alphanumerical 8-character strings beginning by alphabetical) instead of numbers as identifiers of particles, materials and regions in the relevant "WHAT" fields of input commands. If fixed format is used, each name must be contained inside the corresponding 10-character field. If free format geometry input has not been requested (see WHAT(5)) the region names, generated automatically by FLUKA, can be found on standard output.

= 2.0: requests free-format input for all input commands (for geometry body and region input, see WHAT(5)). The six "WHAT" fields must all be input, or replaced by two successive separators (together with zero or more blanks)

= 3.0: **(only partially implemented)** the two previous options are both requested, i.e., alphanumerical 8-character names are used to identify particles, materials and regions in the relevant "WHAT" fields of input commands, and free format is also used (for geometry body and region input, see WHAT(5)). The six "WHAT" fields must all be input, or replaced by two successive separators (together with zero or more blanks)

WHAT(5) : flag to request free format in the geometry input for bodies and regions. This format is described in 8.2.3.2 and 8.2.6.3, and requires the use of names (alphanumerical 8-character strings beginning by alphabetical) as identifiers. Parentheses are allowed.
$<$ 0.0: resets the default
$=$ 0.0: ignored
$>$ 0.0: geometry input for bodies and regions will be in free format

WHAT(6) and **SDUM**: not used

Notes

1. In most cases the user should not worry about the number of geometry regions. Despite the fact that FLUKA input does not follow any specific order, the program is able to manage initialisation of all geometry-dependent arrays by allocating temporary areas of memory even before the actual dimensions of the problem are known. The unused parts of such areas are recovered at a later time when the whole input has been read. However, if the number of regions is very large ($>$ 1000), the program needs to be informed in order to increase the size of such temporary areas. This information must be given at the very beginning: therefore GLOBAL (together with DEFAULTS, MAT–PROP and PLOTGEOM) is a rare exception to the rule that the order of FLUKA input cards is free.
2. The "hard" limit of 10000 regions represents the maximum that can be obtained without recompiling the program. It can be overridden, but only by increasing the value of variable MXXRGN in the INCLUDE file DIMPAR and recompiling the whole code. In this case, however, it is likely that the size of variable NBLNMX in INCLUDE file BLNKCM will have to be increased too.
3. In a "fully analogue" run, each interaction is simulated by sampling each exclusive reaction channel with its actual physical probability. In general, this is not always the case, especially concerning low-energy neutron interactions. Only issuing a GLOBAL declaration with WHAT(2) $<$ 0.0 can it be ensured that analogue sampling is systematically carried out whenever it is possible. The lack of biasing options in input is not sufficient for this purpose. This facility should be used in problems where fluctuations or correlations cannot be ignored, but it is likely to lead to longer computing times.
4. DNEAR designates the distance between the current particle position and the nearest boundary (or a lower bound to that distance), and it is used by FLUKA to optimise the step length of charged particles. The concept and the name have been borrowed from the EGS4 code [116], but the FLUKA implementation is very different because it is fully automatic rather than left to the user, and it is tailored for Combinatorial Geometry, where a region can be described by partially overlapping sub-regions (described in input by means of the OR operator). The sequential order in which overlapping sub-regions are considered when evaluating DNEAR is irrelevant from the point of view of particle tracking, but can affect the random number sequence. This does not have any effect on the average results of the calculation, but the individual histories can differ due the different random number sequence used. Option GLOBAL can be used in those cases where the user wants to reproduce exactly each particle history, or on the contrary to forgo it in order to get a better step optimisation.
5. Free format can be requested also by option FREE, but only for the part of input that follows. FREE cannot be used to request free format geometry input. See the Notes to FREE for the rules governing separators (p. 120).

Example 1:

```
*...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+...
TITLE
   A fully analogue run (no other commands precede this TITLE card)
GLOBAL         2000.       -1.        1.        0.        0.        0.
*  This run needs more than the default maximum number of regions. It is
*  requested to be as analogue as possible and to avoid using DNEAR if
*  it risks to affect the random number sequence.
```

Example 2:

```
*...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+...
TITLE
    Full free-format input (no other commands precede this TITLE card)
GLOBAL           0.0       0.0       0.0       2.0       1.0       0.
*  The following input will be all in free format (both the FLUKA commands
*  and the geometry description)
```

7.34 HI–PROPErt

Specifies the properties of a heavy ion primary

See also BEAM

WHAT(1) = Atomic number Z of the heavy ion
Default = `6.0` (Carbon)

WHAT(2) = Mass number A of the heavy ion
Default = `12.0`

WHAT(3) = Excitation energy of the heavy ion above ground state
(not yet implemented)
Default = `0.0`

WHAT(4) – **WHAT(6), SDUM**: not used

SDUM : not used

Default (option HI–PROPErt not given): a HEAVYION projectile is assumed to be ^{12}C in the ground state.

Note

1. Option HI–PROPErt is used to specify the properties of a generic heavy ion primary declared by a BEAM command (p. 64) with SDUM = HEAVYION, or by a user-written subroutine `SOURCE` (p. 314) with id-number `IJ = -2`.

Example:

```
*...+....1....+....2....+....3....+....4....+....5....+....6....+....7...
*  Primary particles are 10 GeV Au-197 ions in the ground state:
BEAM          -10.0       0.0       0.0       0.0       0.0       0. HEAVYION
HI-PROPE       79.0     197.0       0.0       0.0       0.0       0.
```

7.35 IONFLUCT

Calculates ionisation energy losses of charged hadrons, muons, and electrons/positrons with ionisation fluctuations.

See also DELTARAY

WHAT(1) $\geq$ `1.0`: switches on restricted energy loss fluctuations for hadrons and muons
$\leq$ `-1.0`: switches off restricted energy loss fluctuations for hadrons and muons
$=$ `0.0`: ignored
Default : restricted energy loss fluctuations for hadrons and muons are activated if option DEFAULTS is missing or if it is used with SDUM = CALORIMEtry, EET/TRANSmut, HADROTHErapy, ICARUS, NEW–DEFAults or PRECISIOn.
With any other SDUM value, they are not activated.

WHAT(2) $\geq$ `1.0`: switches on restricted energy loss fluctuations for electrons and positrons
$\leq$ `-1.0`: switches on restricted energy loss fluctuations for electrons and positrons
$=$ `0.0`: ignored
Default : restricted energy loss fluctuations for electrons and positrons are activated if option DEFAULTS is missing or if it is used with SDUM = CALORIMEtry, EM–CASCAde, HADROTHErapy, ICARUS, NEW–DEFAults or PRECISIOn.
With any other SDUM value, they are not activated.

WHAT(3) : If WHAT(1) $\geq$ `1.0` (resp. WHAT(2) $\geq$ `1.0`), WHAT(3) represents the accuracy parameter for the ionisation fluctuation algorithm [57] (see 1.2.1.4) for hadrons and muons (resp. electrons and positrons). The accuracy parameter can take integer values from `1` to `4` (corresponding to increasing levels of accuracy)
$<$ `0.0`: resets to default
Default = `1.0` (minimal accuracy)

WHAT(4) = lower bound of the indices of the materials in which the restricted energy loss fluctuations are activated
("*From material WHAT(4)...*")
Default = `3.0`

WHAT(5) = upper bound of the indices of the materials in which the restricted energy loss fluctuations are activated
("*... to material WHAT(5)...*")
Default = WHAT(4)

WHAT(6) = step length in assigning indices
("*... in steps of WHAT(6)*")
Default = `1.0`

SDUM : not used

Default (option IONFLUCT not given): ionisation fluctuations are simulated or not depending on option DEFAULTS as explained above.
See also Table 7.1 at p. 88.

Note

1. The energy loss fluctuation algorithm is fully compatible with the DELTARAY option (p. 89). (See Example below).

Example:

```
*...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+...
IONFLUCT         0.0       1.0       3.0       7.0      16.0       3.0
IONFLUCT         1.0       0.0       2.0       8.0      10.0       2.0
DELTARAY       1.E-3       0.0       0.0      10.0      11.0
*  The special FLUKA algorithm for ionisation fluctuations is activated
*  with accuracy level 3 for photons and electrons in materials 7, 10, 13 and
*  16. The same algorithm is activated, at an accuracy level = 2, for
*  materials 8 and 10, but in the latter material only for ionisation losses
*  with energy transfer < 1 MeV. Losses with larger energy transfer will
*  result in explicit delta electron production. In material 11, delta rays
*  will be produced if the energy transfer is larger than 1 MeV, but
*  fluctuations for lower energy transfers will be ignored.
```

7.36 IRRPROFIle

defines an irradiation profile for radioactive decay calculations

See also DCYTIMES, DCYSCORE, RADDECAY, RESNUCLEi

WHAT(1) : length of a newly defined irradiation interval (in s)
$>$ 0.0: a new interval is added with length WHAT(1)
$=$ 0.0: ignored
$<$ 0.0: the last defined interval (if any) is deleted
Default : no new irradiation interval is defined

WHAT(2) : beam intensity of the newly defined (see WHAT(1)) irradiation interval
$\geq$ 0.0: beam intensity in particles/s (0.0 is accepted)
$<$ 0.0: considered as 0.0
Default = 0.0 particles/s

WHAT(3) : the same as WHAT(1)

WHAT(4) : the same as WHAT(2)

WHAT(5) : the same as WHAT(1)

WHAT(6) : the same as WHAT(2)

SDUM : not used

Default (option IRRPROFIle not given): no irradiation interval is defined

Note

1. Several cards can be combined up to a maximum of 20 irradiation intervals. Decay times as requested by DCYTIMES commands (p. 82) will be calculated from the end of the last one. Scoring during irradiation can be obtained giving negative decay times in DCYTIMES.

Example:

```
*...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
IRRPROFILE     1800.    1.5E12      250.     3.E10     4500.    4.2E12
* The profile defined consists of 1800 s of irradiation at an intensity of
* 1.5E12 particles/s, followed by 250 s at low intensity (3.E10 particles/s),
* and then a third 4500 s interval at 4.2E12 particles/s
```

7.37 LAM–BIAS

Used to bias the decay length of unstable particles, the inelastic nuclear interaction length of hadrons, photons and muons and the direction of decay secondaries

See also EMF–BIAS

The meaning of **WHAT(1)...WHAT(6)** depends on the value of **SDUM**.

SDUM = DCDRBIAS and SDUM = DCY–DIRE are used to activate and define *decay direction biasing*.

SDUM = GDECAY selects *decay length biasing* and *inelastic nuclear interaction biasing*.

If SDUM = blank, *decay life biasing* and *inelastic nuclear interaction biasing* are selected.

Other LAM–BIAS cards with SDUM = DECPRI, DECALL, INEPRI, INEALL allow to restrict biasing to primary particles or to extend it also to further generations.

For SDUM = DCY–DIRE:

The decay secondary product direction is biased in a direction indicated by the user by means of a unit vector of components **U, V, W** (see Notes 4 and 5):

WHAT(1) = **U** (x-direction cosine) of the preferred direction to be used in decay direction biasing

Default = `0.0`

WHAT(2) = **V** (y-direction cosine) of the preferred direction to be used in decay direction biasing

Default = `0.0`

WHAT(3) = **W** (z-direction cosine) of the preferred direction to be used in decay direction biasing

Default = `1.0`

WHAT(4) $>$ `0.0`: **λ** for decay direction biasing. The degree of biasing decreases with increasing lambda (see Note 4)
= `0.0`: a user provided routine (`UDCDRL`, p. 322) will be called at each decay, to provide both direction and lambda for decay direction biasing
$<$ `0.0`: resets to default (**λ** = `0.25`)

Default = `0.25`

WHAT(5), WHAT(6): not used

For SDUM = DCDRBIAS:

WHAT(1) $>$ `0.0`: decay direction biasing is activated (see Notes 4 and 5):
= `0.0`: ignored
$<$ `0.0`: decay direction biasing is switched off

WHAT(2), WHAT(3): not used

WHAT(4) = lower bound of the particle id-numbers for which decay direction biasing is to be applied ("*From particle WHAT(4)...*")

Default = `1.0`

WHAT(5) = upper bound of the particle id-numbers for which decay direction biasing is to be applied ("*... to particle WHAT(5)...*")

Default = WHAT(4) if WHAT(4) > `0.0, 64.0` otherwise

WHAT(6) = step length in assigning numbers ("*... in steps of WHAT(6)*")

Default = `1.0`

For all other SDUMs:

WHAT(1) : biasing parameter for decay length or life, applying only to unstable particles (with particle numbers $\geq$ 8). Its meaning differs depending on the value of SDUM, as described below.

For SDUM = GDECAY:

WHAT(1) < `0.0`: the mean *decay length* (in cm) of the particle in the *laboratory frame* is set = |WHAT(1)| if smaller than the physical decay length (otherwise it is left unchanged). At the decay point sampled according to the biased probability, Russian Roulette (i.e., random choice) decides whether the particle actually will survive or not after creation of the decay products. The latter are created in any case and their weight is adjusted taking into account the ratio between biased and physical survival probability.

WHAT(1) > `0.0`: the mean *decay length* (in cm) of the particle in the *laboratory frame* is set = WHAT(1) if smaller than the physical decay length (otherwise it is left unchanged). Let P_u = unbiased probability and P_b = biased probability: at the decay point sampled according to P_b, the particle always survives with a reduced weight $W(1 - P_u/P_b)$, where W is the current weight of the particle before the decay. Its daughters are given a weight $W \cdot P_u/P_b$ (as in case WHAT(1) < `0.0`).

WHAT(1) = `0.0`: ignored

For SDUM = blank:

`-1.0` < **WHAT(1)** < `0.0`: the mean *life* of the particle in its *rest frame* is *reduced* by a factor = |WHAT(1)|. At the decay point sampled according to the biased probability, Russian Roulette (i.e., random choice) decides whether the particle actually will survive or not after creation of the decay products. The latter are created in any case and their weight adjusted taking into account the ratio between biased and physical survival probability.

`0.0` < **WHAT(1)** < `1.0`: the mean *life* of the particle in the *rest frame* is *reduced* by a factor = |WHAT(1)|. At the decay point sampled according to the biased probability, the particle always survives with a reduced weight. Its daughters are given the same weight.

|WHAT(1)| > `1.0`: a possible previously given biasing parameter is reset to the default value (no biasing)

WHAT(1) = `0.0`: ignored

WHAT(2) : biasing factor for hadronic inelastic interactions

`-1.0` < **WHAT(2)** < `0.0`: the hadronic inelastic interaction length of the particle is reduced by a factor |WHAT(2)|. At the interaction point sampled according to the biased probability, Russian Roulette (i.e., random choice) decides whether the particle actually will survive or not after creation of the secondaries. The latter are created in any case and their weight is adjusted taking into account the ratio between biased and physical survival probability.

`0.0` < **WHAT(2)** < `1.0`: the hadronic inelastic interaction length of the particle is reduced by a factor WHAT(2). At the interaction point sampled according to the biased probability, the particle always survives with a reduced weight. The secondaries are created in any case and their weight adjusted taking into account the ratio between biased and physical survival probability.

WHAT(2) = `0.0`: ignored

|WHAT(2)| $\geq$ `1.0`: a possible previously set biasing factor is reset to the default value of `1.0` (no biasing).

WHAT(3) $>$ `2.0`: number of the material to which the inelastic biasing factor has to be applied.
$<$ `0.0`: resets to the default a previously assigned value
$=$ `0.0`: ignored if a value has been previously assigned to a specific material; otherwise: all materials (default)
`0.0` $<$ **WHAT(3)** $\leq$ `2.0`: all materials.

WHAT(4) = lower bound of the particle id-numbers for which decay or inelastic interaction biasing is to be applied
("*From particle WHAT(4)...*")

Default = `1.0`

WHAT(5) = upper bound of the particle id-numbers for which decay or inelastic interaction biasing is to be applied
("*... to particle WHAT(5)...*")

Default = WHAT(4) if WHAT(4) $>$ `0.0, 46.0` otherwise

WHAT(6) = step length in assigning numbers
("*... in steps of WHAT(6)*")

Default = `1.0`

For SDUM = DECPRI, DECALL, INEPRI, INEALL:

SDUM = DECPRI : decay biasing, as requested by another LAM–BIAS card with SDUM = GDECAY or blank, must be applied only to primary particles.
= DECALL : decay biasing, as requested by another LAM–BIAS card with SDUM = GDECAY or blank, must be applied to all generations (default).
= INEPRI : inelastic hadronic interaction biasing, as requested by another LAM–BIAS card with SDUM = blank, must be applied only to primary particles.
= INEALL : inelastic hadronic interaction biasing, as requested by another LAM–BIAS card with SDUM = blank, must be applied to all generations (default)

Default (option LAM–BIAS not given): no decay length, decay direction, or inelastic interaction biasing

Notes

1. Option LAM–BIAS can be used for three different kinds of biasing:
   (a) biasing of the particle decay length (or life),
   (b) biasing of the direction of the decay secondaries, and
   (c) biasing of the inelastic hadronic interaction length.
2. Depending on the SDUM value, two different kinds of biasing are applied to the particle decay length (or life). In both cases, the particle is transported to a distance sampled from an imposed (biased) exponential distribution: If WHAT(1) is positive, decay products are created, but the particle survives with its weight and the weight of its daughters is adjusted according to the ratio between the biased and the physical survival probability at the sampled distance. If WHAT(1) is negative, decay is performed and the weight of the daughters is set according to the biasing, but the survival of the primary particle is decided by Russian Roulette according to the biasing. Again, the weights are adjusted taking the bias into account.
3. The laboratory decay length corresponding to the selected mean decay life is obtained by multiplication by $\beta\gamma c$
4. Decay direction biasing is activated by a LAM–BIAS card with SDUM = DCDRBIAS. The direction of decay secondaries is sampled preferentially close to the direction specified by the user by means of a second LAM–BIAS card with SDUM = DCY–DIRE.

5. The biasing function for the decay direction is of the form $e^{-\frac{1-cos(\theta)}{\lambda}}$ where θ is the polar angle between the sampled direction and the preferential direction (transformed to the centre of mass reference system). The degree of biasing is largest for small positive values of $\boldsymbol{\lambda}$ (producing direction biasing strongly peaked along the direction of interest) and decreases with increasing $\boldsymbol{\lambda}$. Values of $\boldsymbol{\lambda} \geq$ `1.0` result essentially in no biasing.
6. Biasing of hadronic inelastic interaction length can be done either in one single material (indicated by WHAT(3)) or in all materials (default). No other possibility is foreseen for the moment.
7. When choosing the Russian Roulette alternative, it is suggested to set also a weight window (options WW–FACTOR and WW–THRESh, p. 244, 249) in order to avoid too large weight fluctuations.
8. Reduction factors excessively small can result in an abnormal increase of the number of secondaries to be loaded on the stack, especially at high primary energies. In such cases, FLUKA issues a message that the secondary could not be loaded because of a lack of space. The weight adjustment is modified accordingly (therefore the results are not affected) but if the number of messages exceeds a certain limit, the run is terminated.
9. Biasing of the hadronic inelastic interaction length can be applied also to photons (provided option PHOTONUC is also requested, p. 174) and muons (provided option MUPHOTON is also requested, p. 156); actually, it is often a good idea to do this in order to increase the probability of photonuclear interaction.
10. For photons, a typical reduction factor of the hadronic inelastic interaction length is the order of `0.01` to `0.05` for a shower initiated by 1 GeV photons or electrons, and of `0.1` to `0.5` for one at 10 TeV.

Example:

```
*...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+...
LAM-BIAS      -3.E+3        1.        1.       13.       16.        0.GDECAY
*  The mean decay length of pions and kaons (particles 13, 14, 15 and 16)
*  is set equal to 30 m. Survival of the decaying particle is decided by
*  Russian Roulette.
LAM-BIAS         0.0      0.02       11.        7.        0.        0.INEPRI
*  The interaction length for nuclear inelastic interactions of primary
*  photons (particle 7) is reduced by a factor 50 in material 11.
*  (Note that such a large reduction factor is often necessary for photons,
*  but is not recommended for hadrons). The photon survives after the
*  nuclear interaction with a reduced weight.
```

7.38 LOW–BIAS

Requests non-analogue absorption and/or an energy cut-off during low-energy neutron transport on a region by region basis

See also PART–THR, LOW–DOWN, LOW–NEUT

WHAT(1) > 0.0: group cut-off (neutrons in energy groups with number ≥ WHAT(1) are not transported).
This value can be overridden in user routine UBSSET (p. 320) by assigning a value to variable IGCUTO
Default = 0.0 (no cut-off)

WHAT(2) > 0.0: group limit for non-analogue absorption (neutrons in energy groups ≥ WHAT(2) undergo non-analogue absorption).
Non-analogue absorption is applied to the NMGP-WHAT(2)+1 groups with energies equal or lower than those of group WHAT(2) if WHAT(2) is not > NMGP, otherwise it isn't applied to any group (NMGP is the number of neutron groups in the cross-section library used: it is = 72 in the standard FLUKA neutron library).
This value can be overridden in user routine UBSSET (p. 320) by assigning a value to variable IGNONA.
Default : if option DEFAULTS is used with SDUM = CALORIMEtry, ICARUS, NEUTRONS or PRECISIOn, the default is = NMGP+1 (usually 73), meaning that non-analogue absorption is not applied at all.
If DEFAULTS is missing, or is present with any other SDUM value, the default is NMGP (usually 72), i.e., the number of the last group (usually a thermal group).

WHAT(3) > 0.0: non-analogue *survival* probability. Must be ≤ 1.0
This value can be overridden in user routine UBSSET (p. 320) by assigning a value to variable PNONAN.
Default : if option DEFAULTS is used with SDUM = EET/TRANsmut, HADROTHErapy, NEW–DEFAults or SHIELDINg, the default is 0.95.
If DEFAULTS is missing, or is present with any other SDUM value, the default is 0.85.

WHAT(4) = lower bound of the region indices in which the indicated neutron cut-off and/or survival parameters apply
("*From region WHAT(4)...*")
Default = 2.0

WHAT(5) = upper bound of the region indices in which the indicated neutron cut-off and/or survival parameters apply
("*...to region WHAT(5)...*")
Default = WHAT(4)

WHAT(6) = step length in assigning indices.
("*...in steps of WHAT(6)*")
Default = 1.0

SDUM : not used

Default (option LOW–BIAS not given): the physical survival probability is used for all groups except thermal ones, which are assigned a probability of 0.85. However, if option DEFAULTS has been chosen with SDUM = EET/TRANsmut, HADROTHErapy, NEW–DEFAults or SHIELDINg, this default value is changed to 0.95.
If SDUM = CALORIMEtry, ICARUS, NEUTRONS or PRECISIOn, the default is equal to the physical survival probability for all groups, including thermal.
See also Table 7.1 at p. 88.

Notes

1. The groups are numbered in *decreasing* energy order (see Chap. 10 for a detailed description). Setting a group cut-off larger than the last group number (e.g., 73 when using a 72-group cross-section set) results in all neutrons been transported, i.e., no cut-off is applied.
2. Similarly, if WHAT(2) is set larger than the last group number, non-analogue neutron absorption isn't applied to any group (this is recommended for calorimetry studies and all cases where fluctuations and correlations are important).
3. The survival probability is defined as $1-\frac{\Sigma_{abs}}{\Sigma_T}$ where Σ_{abs} is the inverse of the absorption mean free path and Σ_T the inverse of the mean free path for absorption plus scattering (total macroscopic cross-section). The LOW–BIAS option allows the user to control neutron transport by imposing an artificial survival probability and corrects the particle weight taking into account the ratio between physical and biased survival probability.
4. In some programs (e.g., MORSE) [44] the survival probability is always forced to be = 1.0. In FLUKA, if the LOW–BIAS option is not chosen, the physical survival probability is used for all non-thermal groups, and the default 0.85 is used for the thermal group. (The reason for this exception is to avoid endless thermal neutron scattering in materials having a low thermal neutron absorption cross-section). To get the physical survival probability applied to *all* groups, as needed for fully analogue calculations, the user must use LOW–BIAS with WHAT(2) larger than the last group number.
5. In selecting a forced survival probability for the thermal neutron group, the user should have an idea of the order of magnitude of the actual physical probability. The latter can take very different values: for instance it can range between a few per cent for thermal neutrons in ^{10}B to about 80-90 % in Lead and 99 % in Carbon. Often, small values of survival probability will be chosen for the thermal group in order to limit the length of histories, but not if thermal neutron effects are of particular interest.
6. Concerning the other energy groups, if there is interest in low-energy neutron effects, the survival probability for energy groups above thermal in non-hydrogenated materials should be set at least = 0.9, otherwise practically no neutron would survive enough collisions to be slowed down. In hydrogenated materials, a slightly lower value could be acceptable. Setting less than 80 % is likely to lead to erroneous results in most cases.
7. Use of a survival probability equal or larger than the physical one is likely to introduce important weight fluctuations among different individual particles depending on the number of collisions undergone. To limit the size of such fluctuations, which could slow down statistical convergence, it is recommended to define a weight window by means of options WW–THRESh (p. 249) and WW–FACTOr (p. 244).

Example:

```
*...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+...
LOW-BIAS        60.0      47.0      0.95       5.0      19.0       0.0
LOW-BIAS        73.0      72.0      0.82       7.0      15.0       4.0
*  Note that the second LOW-BIAS card overrides the settings of the first one
*  concerning regions 7, 11 and 15. Therefore, we will have an energy cut-off
*  equal to the upper edge of the 60th group (61.442 eV in the standard FLUKA
*  neutron library) in regions 5,6,8,9,10,12,13,14,16,17,18 and 19. In these
*  same regions, analogue neutron absorption is requested down to an energy
*  equal to the upper edge of group 47 (15.034 keV in the standard library),
*  and biased absorption, with a fixed probability of 95\%, at lower energies.
*  In regions 7, 11 and 15, no cut-off is applied (supposing we are using the
*  standard 72-group library), and non-analogue absorption is requested only
*  for group 72 (the thermal group in our case), with a probability of 82\%.
```

7.39 LOW–DOWN

Biases the downscattering probability during low-energy neutron transport on a region by region basis

See also LOW-BIAS, LOW–NEUT

WHAT(1) > 0.0: group limit for biased downscattering probability.
Neutrons in energy groups IG ≥ WHAT(1) are given downscattering importances = $\texttt{WHAT(2)}^{\texttt{IG}-\texttt{WHAT(1)}}$.
This value can be overridden in user routine UBSSET (p. 320) by assigning a value to variable IGDWSC.
Default = 0.0 (no biased downscattering)

WHAT(2) = biasing factor for down-scattering from group IG-1 into group IG.
This value can be overridden in user routine UBSSET (p. 320) by assigning a value to variable FDOWSC.
Default = 1.5

WHAT(3) : not used

WHAT(4) = lower bound of the region indices in which downscattering biasing is to be applied
("*From region WHAT(4)...*")
Default = 2.0

WHAT(5) = upper bound of the region indices in which downscattering biasing is to be applied
("*...to region WHAT(5)...*")
Default = WHAT(4)

WHAT(6) = step length in assigning indices.
("*...in steps of WHAT(6)*")
Default = 1.0

SDUM : not used

Default (option LOW–DOWN not given): no downscatter biasing

Notes

1. This option can be useful only in very particular problems, for instance to calculate the response of instruments based on moderation (Bonner spheres, rem-counters). Very powerful but also very dangerous, it can lead to *errors of orders of magnitude* if not used by experts.

   !!!! Use with the maximum care !!!!
   !!!! especially with hydrogenated materials !!!!

2. The groups are numbered in *decreasing* energy order (see Chap. 10 for a detailed description).
3. When this option is used, the natural probabilities of scatter from group I to group J, P(I→J), are altered by an importance factor V(J). Selection of the outgoing group J is made from a biased distribution function P(I→J)·V(J) with an associated weight correction.

Example:

```
*...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+...
LOW-DOWN        38.0       1.1       0.0       4.0      25.0       0.0
*  In all regions from 4 to 25, neutron downscattering is biased from
*  group 38 to last. Assuming we are using the standard FLUKA library with
*  72 groups, that means all energies below 273.24 keV. In group 38, the
*  downscattering relative importance is set equal to 1.1, in group 39 to
*  1.1**2 = 1.21, in group 40 to 1.1**3 = 1.331 etc.
```

7.40 LOW–MAT

Sets the correspondence between FLUKA *materials and low-energy neutron cross-sections*

See also LOW–NEUT

WHAT(1) = number of the FLUKA material, either taken from the list of standard FLUKA materials (see Table 5.3, p. 44, or defined via a MATERIAL option (p. 142).
Default : No default!

WHAT(2) = first numerical identifier of the corresponding low-energy neutron material.
Not used if = 0.0

WHAT(3) = second numerical identifier of the corresponding low-energy neutron material.
Not used if = 0.0

WHAT(4) = third numerical identifier of the corresponding low-energy neutron material.
Not used if = 0.0

WHAT(5) = **(Not implemented!)** compound material if > 0.0. This applies only to pre-mixed low-energy neutron compound materials, which could possibly be available in the future; at the moment however, none is yet available. (It would be allowed anyway only if the corresponding FLUKA material is also a compound).
Default : compound if the FLUKA material is a compound, otherwise not.

WHAT(6) = **(Not implemented!)** atomic or molecular density (in atoms/(10^{-24}cm^3), or number of atoms contained in a 1-cm long cylinder with base area = 1 barn.
To be used *only* if referring to a pre-mixed compound data set (see COMPOUND, Note 7, p. 78 and explanation of WHAT(5) here above).
Note that no such data set has been made available yet.

SDUM = name of the low-energy neutron material.
Default : same name as the FLUKA material.

Default (option LOW–MAT not given): correspondence between FLUKA and low-energy neutron materials is by *name*; in case of ambiguity the first material in the relevant list (see Table 10.3, p. 295) is chosen.

Notes

1. Each material in the FLUKA low-energy neutron library (see Chap. 10) is identified by an alphanumeric name (a string of ≤ 8 characters, all in upper case), and by three integer numbers. Correspondence with FLUKA materials (standard or user-defined) is based on any combination of name and zero or more identifiers. In case of ambiguity, the first material in the list fulfilling the combination is selected.
2. Option LOW–MAT should be avoided if it is not really necessary (experience has shown that it is often misinterpreted by beginner users). The option is **not required** if the following 3 conditions are all true:
   i) the low-energy neutron material desired is unique or is listed before any other material with the same name in Table 10.3
   and
   ii) that name is the same as one in the FLUKA list (Table 10.3) or as given by a MATERIAL option
   and
   iii) there is only one FLUKA material associated with that low-energy neutron material

3. On the other hand, the option **is required** in any one of the following cases:
   i) there is more than one low-energy neutron material with that name (this can happen because of data sets coming from different sources, or corresponding to different neutron temperatures, or concerning different isotopes, or weighted on different spectra, etc), and the one desired is not coming first in the list. In this case it is sufficient to provide just as many identifiers as required to remove ambiguity
   or
   ii) The FLUKA name is different from the name of the low-energy neutron material
   or
   iii) There is more than one FLUKA material associated with the given low-energy neutron material. This can happen for instance when the same material is present with different densities in different regions. In reality this is a special case of ii) above, since to distinguish between the different densities, different names must be used and one at least will not be equal to the name of the low-energy neutron material.
4. **(Not implemented!)** If WHAT(5) is set $>$ 0.0 because a pre-mixed compound low-energy neutron material is used, average cross-sections are used (as for instance in the MORSE code). Otherwise, if each of the FLUKA elemental components has been associated with one of the elemental low-energy neutron components and the composition of the compound has been defined by a COMPOUND option, low-energy neutron interactions will take place randomly with each individual component, with the appropriate probability.
   It is however possible to have in the same run detailed individual interactions at high energies and average compound interactions for low-energy neutrons. But *not the other way around!*
5. See Chap. 15 for a complex example showing the use of MATERIAL, COMPOUND and LOW–MAT.

Example 1:

```
*...+....1....+....2....+....3....+....4....+....5....+....6....+....7...+...8
MATERIAL          6.    12.011     2.000        6.       0.0       0. CARBON
MATERIAL          6.    12.011     1.700       15.       0.0       0. GRAPHITE
MATERIAL          6.    12.011     3.520       16.       0.0       0. DIAMOND
LOW-MAT         15.0        6.       -2.      293.       0.0       0. CARBON
LOW-MAT         16.0        6.       -2.      293.       0.0       0. CARBON
*  We have three materials with the same atomic composition, but different
*  density (amorphous carbon, graphite and diamond). Both graphite and
*  diamond are declared as having the same low-energy neutron cross-section
*  as carbon.
```

Example 2:

```
*...+....1....+....2....+....3....+....4....+....5....+....6....+....7...+...8
MATERIAL          5.    10.811      2.34        5.       0.0       0. BORON
MATERIAL         26.    55.847      7.87       11.       0.0       0. IRON
MATERIAL         60.    144.24      7.01       13.       0.0       0. NEODYMIU
MATERIAL          0.        0.      7.40       20.       0.0       0. NDFEB
COMPOUND          2.       13.       14.       11.        1.       5. NDFEB
LOW-MAT         13.0       26.       -2.      293.       0.0       0. IRON
*  A permanent magnet is made of an alloy of iron, neodymium and boron.
*  Nd cross-sections are not available yet.  Let us assume for the time being
*  that it has the same low energy-neutron cross-section as iron.
```

7.41 LOW–NEUT

Activates low-energy neutron transport

See also LOW–BIAS, LOW–MAT

WHAT(1) = number of neutron groups in the neutron cross-section library used. The ENEA standard neutron library has 72 groups (see Chap. 10).
Default = 72.0

WHAT(2) = number of gamma groups.
Default : No default if WHAT(1) is given, 22.0 otherwise. (The ENEA neutron library has 22 gamma groups).

WHAT(3) = maximum energy of the low-energy cross-section neutron library. For the ENEA neutron library, the maximum energy is 0.0196 GeV. Note that rounding (for instance 0.020 GeV instead of 0.0196) is not allowed!
Default = 0.0196

WHAT(4) = printing flag (see p. 274 for a detailed description of the printed output):
from 0.0 to 3.0 increases the amount of output about cross-sections, kerma factors, etc.:
1.0: Standard output includes integral cross-sections, kerma factors and probabilities
2.0: In addition to the above, downscattering matrices and group neutron-to-gamma transfer probabilities are printed
3.0: In addition to the above, scattering probabilities and angles are printed
Default : 0.0 (minimum output)

WHAT(5) = number of neutron groups to be considered as thermal ones. (The ENEA neutron library has one thermal group).
= 0.0: ignored
> 0.0: resets to the default = 1.0
Default = 1.0

WHAT(6) = $i_0 + 10 \cdot i_1$:
i_0 = 1: available pointwise cross-sections are used (see Note 4 below) and explicit and correlated ^{6}Li(n,γ)^{7}Li, ^{10}B(n,tγ)^{4}He, ^{40}Ar(n,γ)^{41}Ar, ^{x}Xe(n,γ)$^{x+1}$Xe and ^{113}Cd(n,γ)^{114}Cd photon cascades are requested
= 0: ignored
$\leq$ -1: resets to the default (pointwise cross-sections are not used)
i_1 = 1: fission neutron multiplicity is forced to 1, with the proper weight
= 0: ignored
$\leq$ -1: resets to the default (normal fission multiplicity)
Default = -11., unless option DEFAULTS has been chosen with SDUM = CALORIMEtry, ICARUS, NEUTRONS or PRECISIOn, in which case the default is 1.0, meaning that pointwise cross-sections are used when available and fission multiplicity is not forced)

SDUM : not used

Default (option LOW–NEUT not given): if option DEFAULTS has been chosen with SDUM = CALORIMEtry, EET/TRANsmut, HADROTHErapy, ICARUS, NEUTRONS, NEW–DEFAults, PRECISIOn or SHIELDINg, low-energy neutrons are transported and a suitable cross-section library must be available.
In all other cases, low-energy neutrons are not transported, and their energy is deposited as explained in Note 2 below.

Notes

1. In FLUKA, transport of neutrons with energies lower than a certain threshold is performed by a multigroup algorithm. For the neutron cross-section library currently used by FLUKA, this threshold is 0.0196 GeV. The multi-group transport algorithm is described in Chap. 10.
2. Evaporation option (see EVENTYPE, p. 116) is mandatory if LOW–NEUT is requested. If low-energy neutrons are not transported (because of the chosen DEFAULTS, or because a DEFAULTS card is absent), the energy of neutrons below threshold (default or set by PART–THR, p. 172) is deposited on the spot.
   This is true also for evaporation neutrons. If there is no interest in transporting low-energy neutrons, but this feature is not desired, it is suggested to request LOW–NEUT, and to use LOW–BIAS (p. 135) with a group cut-off WHAT(1) = `1.0`.
3. Gamma data are used only for capture gamma generation and not for transport (transport is done via the **E**lectro**M**agnetic **F**LUKA module EMF using continuous cross-sections).
   The actual precise energy of a photon generated by (n,γ) reactions is sampled randomly within the gamma energy group concerned, except for a few important reactions where a single monoenergetic photon is emitted. By default, for the ^{1}H(n,γ)^{2}H reaction the actual photon energy of 2.226 MeV is used. It is possible to do the same with the capture gammas in ^{6}Li, ^{10}B, ^{40}Ar, ^{x}Xe and ^{113}Cd, by setting WHAT(6) = `1.0` or `11.0`.
4. Pointwise neutron transport is available, by setting WHAT(6) = `1.0` or `11.0`, for the following nuclides: ^{1}H (above 10 keV), ^{6}Li (all reactions), ^{10}B (only for the reaction ^{10}B(n,tγ)^{4}He). Recoil protons are always transported explicitly, and so is the proton from the ^{14}N(n,p) reaction, for which a pointwise treatment is always applied.
5. The groups are numbered in *decreasing* energy order (see Chap. 10 for a detailed description). Note that the energy limits of the thermal neutron group in the ENEA neutron library (see Sec. 10.4) are: 10^{-14} GeV (10^{-5} eV) and 4.14×10^{-10} GeV (0.414 eV)
6. Here are the settings for transport of low-energy neutrons corresponding to available DEFAULTS options:

   CALORIMEtry, ICARUS, NEUTRONS, PRECISIOn: low-energy neutrons are transported, using pointwise cross-section when available

   EET/TRANsmut, HADROTHErapy, NEW–DEFAults (or DEFAULTS missing), SHIELDINg: low-energy neutrons are transported using always multigroup cross-sections

   Any other SDUM value: no low-energy neutron transport

Example:

```
*...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+...
LOW-NEUT        72.0      22.0    0.0196       2.0       1.0      11.0
*  The low-energy neutron library used is the (72n, 22gamma) multigroup  ENEA
*  library. The user requests a printout of cross-sections, kerma factors,
*  probabilities, downscattering matrices and n-->gamma transfer probabilities.
*  Pointwise cross-sections will be used where available, and only one
*  neutron per low-energy fission will be emitted, with an adjusted weight.
```

7.42 MATERIAL

Defines a single-element material or (coupled to a COMPOUND card) a compound

See also COMPOUND, LOW–MAT, MAT–PROP

WHAT(1) = atomic number (meaningful only when *not coupled* to a COMPOUND card; otherwise WHAT(1) must be = 0.0)
No default

WHAT(2) = atomic weight (meaningful only when *not coupled* to a COMPOUND card; otherwise WHAT(2) must be = 0.0)
No default

WHAT(3) = density in g/cm^3. Note that if the density is lower than 0.01, the material is considered to be a gas at atmospheric pressure unless set otherwise by MAT–PROP (p. 144)
No default

WHAT(4) = number (index) of the material
Default = NMAT+1 (NMAT is the current number of defined materials. Its value is = 25 before any MATERIAL card is given, and doesn't change if WHAT(4) overrides a number which has already been assigned)

WHAT(5) ≥ 2.0: alternate material number for ionization processes (this material will be used instead of WHAT(1) for dE/dx etc.)
> 0.0 and ≤ 2.0: ignored

≤ 0.0: reset to default

Default : no alternate material

WHAT(6) = mass number of the material: set = 0.0 unless a specific individual isotope is desired. If not zero a nucleus of the given mass number is used by the EVAP generator for inelastic collisions, otherwise the natural isotopic composition of the WHAT(1) element is used. For isotopic composition other than natural or single isotope, see COMPOUND.

SDUM = name of the material
Default : COPPER (FLUKA material 12.0)

Default (option MATERIAL not given): standard pre-defined material numbers are used (see list in Table 5.3, p. 44).

Notes

1. MATERIAL cards can be used in couple with COMPOUND cards in order to define compounds, mixtures or isotopic compositions. See COMPOUND for input instructions (p. 78).
2. Material number 1.0 is always Black Hole (called also External Vacuum) and it cannot be redefined. (All particles vanish when they reach the Black Hole, which has an infinite absorption cross-section)
3. Material number 2.0 is always Vacuum (of zero absorption cross-section) and it cannot be redefined.
4. Although the material number can be omitted, it is not recommended to do so. It is allowed to override a number already assigned (either by default, see list in Table 5.3, or by a previous MATERIAL card).
5. If the number has not been assigned before, it must be the next number available (26.0, 27.0... for successive MATERIAL cards). It is not allowed to leave empty gaps in the number sequence.
6. Materials having a different density at the macroscopic and at the microscopic levels (e.g., spongy matter or approximations for not entirely empty vacuum) need a special treatment regarding stopping power (density effect). In such cases, see MAT–PROP, p. 144.
7. See Chap. 15 for an example of use of MATERIAL, COMPOUND and LOW–MAT.

Example:

```
*...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+...
MATERIAL          1.    1.0079  8.988E-5        3.       0.0       1. HYDROGEN
MATERIAL          6.    12.011     2.265       10.       0.0       0. CARBON
MATERIAL          6.    12.011       2.0       11.       0.0       0. GRAPHITE
MATERIAL         41.   92.9064      8.57       15.       0.0       0. NIOBIUM
MATERIAL         48.   112.411     8.650       26.       0.0       0. CADMIUM
MATERIAL         24.    51.996      7.19       27.       0.0       0. CHROMIUM
MATERIAL         27.  58.93320      8.90       28.       0.0       0. COBALT
* Several cases are illustrated:
* Hydrogen, pre-defined as material 3, is re-defined with the same number, but
* as monoisotopic 1-H. Carbon, originally pre-defined as material 6.0, is
* re-defined twice: first with a different density as 10.0, overriding the
* default 10.0 (Aluminium), and then with a different name, overriding the
* default 11.0 (Iron). Niobium is defined as material 15.0, overriding the
* default pre-defined material 15.0 (Gold). Cadmium, Chromium and Cobalt are
* added to the list, and are assigned consecutive numbers starting from the
* first free slot (26.0)
```

7.43 MAT–PROP

Provides extra information about materials

See also MATERIAL, STERNHEIme

This command can be used for several different tasks:

1. to supply extra information about gaseous materials and materials with fictitious or effective density
2. to override the default average ionisation potential
3. to do a rough rescaling of thermal neutron cross-sections if the actual material temperature is different from that of the low-energy neutron cross-section data set
4. to set a flag to call the user routine `USRMED` (p. 323) every time a particle is going to be transported in selected material(s)

For SDUM ≠ LOWNTEMP, USERDIREctive:

WHAT(1) > `0.0`: gas pressure in atmospheres (see Note 1 below).
= `0.0`: ignored
< `0.0`: resets to 1 atm a possible previously input pressure value
Default = `1.0`

WHAT(2) = `RHOR` factor: this factor multiplies the density of a material when calculating the density effect parameters (e.g., if a reduced density is used to simulate voids, but of course the density effect parameters must be computed with the actual local physical density at the microscopic level). See Note 3 below.
= `0.0`: ignored
< `0.0`: a possible previously input value is restored to default = `1.0`
Default = `1.0`

WHAT(3) > `0.0`: average ionisation potential to be used for dE/dx calculations (eV)
< `0.0`: a default value of the average ionisation potential is obtained from the systematics of Ziegler [178] or Sternheimer, Berger and Seltzer [163, 164]
= `0.0`: ignored
Default : ionisation potential calculated from systematics

WHAT(4) = lower bound of the indices of materials in which gas pressure, `RHOR` factor or ionisation potential are set
("*From material WHAT(4)...*")
Default = `3.0`

WHAT(5) = upper bound of the indices of materials in which gas pressure, `RHOR` factor or ionisation potential are set
("*... to material WHAT(5)...*")
Default = WHAT(4)

WHAT(6) = step length in assigning indices
("*... in steps of WHAT(6)*")
Default = `1.0`

Default : (option MAT–PROP not given): if the density of the default material or that assigned by a MATERIAL card is > `0.01`, the material is not assumed to be a gas. Otherwise it is a gas at a default pressure of 1 atmosphere. If the material is a compound, the average ionisation potential is that resulting from applying Bragg's rule of additivity to stopping power.

For SDUM = LOWNTEMP:

WHAT(1) $>$ 0.0: Temperature ratio (T_{actual}/T_{xsec}) with respect to the nominal one. The nominal temperature T_{xsec} (given by WHAT(3), see below) is the temperature for which the neutron cross-sections of the FLUKA material(s) concerned have been prepared (see Table 10.3, p. 295). See Notes 4, 5, 6 for more details and limitations.
= 0.0: ignored
$<$ 0.0: a possible previously given value is restored to default = 1.0
Default = 1.0

WHAT(2) $>$ 0.0: Number of (thermal) groups to which the temperature ratio has to be applied. It must be $\leq N_{th}$, where N_{th} is the number of thermal groups in the cross-section library. If WHAT(2) $<$ N_{th}, the last WHAT(2) groups are affected (see the LOW–NEUT option for setting the number of thermal groups).
= 0.0: ignored
$<$ 0.0: a possible previously given value is restored to default = 0.0
Default = 0.0 (i.e., no correction, the whole command is ignored).

WHAT(3) $>$ 0.0: Nominal temperature (K) of the indicated FLUKA materials
= 0.0: ignored
$<$ 0.0: a possible previously given value is restored to default (= 3rd identifier, see below)
Default = the temperature given as 3rd identifier of the associated low-energy neutron data set (see LOW–MAT and the cross-section description in Chap. 10)

WHAT(4) = lower bound of the indices of materials in which the temperature rescaling must be applied ("*From material WHAT(4)...*")
Default = 3.0

WHAT(5) = upper bound of the indices of materials in which the temperature rescaling must be applied ("*... to material WHAT(5)...*")
Default = WHAT(4)

WHAT(6) = step length in assigning indices ("*... in steps of WHAT(6)*")
Default = 1.0

For SDUM = USERDIREctive:

WHAT(1) = 0.0: ignored
$>$ 0.0: a call to the user routine USRMED (p. 323) will be performed at run time every time a particle is going to be transported in the selected materials (spot depositions *are* anyway performed).
$<$ 0.0: a possible previously given value is restored to default (i.e., no call to USRMED is made)

WHAT(2) and **WHAT(3)**: not used

WHAT(4) = lower bound of the indices of materials in which the call to USRMED must be performed ("*From material WHAT(4)...*")
Default = 3.0

WHAT(5) = upper bound of the indices of materials in which the call to USRMED must be performed ("*... to material WHAT(5)...*")
Default = WHAT(4)

WHAT(6) = step length in assigning indices ("*... in steps of WHAT(6)*")
Default = 1.0

Default (option MAT–PROP not given): no extra information about the assigned materials is supplied

Notes

SDUM = blank (i.e., ≠ LOWNTEMP, USERDIREctive):

1. When issuing a MATERIAL definition the gas pressure is set to 1 atm if the density RHO is < 0.01. If this value is not acceptable to the user, a MAT–PROP card must be issued *after* the MATERIAL card to force a different value of the gas pressure. Note that this is one of the rare cases (with GLOBAL, DEFAULTS and PLOTGEOM) where sequential order of input cards is of importance in FLUKA.
   A non-zero value of WHAT(1) must be given only for gases: it is important when calculating the density effect parameters of the stopping power (see Note 1 to option STERNHEIme, p. 202, and Note 2 here below) .
2. If WHAT(1) is set to a value > 0.0, the transport of charged particles will be calculated according to a density RHO defined at the actual pressure by the corresponding MATERIAL card, while the density effect correction to stopping power will be calculated using a density ρ(NTP) = RHO/WHAT(1) and then re-scaled to the actual density RHO.
3. When giving a WHAT(2) non-zero value, remember that if RHO indicates the "transport (effective) density", the "physical density" used to calculate the density effect on stopping power will be RHOR*RHO = WHAT(2)*RHO.

SDUM = LOWNTEMP:

4. The temperature ratio is used to rescale the thermal group velocities, absorption probabilities, gamma generation probabilities, fission probabilities and kermas. For absorption, fission and gamma generation a 1/v dependence of the corresponding cross-sections is implicitly assumed: **if this is not the case the whole scheme is meaningless!**.
   No modification is made to the elastic cross-section and hence to the downscattering matrix: **this can be a very bad approximation, especially for light materials**. No modification is applied for possible Doppler broadening effects on resonances for thermal and epithermal neutrons: again, this can be a bad approximation. The total cross-section is rescaled according to the modified absorption and fission cross-sections.
5. cross-section rescaling is applied to the FLUKA materials at run time. That is, if for example two ^{10}B materials are defined and both point to the same cross-section data set, a possible temperature rescaling will affect only the FLUKA material indicated by MAT–PROP, while the other one will be unaffected, although they share the same low-energy neutron cross-section data set.
6. Velocity setting is applied for compounds *independently* from cross-section rescaling. That is, a (nominal) temperature input for a compound is fully meaningful and will be used for velocity computation. However cross-section rescaling is applied on single constituents (of course!) and therefore...
   !!!!! IMPORTANT WARNING !!!!
   ...it cannot be used for compounds unless the corresponding neutron cross-section data set is a pre-mixed one (see Notes 4, p. 139 and 7, p. 78) *(option presently not available)*. Otherwise a new compound must be created with new elemental constituents and the correction must be invoked for each constituent.

SDUM = USERDIREctive:

7. User routine USRMED is typically used to implement albedo and refraction, especially in connection with optical photon transport as defined by OPT–PROP (p. 164). See 12.2.27 for instructions.

Example 1:

```
* Call USRMED every time a particle is going to be transported in Pb Glass or
* in plexiglas (PMMA)
*...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+...
MATERIAL          1.   1.00794 8.3748E-5        7.       0.0       1. HYDROGEN
MATERIAL          6.    12.011     2.265        8.       0.0       0. CARBON
MATERIAL          8.   15.9994  0.001429        9.       0.0       0. OXYGEN
MATERIAL         14.   28.0855      2.33       10.       0.0       0. SILICON
MATERIAL         22.     47.88      4.54       11.       0.0       0. TITANIUM
MATERIAL         33.   74.9216      5.73       12.       0.0       0. ARSENIC
MATERIAL         82.    207.2     11.35       13.       0.0       0. LEAD
* Compositions as in Seltzer & Berger, Int. J. Appl. Rad. Isot. 33, 1189 (1982)
MATERIAL          0.        0.      6.22       18.       0.0       0. LEADGLAS
COMPOUND   -0.156453        9. -0.080866       10. -0.008092      11. LEADGLAS
COMPOUND   -0.002651       12. -0.751938       13.       0.0       0. LEADGLAS
```

```
*
MATERIAL          0.        0.      1.19       15.       0.0       0. PMMA
COMPOUND   -0.080538        7. -0.599848        8. -0.319614       9. PMMA
*
MAT-PROP         1.0       0.0       0.0       15.       18.       3. USERDIRE
```

Example 2:

```
* Lung tissue with ICRP composition and Sternheimer parameters
*...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+...
MATERIAL          1.   1.00794 8.3748E-5        3.       0.0       1. HYDROGEN
MATERIAL          6.    12.011     2.265        6.       0.0       0. CARBON
MATERIAL          7.  14.00674 0.0011653        7.       0.0       0. NITROGEN
MATERIAL          8.   15.9994  0.001429        8.       0.0       0. OXYGEN
MATERIAL         11.  22.98977     0.971        9.       0.0       0. SODIUM
MATERIAL         12.    24.305      1.74       10.       0.0       0. MAGNESIU
MATERIAL         15.  30.97376       2.2       11        0.0       0. PHOSPHO
MATERIAL         16.    32.066       2.0       12.       0.0       0. SULFUR
MATERIAL         17.   35.4527 2.9947E-3       13        0.0       0. CHLORINE
MATERIAL         19.   39.0983     0.862       14.       0.0       0. POTASSIU
MATERIAL         20.    40.078      1.55       15.       0.0       0. CALCIUM
MATERIAL         26.    55.847     7.874       16.       0.0       0. IRON
MATERIAL         30.     65.39     7.133       17.       0.0       0. ZINC
* Local density of lung is 1.05 g/cm3
MATERIAL          0.0       0.0      1.05       18.       0.0       0. LUNG
COMPOUND   -0.101278        3.  -0.10231        6.  -0.02865       7. LUNG
COMPOUND   -0.757072        8.  -0.00184        9.  -0.00073      10. LUNG
COMPOUND     -0.0008       11.  -0.00225       12.  -0.00266      13. LUNG
COMPOUND    -0.00194       14.  -0.00009       15.  -0.00037      16. LUNG
COMPOUND    -0.00001       30.        0.        0.        0.       0. LUNG
* Average density of lung is 1.05*0.286 = 0.3 g/cm3. Average ionisation
* potential is 75.3 eV (At. Data Nucl. Data Tab. 30, 261 (1984))
MAT-PROP          0.0     0.286      75.3       18.        0.       0.
STERNHEI      3.4708    0.2261    2.8001   0.08588    3.5353       0.  18
```

Example 3:

```
* Definition of air at non-standard pressure.
*...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+...
MATERIAL          6.    12.011     2.265        6.       0.0       0. CARBON
MATERIAL          7.  14.00674 0.0011653        7.       0.0       0. NITROGEN
MATERIAL          8.   15.9994  0.001429        8.       0.0       0. OXYGEN
MATERIAL         18.    39.948  1.662E-3        9.       0.0       0. ARGON
* AIR defined as air with normal NTP density (0.001205)
MATERIAL          0.0       0.0  0.001205       10.       0.0       0. AIR
COMPOUND   -0.000124        6. -0.755267        7. -0.231781       8. AIR
COMPOUND   -0.012827        9.                                       AIR
* AIR2 defined as air with a density 0.002410, double of that at NTP
MATERIAL          0.0       0.0  0.002410       11.       0.0       0. AIR2
COMPOUND   -0.000124        6. -0.755267        7. -0.231781       8. AIR2
COMPOUND   -0.012827        9.                                       AIR2
* The pressure of AIR2 is 2 atm. Set also the ionisation potential = 85.7 eV
MAT-PROP          2.0       0.0      85.7       10.
STERNHEI     10.5961    1.7418    4.2759   0.10914    3.3994       0.  11
```

Example 4:

```
* The total and capture cross-sections of Au have a good 1/v dependence in the
* thermal region. Here we assume Gold to be at a temperature of 300K, while
* the cross-sections in the FLUKA library are at 293K.
*...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+...
```

```
MATERIAL        79.0  196.9665     19.32       15.       0.0       0.
* Next card is not strictly necessary, since it doesn't change any default
LOW-MAT         15.0      79.0     197.0     293.0       0.0       0. GOLD
* (300/293 = 1.02389). The present library has only 1 thermal group.
MAT-PROP     1.02389       1.0      293.       15.       0.0       0. LOWNTEMP
```

7.44 MCSTHRES

Defines some of the accuracy requirements for Multiple Coulomb Scattering (MCS) of heavy charged particles (hadrons and muons).

See also MULSOPT

WHAT(1) $\geq$ `0.0`: detailed multiple Coulomb scattering for primary charged hadrons and muons down to the minimum energy allowed by Molière's theory

$<$ `0.0`: detailed multiple Coulomb scattering for primary charged hadrons and muons down to a kinetic energy equal to |WHAT(1)| (GeV)

Default = `1.0` if option DEFAULTS (p. 83) has been chosen with SDUM = CALORIMEtry, HADROTHErapy, ICARUS or PRECISIOn.
If SDUM = EET/TRANsmut, the default is = `-0.01` (transport of primaries with multiple Coulomb scattering down to 10 MeV).
With any other SDUM value, or if DEFAULTS is missing, the default is = `-0.02` (transport of secondaries with multiple Coulomb scattering down to 20 MeV).

WHAT(2) $\geq$ `0.0`: detailed multiple Coulomb scattering for secondary charged hadrons and muons down to the minimum energy allowed by Molière's theory

$<$ `0.0`: detailed multiple Coulomb scattering for secondary charged hadrons and muons down to a kinetic energy equal to |WHAT(2)| (GeV)

Default = `1.0` if DEFAULTS has been chosen with SDUM = CALORIMEtry, HADROTHErapy, ICARUS or PRECISIOn.
If SDUM = EET/TRANsmut, NEW–DEFAults or SHIELDINg, the default is = `-0.02` (transport of secondaries with multiple Coulomb scattering down to 20 MeV).
With any other SDUM value, or if DEFAULTS is missing, the default is = `-1.0` (transport of secondaries with multiple Coulomb scattering down to 1 GeV).

WHAT(3) – **WHAT(6), SDUM**: not used

Default : (option MCSTHRES not given): the defaults depend on option DEFAULTS as explained above and in Note 6. See also Table 7.1 on p. 88.

Notes

1. The MCSTHRES option is not used often, since option DEFAULTS ensures the MCS parameter setting most appropriate for a wide range of problems. In most cases, it is suggested to have multiple Coulomb scattering fully activated for both primary and secondary particles over the whole energy range. This corresponds to using WHAT(1) $\geq$ `0.0` and WHAT(2) $\geq$ 0.0 (or at least WHAT(2) $<$ `0.0` with an absolute value much smaller than beam energy).
2. WHAT(1) = `0.0` should generally be avoided. The reason is twofold:
   (a) tracking accuracy would be spoiled for no substantial gain in speed
   (b) FLUKA tracking without MCS does not take into account the variation of nuclear interaction cross-section with energy
3. However, there are some cases where it can be useful to set WHAT(1) and/or WHAT(2) to a negative number with absolute value *larger* than beam energy. In this case no MCS is performed but tracking and maximum energy loss per step are controlled anyway by the most sophisticated transport algorithm available (see FLUKAFIX, p. 119 and STEPSIZE, p. 200).
   Complete suppression of multiple scattering can be useful in some particular cases, for instance when replacing a gas of extremely low density by a gas of the same composition but of much larger density in order to increase the frequency of inelastic interactions (of course, the results must then be scaled by the density ratio). In such cases, one should also select the biased density so that no re-interaction of secondaries can take place. An alternative way to switch off completely multiple Coulomb scattering of hadrons and muons is to use MULSOPT (p. 153) with WHAT(2) $\geq$ `3.0` (MULSOPT, however, can deal also with electrons and positrons, while MCSTHRES can't; on the other hand, MULSOPT does not allow to distinguish between primary and secondary particles).

4. In order to get the most accurate treatment of Multiple Coulomb Scattering, a step optimisation and higher order corrections can be requested by option MULSOPT (but with an important increase in CPU time).
5. In pure electromagnetic or low-energy neutron problems, option MCSTHRES does not need to be given and has no effect.
6. Here are the MCS settings corresponding to available DEFAULTS options:

   CALORIMEtry, HADROTHErapy, ICARUS, PRECISIOn: Multiple scattering threshold at minimum allowed energy both for primary and secondary charged particles

   EET/TRANsmutation: MCS threshold = 10 MeV for primaries and 20 MeV for secondaries

   NEW–DEFAults (or DEFAULTS missing), SHIELDING: 20 MeV threshold for both primaries and secondaries

   Any other SDUM value: 20 MeV for primaries and 1 GeV for secondaries

Example (the comment lines shown are allowed input lines):

```
*...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+...
BEAM           120.0       0.0       0.0       0.0       0.0      1.0 PION+
MCSTHRES         1.0     -0.01       0.0       0.0       0.0      0.0
*         In this example, the primary beam consists of 120 GeV/c pi+
*         mesons which are transported by simulating accurately multiple
*         Coulomb scattering at all energies. For the secondary hadrons
*         generated, MCS is performed instead only until they reach 10 MeV.
```

7.45 MGNFIELD

Sets the tracking conditions for transport in magnetic fields and also may define an homogeneous magnetic field.

See also ASSIGNMA, STEPSIZE

WHAT(1) = largest angle in degrees that a charged particle is allowed to travel in a single step
Default = `57.0` (but a maximum of `30.0` is recommended!)

WHAT(2) = upper limit to error of the boundary iteration in cm (minimum accuracy accepted in determining a boundary intersection). It also sets the minimum radius of curvature for stepping according to WHAT(1)
Default = `0.05` cm.

WHAT(3) = minimum step length if the step is forced to be smaller because the angle is larger than WHAT(1).
Default = `0.1` cm.

WHAT(4) – **WHAT(6)**: = $\mathbf{B}_x$, $\mathbf{B}_y$, $\mathbf{B}_z$ components of magnetic field on the coordinate axes (in tesla).
Default ($\mathbf{B}_x = \mathbf{B}_y = \mathbf{B}_z =$ `0.0`): a user-supplied subroutine `MAGFLD` (p. 309) is assumed to provide the actual values (see Notes 2 and 3 below)

SDUM : not used

Default (option MGNFIELD not given): the defaults indicated for WHAT(1–6) apply if a magnetic field exists in the current region because of an ASSIGNMAt command (p. 62).

Notes

1. If $\mathbf{B}_x = \mathbf{B}_y = \mathbf{B}_z =$ `0.0`, the user-written subroutine `MAGFLD` is called at each step to get the direction cosines and the module (in tesla) of the magnetic field as a function of region or of coordinates. A sample subroutine is provided with the FLUKA code; instructions on how to write user-supplied routines can be found in Chap. 12.
2. Note that the argument list of subroutine `MAGFLD` is `(X,Y,Z,BTX,BTY,BTZ,B,NREG,IDISC)`, where `BTX, BTY, BTZ` are the *direction cosines* of the magnetic field at point `X,Y,Z` (*not the components* of the field! The field magnitude is given by `B`). For this reason, it is imperative that `MAGFLD` return normalised values of `BTX, BTY` and `BTZ` such that the sum of their squares be = `1.0` *in double precision*.
   Three zero values are not accepted: if the field is zero at the point concerned, you must return for instance `0.0, 0.0, 1.0` and `B = 0.0`.
   On the contrary, note that $\mathbf{B}_x$, $\mathbf{B}_y$, $\mathbf{B}_z$ in the MGNFIELD option (p. 151), given by WHAT(4)...WHAT(6) as described above, are the field components and not the cosines.
3. Magnetic field tracking is performed only in regions defined as magnetic field regions by command ASSIGNMAt (p. 62). It is strongly recommended to define as such only regions where a non-zero magnetic field effectively exists, due to the less efficient and accurate tracking algorithm used in magnetic fields.
   To define a region as having a magnetic field and to return systematically `B = 0.0` in that region via subroutine `MAGFLD`, is not allowed.
4. The maximum error on the boundary iteration, WHAT(2), must be compatible with the minimum linear dimension of any region.
5. It is recommended to activate also option STEPSIZE (p. 200) inside and close to regions where a magnetic field is present. That option can be used to set a minimum and a maximum step size (in cm) for every region.
6. In case of conflict, WHAT(3) overrides the step size requested by option STEPSIZE. Therefore, it is suggested to set it not larger than the latter. The purpose of this constraint is to avoid tracking in detail low-energy particles along a helix of very small radius, by forcing several turns into a single step (all the energy will be deposited at the same point).

Example:

```
*...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+...
ASSIGNMAT        2.0      15.0      30.0       5.0       1.0       0.0
* A magnetic field is present in vacuum regions 15, 20, 25 and 30.
MGNFIELD        20.0       0.2      0.10       0.0       0.0       0.0
STEPSIZE       -0.05       0.0      20.0      25.0       0.0       0.0
STEPSIZE         0.3       0.0      15.0       0.0       0.0       0.0
* The maximum deviation angle due to magnetic field in any step is set
* = 20 degrees, and boundary crossings must be identified with an error
* not larger than 2 mm. If the max. angle constraint forces the step to
* be shorter than 10 cm, the step will be set = 10 cm in region 20, 25, 30,
* but will be set = 3 mm in region 15 (WHAT(1) of STEPSIZE overrides the
* general setting due to WHAT(3) of MGNFIELD). Whatever the size of the
* step, however, the accuracy of the boundary crossing position must be
* equal or better than 0.5 mm in regions 20 and 25 (probably contiguous,
* since the same accuracy must be set for regions on either side of a
* boundary). The value of the magnetic field will be provided at each
* step by the user routine MAGFLD.
```

7.46 MULSOPT

Sets the tracking conditions for multiple Coulomb scattering (MCS), for both hadrons/muons and e^+e^-. *Can also be used to activate single scattering.*

See also EMFFIX, FLUKAFIX, MCSTHRES, STEPSIZE

For SDUM ≠ GLOBAL, GLOBEMF, GLOBHAD:

WHAT(1) : controls the step optimisation for multiple Coulomb scattering (see Note 1) and the number of single scatterings on a material by material basis

≤ `-1.0`: a possible previous request of optimisation is cancelled and the number of single scatterings in the materials indicated by WHAT(4)–WHAT(6) is reset to the default value (i.e., `0.` or the global default possibly set previously by this option with SDUM = GLOBAL/GLOBHAD/GLOBEMF)

= `0.0`: ignored

= $i_0 + i_1 \times 10 + i_2 \times 100000$, with $0 \le i_0 \le 1$, $0 \le i_1 \le 10000$, $0 \le i_2 \le 10000$:

$i_0 \ge 1$: the optimisation is activated

$i_1 - 1$ = number of single scattering steps for hadrons and muons in the materials indicated by WHAT(4)–WHAT(6)

$i_1 = 0$: ignored

$i_2 - 1$ = number of single scattering steps for electrons and positrons in the materials indicated by WHAT(4)–WHAT(6)

$i_2 = 0$: ignored

Default = `-1.0` (no multiple scattering optimisation and no single scattering)

|WHAT(2)| = `1.0`: spin-relativistic corrections are activated for charged hadrons and muons at the 1^{st} Born approximation level

|WHAT(2)| = `2.0`: spin-relativistic corrections are activated for hadrons and muons at the 2^{nd} Born approximation level

WHAT(2) < `0.0`: nuclear finite size effects (form factors) are activated (see Note 2).

= `-3.0`: nuclear finite size effects are considered but not the spin-relativistic effects

≥ `3.0`: multiple scattering for hadrons and muons is completely suppressed (see Note 3).

Default = `0.0` (no corrections)

|WHAT(3)| = `1.0`: spin-relativistic corrections activated for $e^{\pm}$ in the 1^{st} Born approximation

|WHAT(3)| = `2.0`: spin-relativistic corrections activated for $e^{\pm}$ in the 2^{nd} Born approximation

WHAT(3) < `0.0`: nuclear finite size effects are activated

≥ `3.0`: multiple scattering for e^+ and e^- is completely suppressed

Default = `0.0` (no corrections)

WHAT(4) = lower bound of the material indices in which the corrections are activated ("*From material WHAT(4)...*")

Default = `3.0`

WHAT(5) = upper bound of the material indices in which the corrections are activated ("*...to material WHAT(5)...*")

Default = WHAT(4)

WHAT(6) = step length in assigning indices.
("*... in steps of WHAT(6)*")
Default = 1.0

SDUM = FANO–ON : Fano correction for inelastic interactions of charged hadrons and muons on atomic electrons [48] is switched on
= FANO–OFF : Fano correction for inelastic interactions of charged hadrons and muons on atomic electrons is switched off
= MLSH–ON : Molière screening angle on for hadrons and muons
= MLSH–OFF : Molière screening angle for hadrons and muons as modified by Berger & Seltzer for electrons
= blank : see default
Default : Fano correction on, original Molière screening angle for hadrons on

Default (option MULSOPT not given): no MCS optimisation

For SDUM = GLOBAL, GLOBEMF, GLOBHAD:

(GLOBEMF restricts the input value to e^+e^-, GLOBHAD to charged hadrons and muons)

WHAT(1) : controls the minimum MCS step size used by the boundary approach algorithm for $e^+e^-$and heavy charged particles
$\geq$ 0.0 and $<$ 0.2: ignored
$\geq$ 0.2: the minimum step size is set equal to the size corresponding to B = 5 in Molière theory, multiplied by WHAT(1)
$<$ 0.0: the minimum step size is reset to default
Default = 1.0 (maximum accuracy)

WHAT(2) : index of step stretching factor tabulation to be used by the electron/positron transport algorithm when approaching a boundary.
The values of the index implemented for the moment are 1,2,3,4.
Values 11,12,13,14 cause the sensing algorithm to multiply the range/MCS step rather than the current step.
Values 101,111,102,112,103,113,104,114 have the additional effect of making the algorithm resample as unphysical any step cut at a boundary and "reflected" from the boundary.
= 0.0: ignored
$<$ 0.0: the tabulation index is reset to default
Default = 1.0 (maximum accuracy)

WHAT(3) : controls the optimal step to be used by the optimisation option (and to some extent by the hadron/muon boundary approach algorithm).
$\geq$ 0.0 and $<$ 0.2: ignored
$\geq$ 0.2: the minimum step size is set equal to the size corresponding to B = 5 in Molière theory [22, 113–115], multiplied by WHAT(3)
$<$ 0.0: the minimum step is reset to its default value
Default : minimum step size equal to that corresponding to B = 5, multiplied by 20.0

WHAT(4) $>$ 0.0: single scattering option activated at boundaries or for too short steps
$<$ 0.0: resets to default
= 0.0: ignored
Default : single scattering is not activated

WHAT(5) (meaningful only if single scattering is activated at boundaries and when the step is too short: see WHAT(4) above)
$>$ 0.0: single scattering option activated for energies too small for Molière theory to apply

$<$ 0.0: single scattering is not activated
$=$ 0.0: ignored
Default : single scattering is not activated

WHAT(6) (meaningful only if single scattering is activated at boundaries and when step is too short: see WHAT(4) above)
$>$ 0.0: number of single scatterings to be performed when crossing a boundary
$=$ 0.0: ignored
$<$ 0.0: resets the default
Default $=$ 1.0

Notes

1. When optimisation is requested, the program always makes the minimum step for which the Molière theory of multiple scattering is applicable. Optimisation via MULSOPT is available only for charged hadrons and muons. For electrons and positrons, option EMFFIX is recommended (p. 107).
2. The correction for the nuclear finite size has been implemented using simple Thomas-Fermi form factors according to Tsai [170]. The user can provide more sophisticated values by supplying a function FORMFU which must return the square of the nuclear form factor. See 12.2.7.
3. Complete suppression of multiple scattering can be useful in some particular cases, for instance when replacing a gas of extremely low density by a gas of the same composition but of much larger density in order to increase the frequency of inelastic interactions or bremsstrahlung reactions (of course, the results must then be scaled by the density ratio). In such cases, one should also select the biased density so that no re-interaction of secondaries can take place.
4. Runs in which the nuclear form factor is taken into account and/or the 2^{nd} Born approximation is requested are very CPU-time consuming at low energy (but not at high energy).
5. Setting WHAT(6) $>$ 1000.0 with SDUM $=$ GLOBAL, GLOBHAD or GLOBEMF, replaces systematically multiple scattering with single scattering everywhere. This choice is generally extremely demanding in CPU time, except for particles of very low energy (a few keV), which have a very short history anyway. In such cases, the single scattering option is even recommended [59].

Example 1:

```
* Activate spin-relativistic corrections and nuclear finite size effects
* for heavy charged particles in the first Born approximation.
* Activate spin-relativistic corrections but not nuclear size effects
* for electrons and positrons in materials 5, 10 and 15
*...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+...
MULSOPT          1.0      -1.0       2.0       5.0      15.0       5.0
```

Example 2:

```
* Maximum accuracy requested for the electron step size used in the boundary
* approach and in the optimisation algorithm. Single scattering activated for
* electrons at boundary crossing and when the step is too short for Moliere
* (but not when the energy is too low for Moliere). Boundaries will be
* crossed with 2 single scatterings.
*...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+...
MULSOPT          1.0       1.0       1.0       1.0       0.0       2. GLOBEMF
```

Example 3:

```
* Single scattering activated everywhere for all charged particles
 *...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+...
MULSOPT          0.0       0.0       0.0       1.0       1.0 99999999.GLOBAL
```

7.47 MUPHOTON

Controls muon photonuclear interactions

See also PAIRBREM, PHOTONUC

WHAT(1) : flag to switch on muon nuclear interactions:
= `-1.0`: resets to default (muon nuclear interactions are not simulated at all)
= `0.0`: ignored
= `1.0`: full simulation of muon nuclear interactions and production of secondary hadrons
= `2.0`: muon nuclear interactions are simulated but no secondary hadron is produced; the energy lost by the muon is deposited at the point of interaction
Default = `0.0` (ignored)

WHAT(2) : reserved for code development

WHAT(3) : reserved for code development

WHAT(4) = lower bound of the indices of materials in which muon nuclear interactions must be simulated ("*From material WHAT(4)...*")
Default = `3.0`

WHAT(5) = upper bound of the indices of materials in which muon nuclear interactions must be simulated ("*... to material WHAT(5)...*")
Default = `WHAT(4)`

WHAT(6) = step length in assigning indices ("*... in steps of WHAT(6)*")
Default = `1.0`

SDUM : not used

Default (option MUPHOTON not given): muon nuclear interactions are not simulated

Notes

1. Other high-energy interactions of muons with nuclei (pair production, bremsstrahlung) are controlled by option PAIRBREM (p. 170), which applies also to charged hadrons.
2. Use of WHAT(1) = `2.0` (interaction without transport of the secondaries) gives the correct muon straggling but simulates only in an approximate way the energy deposition distribution. A similar approach is found in A. Van Ginneken's codes CASIM and MUSIM [172, 173].

Example:

```
* Explicit pair production and bremsstrahlung requested for heavy charged
* particles in materials 12 and 13, provided the energy of the secondary
* electrons and positrons is > 500 keV. No threshold is requested for photon
* production. For muons, explicit nuclear interactions are also requested.
*...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+...
PAIRBREM         3.0       0.0    0.0005      12.0      13.0
MUPHOTON         1.0       0.0       0.0      12.0      13.0
```

7.48 MYRQMD

Not yet implemented. Prepared for new QMD generator

7.49 OPEN

Defines input/output files to be connected at run-time.

WHAT(1) > 0.0: logical unit number of a *formatted* file to be opened
< 0.0: logical unit number of an *unformatted* file to be opened
Default : no default (WHAT(1) must not be = 0.0)

WHAT(2)– **WHAT(6)**: not used

SDUM = NEW : the file is opened with status NEW
= OLD : the file is opened with status OLD
= UNKNOWN : the file is opened with status UNKNOWN
= READONLY : the file is opened with status OLD and (only on VAX VMS/OpenVMS) in mode READONLY
= SCRATCH : the file is opened with status SCRATCH
Default = OLD if |WHAT(1)| = 9, 12, 13, 14, otherwise = NEW
Unless SDUM = SCRATCH, the name of the file to be opened must be given in the card which immediately follows.

Default (option OPEN not given): no file is opened at run time. In this case the I/O files must be pre-connected via ASSIGN on VAX VMS/OpenVMS (see 3). On UNIX and Linux, the **rfluka** script provided with the FLUKA code creates the necessary symbolic links. On some UNIX systems (e.g., HPUX–9), OPEN *must* be given in any case for the data files. See more details in Note 1.

Notes

1. The input/output files used by FLUKA are of several kinds:
   - Standard input (logical unit 5) and standard output (logical unit 11) must be redirected via < and > (on UNIX and Linux), or pre-connected via FILEDEF, ASSIGN, etc. on other systems.
   - Cross-section unformatted data files (logical unit numbers 9, 13 and 14) can be opened with the OPEN option (SDUM = OLD or READONLY), or can be pre-connected (on most UNIX systems, pre-connection is obtained by means of symbolic links). If OPEN is used, the full file name must be given in the card which follows.
   - The same is true for the initial seeds file for the random number generator, if used (logical unit number defined by card RANDOMIZE). However, while WHAT(1) must be negative for unformatted data files, it must be positive for the seeds data.
   - Scratch files (unit 8 for EMF auxiliary output and unit 16 for Combinatorial Geometry working space, see p. 280) can also be OPENed (with SDUM = SCRATCH) or (not on UNIX) pre-connected. No file name card must be given for scratch files.
   - The "next seeds" file from the random number generator (logical unit number 2, see p. 280) can be opened by any of the three ways described above (i.e., by OPEN, by pre-connection or automatically) on any of the supported systems.
   - Error message file (logical unit number 15, p. 281) and estimator output files (created by scoring options such as USRBIN, USRBDX, DETECT etc., p. 282):
     - On UNIX systems, they can either be opened by the user (with option OPEN, SDUM = NEW or UNKNOWN and file name given on the next card) or automatically by the program with a default name of the form **fort.xxx** or **ftn.xxx**, where **xxx** is the logical unit number.
     - On VAX VMS/OpenVMS, all the three possibilities are available: pre-connection, OPEN option and automatic opening with a default name of the form **FORxxx.DAT**.
   - Files created by user-written code (Chap. 12): all three possibilities are available. Of course, a Fortran statement OPEN can also be used in this case.
2. It is possible to pre-connect some of the files and to OPEN others.

Examples:

```
*...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+...
*   opening the file with the random number seeds for the next run
OPEN              2.                                                  NEW
newseed.random
*   the neutron cross-section file (on VAX)
OPEN             -9.                                                  READONLY
neuxsc_72.bin
*   the working space for Combinatorial Geometry
OPEN              16.                                                 SCRATCH
```

7.50 OPT–PROD

Requests and controls production of Cherenkov, Transition and Scintillation Radiation in specified materials.

See also OPT–PROP, Chap. 13, and examples in 13.2

For SDUM = CERE–OFF: switches off Cherenkov production

WHAT(1) – **WHAT(3)**: not used

WHAT(4) – **WHAT(6)**: assignment to materials, see below

SDUM = CERE–OFF

For SDUM = TRD–OFF: switches off Transition Radiation production

WHAT(1) – **WHAT(3)**: not used

WHAT(4) – **WHAT(6)**: assignment to materials, see below

SDUM = TRD–OFF

For SDUM = SCIN–OFF: switches off Scintillation light production

WHAT(1) – **WHAT(3)**: not used

WHAT(4) – **WHAT(6)**: assignment to materials, see below

SDUM = SCIN–OFF

For SDUM = CERENKOV: switches on Cherenkov production and defines photon energy range

WHAT(1) = minimum Cherenkov photon emission energy in GeV
Default = 2.07×10^{-9} (2.07 eV, corresponding to 600 nm)

WHAT(2) = maximum Cherenkov photon emission energy in GeV
Default = 4.96×10^{-9} (4.96 eV, corresponding to 250 nm)

WHAT(3) : not used

WHAT(4) – **WHAT(6)**: assignment to materials, see below

SDUM = CERENKOV

For SDUM = CEREN–WV: switches on Cherenkov production and defines photon wavelength range

WHAT(1) = minimum Cherenkov photon emission wavelength in cm

Default = 2.50×10^{-5} (250 nm, or 1.2×10^{6} GHz)

WHAT(2) = maximum Cherenkov photon emission wavelength in cm

Default = 6.00×10^{-5} (600 nm, or 5×10^{5} GHz)

WHAT(3) : not used

WHAT(4) – **WHAT(6)**: assignment to materials, see below

SDUM = CEREN–WV

For SDUM = CEREN–OM: switches on Cherenkov production and defines photon angular frequency range

WHAT(1) = minimum Cherenkov photon angular frequency $\omega = 2\pi\nu$ in rad/s (ν = frequency) in rad/s

Default = 3.14×10^{15} rad/s (corresponding to 600 nm)

WHAT(2) = maximum Cherenkov photon emission angular frequency $\omega = 2\pi\nu$ in rad/s

Default = 7.53×10^{15} rad/s (corresponding to 250 nm)

WHAT(3) : not used

WHAT(4) – **WHAT(6)**: assignment to materials, see below

SDUM = CEREN–OM

For SDUM = TR–RADIA: switches on Transition Radiation production and defines its energy range

WHAT(1) = minimum TRD photon emission energy

WHAT(2) = maximum TRD photon emission energy

WHAT(3) : not used

WHAT(4) – **WHAT(6)**: assignment to materials, see below

SDUM = TR–RADIA

For SDUM = SCINTILL: switches on Scintillation Light production and defines photon energy

WHAT(1) = i_{th} scintillation photon emission energy in GeV (i_{max}=3, see Note 4)

WHAT(2) $>$ 0: fraction of deposited energy going into i_{th} scintillation photon emission
$\leq$ `-100`: forces to use a user routine **(not yet implemented)**
$\geq$ `-99.0` and $\leq$ `0.0`: ignored

WHAT(3) : time constant of scintillation light in seconds

WHAT(4) – **WHAT(6)**: assignment to materials, see below

SDUM = SCINTILL

For SDUM = SCINT–WV: switches on Scintillation Light production and defines photon wavelength

WHAT(1) = i_{th} scintillation photon emission wavelength in cm (i_{max}=3, see Note 4)
Default = 2.50×10^{-5} (250 nm, or 1.2×10^{6} GHz)

WHAT(2) > 0: fraction of deposited energy going into i_{th} scintillation photon emission
≤ `-100`: forces to use a user routine **(not yet implemented)**
≥ `-99.0` and ≤ `0.0`: ignored

WHAT(3) : time constant of scintillation light in seconds

WHAT(4) – **WHAT(6)**: assignment to materials, see below

SDUM = SCINT–WV

For SDUM = SCINT–OM: switches on Scintillation Light production and defines photon angular frequency range

WHAT(1) = i_{th} scintillation photon emission angular frequency $\omega = 2\pi\nu$ in rad/s, ν = frequency. (i_{max}=3, see Note 4)
Default = 3.14×10^{15} rad/s (corresponding to 600 nm)

WHAT(2) =fraction of deposited energy going into i_{th} scintillation photon emission
≤ `-100`: forces to use a user routine **(not yet implemented)**
≥ `-99.0` and ≤ `0.0`: ignored

WHAT(3) : time constant of scintillation light in seconds

WHAT(4) – **WHAT(6)**: assignment to materials, see below

SDUM = SCINT–OM

For all previous SDUMs:

WHAT(4) = lower bound of the indices of materials in which the indicated Cherenkov, Scintillation or TRD photon emission range is defined
("*From material WHAT(4)...*")
Default = `3.0`

WHAT(5) = upper bound of the indices of materials in which the indicated Cherenkov, Scintillation or TRD photon emission range is defined
("*... to material WHAT(5)...*")
Default = WHAT(4)

WHAT(6) = step length in assigning indices
("*... in steps of WHAT(6)*")
Default = `1.0`

Default : (option OPT–PROD not given): no Cherenkov, scintillation or TRD photon production

Notes

1. Optical photons such as those produced by Cherenkov effect are distinguished by their FLUKA name (OPTIPHOT) and by their FLUKA id-number (-1), as shown in 5.1.
2. To transport optical photons, it is necessary to define the optical properties of the relevant materials by means of option OPT–PROP (p. 164). Users can also write their own routines USRMED (p. 323), which is called at every step and at boundary crossings when activated with MAT–PROP (p. 144), and FRGHNS (p. 307), which defines surface roughness.
3. The energy/wavelength/frequency range as defined by OPT–PROD for Cherenkov photon production is not necessarily the same as that defined for transport by means of OPT–PROP. The default values, however, are the same.
4. In case of scintillation light, only monochromatic photons are considered for the moment, with a maximum of $i = 3$ different lines. The lines can be defined repeating i times the OPT–PROD card with SDUM = SCINTILL.

Example:

```
* Request production of Cherenkov photons with energies between 2 and 3 eV in
* materials 16, 17, 19 and 20, with wavelengths between 300 and 600 nm in
* materials 18, 20 and 22, and with frequencies between 0.5 and 1 million GHz
* in material 21
*...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+...
OPT-PROD       2.E-9     3.E-9       0.0      16.0      17.0       0. CERENKOV
OPT-PROD       2.E-9     3.E-9       0.0      19.0      20.0       0. CERENKOV
OPT-PROD       3.E-5     6.E-5       0.0      18.0      22.0       2. CEREN-WV
OPT-PROD     3.14E15   6.28E15       0.0      21.0       0.0       0. CEREN-OM
* Optical photon transport requested between 300 and 500 nm for all materials
* with number between 16 and 21
OPT-PROP       3.E-5     5.E-5     6.E-5      16.0      22.0       0. WV-LIMIT
* User routine USRMED called when an optical photon is going to be transported
* in materials 17 and 21
MAT-PROP         1.0       0.0       0.0       17.       21.       4. USERDIRE
```

A more detailed example including scintillation light is reported in section 13.2.

7.51 OPT–PROP

Defines optical properties of specified materials.

See also OPT–PROD, Chap. 13, and examples in 13.2

For SDUM = WV–LIMIT: defines wavelength range for optical photon transport

WHAT(1) $>$ `0.0`: minimum wavelength (in cm) for optical photon transport
$=$ `0.0`: ignored
$<$ `0.0`: resets to default
Default $= 2.5 \times 10^{-5}$ (250 nm)

WHAT(2) $>$ `0.0`: central wavelength (in cm) for optical photon transport
$=$ `0.0`: ignored
$<$ `0.0`: resets to default
Default $= 5.89 \times 10^{-5}$ (589 nm, Na D line)

WHAT(3) $>$ `0.0`: maximum wavelength (in cm) for optical photon transport
$=$ `0.0`: ignored
$<$ `0.0`: resets to default
Default $= 6.0 \times 10^{-5}$ (600 nm)

WHAT(4) – **WHAT(6)**: assignment to materials, see below

SDUM = WV–LIMIT

For SDUM = OM–LIMIT: defines angular frequency range for optical photon transport

WHAT(1) $>$ `0.0`: minimum angular frequency for optical photon transport $\omega = 2\pi\nu$ in rad/s (ν = frequency)
$=$ `0.0`: ignored
$<$ `0.0`: resets to default
Default $= 3.14 \times 10^{15}$ rad/s (corresponding to 600 nm)

WHAT(2) $>$ `0.0`: central angular frequency for optical photon transport $\omega = 2\pi\nu$ in rad/s
$=$ `0.0`: ignored
$<$ `0.0`: resets to default
Default $= 3.20 \times 10^{15}$ rad/s (corresponding to 589 nm, Na D line)

WHAT(3) $>$ `0.0`: maximum angular frequency for optical photon transport $\omega = 2\pi\nu$ in rad/s
$=$ `0.0`: ignored
$<$ `0.0`: resets to default
Default $= 7.53 \times 10^{15}$ rad/s (corresponding to 250 nm)

WHAT(4) – **WHAT(6)**: assignment to materials, see below

SDUM = OM–LIMIT

For SDUM = RESET: all material optical properties are zeroed

WHAT(1) – **WHAT(3)**: no meaning

WHAT(4) – **WHAT(6)**: assignment to materials, see below

SDUM = RESET

For SDUM = METAL: flag the material as a metal

WHAT(1) = 1^{st} optical property **(not used at the moment)**

WHAT(2) = 2^{nd} optical property **(not used at the moment)**

WHAT(3) = 3^{rd} optical property: $(1 - r)$, where r is the reflectivity index at the central wavelength (or at the central angular frequency, depending on which one of the two quantities has been defined). See also Note 2

Default = 0.0

WHAT(4) – **WHAT(6)**: assignment to materials, see below

SDUM = METAL

For SDUM = *blank*:

WHAT(1) = 1^{st} optical property: refraction index n_{refr} at the central wavelength (or at the central angular frequency, depending on which one of the two quantities has been defined)

< `-99:` forces to use user routine `RFRNDX` (see Note 1)

Default = 1.0

WHAT(2) = 2^{nd} optical property: absorption coefficient μ_{abs} (in cm^{-1}) at the central wavelength (or at the central angular frequency, depending on which one of the two quantities has been defined)

< `-99:` forces to use a user routine `ABSCFF` (see Note 1)

Default = 0.0

WHAT(3) = 3^{rd} optical property: diffusion coefficient μ_{diff} (in cm^{-1}) at the central wavelength (or at the central angular frequency, depending on which one of the two quantities has been defined)

< `-99:` forces to use a user routine `DFFCFF` (see Note 1)

Default = 0.0

WHAT(4) – **WHAT(6)**: assignment to materials, see below

SDUM = *blank*

For SDUM containing "`&1`" (resp. "`&2`") (resp. "`&3`"):

WHAT(1) = 4^{th} (resp. 7^{th}) (resp. 10^{th}) optical property of the material (derivatives of the refraction index, see Note 2)

Default = 0.0

WHAT(2) = 5^{th} (resp. 8^{th}) (resp. 11^{th}) optical property of the material (derivatives of the absorption coefficient, see Note 2)

Default = 0.0

WHAT(3) = 6^{th} (resp. 9^{th}) (resp. 12^{th}) optical property of the material (derivatives of the diffusion coefficient, see Note 2)

WHAT(4) – **WHAT(6)**: assignment to materials, see below

SDUM = `&1`, `&2` or `&3` in any position in column 71 to 78 (or in the last field if free format is used)

For all previous SDUMs:

WHAT(4) = lower bound of the indices of materials to which the indicated optical properties refer ("*From material WHAT(4)...*")
Default = `3.0`

WHAT(5) = upper bound of the indices of materials to which the indicated optical properties refer ("*... to material WHAT(5)...*")
Default = WHAT(4)

WHAT(6) = step length in assigning indices ("*... in steps of WHAT(6)*")
Default = `1.0`

For SDUM = SENSITIV: sets up the optical photon detection sensitivity parameters

(See also SDUM = WV–SENSI, SDUM = OM–SENSI, Note 3 and the examples in 13.2)

WHAT(1) = 0^{th} optical photon sensitivity parameter
`<-99:` forces to use user routine `QUEFFC` (see Note 1)

WHAT(2) = 1^{st} optical photon sensitivity parameter

WHAT(3) = 2^{nd} optical photon sensitivity parameter

WHAT(4) = 3^{rd} optical photon sensitivity parameter

WHAT(5) = maximum optical photon sensitivity over the allowed range (must be consistent with the previous values). It can be overestimated.
Default = `1.0`

WHAT(6) = not used

SDUM = SENSITIV

For SDUM = WV–SENSI: sets up the wavelength of the optical photon sensitivity

(See also SDUM = SENSITIV, SDUM = OM–SENSI and Note 3.)

WHAT(1) `>0.0:` minimum wavelength (in cm) for optical photon sensitivity
`=0.0:` ignored
`<0.0:` resets to default
Default = 2.5×10^{-5} (250 nm)

WHAT(2) `>0.0:` central wavelength (in cm) for optical photon sensitivity
`=0.0:` ignored
`<0.0:` resets to default
Default = 5.89×10^{-5} (589 nm, Na D line)

WHAT(3) `>0.0:` maximum wavelength (in cm) for optical photon sensitivity
`=0.0:` ignored

$<$ `0.0`: resets to default

Default $=$ 6.0×10^{-5} (600 nm)

WHAT(4) – **WHAT(6)**: not used

SDUM = WV–SENSI

For SDUM = OM–SENSI: sets up the angular frequency of the optical photon sensitivity

(See also SDUM = SENSITIV, SDUM = WV–SENSI and Note 3.)

WHAT(1) $>$ `0.0`: minimum angular frequency for optical photon sensitivity $\omega = 2\pi\nu$ in rad/s (ν = frequency)

$=$ `0.0`: ignored

$<$ `0.0`: resets to default

Default $=$ 3.14×10^{15} rad/s (corresponding to 600 nm)

WHAT(2) $>$ `0.0`: central angular frequency for optical photon sensitivity $\omega = 2\pi\nu$ in rad/s

$=$ `0.0`: ignored

$<$ `0.0`: resets to default

Default $=$ 3.20×10^{15} rad/s (corresponding to 589 nm, Na D line)

WHAT(3) $>$ `0.0`: maximum angular frequency for optical photon sensitivity $\omega = 2\pi\nu$ in rad/s

Default $=$ 7.53×10^{15} rad/s (corresponding to 250 nm)

WHAT(4) – **WHAT(6)**: not used

SDUM = OM–SENSI

For SDUM = SPEC–BDX: flags special boundary crossings for optical photons

At the selected boundary crossings special user defined properties are defined by means of the user routine `OPHBDX` (see Note 1).
A maximum of 40 boundaries can be flagged by issuing option OPT–PROP with SDUM = SPEC–BDX as many times as needed.

WHAT(1) $\geq$ `1.0`: special boundary treatment activated for the n_{th}+1 boundary

$=$ `0.0`: ignored

$\leq$ `-1.0`: special boundary treatment deactivated for the n_{th}+1 boundary

WHAT(2) = one of the two regions defining the n_{th}+1 boundary

WHAT(3) = the other region defining the n_{th}+1 boundary

WHAT(4) $\geq$ `1.0`: special boundary treatment activated for the n_{th}+2 boundary

$=$ `0.0`: ignored

$\leq$ `-1.0`: special boundary treatment deactivated for the n_{th}+2 boundary

WHAT(5) = one of the two regions defining the n_{th}+2 boundary

WHAT(6) = the other region defining the n_{th}+2 boundary

SDUM = SPEC–BDX

Default : (option OPT–PROP not given): no optical photon transport

Notes

1. The optional user routines concerning optical photons are:
   i) `RFRNDX`: to specify a refraction index as a function of wavelength, frequency or energy. Activated by setting WHAT(1) < -99 when SDUM =*blank*.
   ii) `ABSCFF`: to specify an absorption coefficient as a function of wavelength, frequency or energy. This is activated by setting WHAT(2) < -99 when SDUM =*blank*.
   iii) `DFFCFF`: to specify a diffusion coefficient as a function of wavelength, frequency or energy. Activated by setting WHAT(3) < -99 when SDUM =*blank*.
   iv) `RFLCTV`: to specify the reflectivity of a material. This can be activated with SDUM = METAL and WHAT(3) < -99
   v) `OPHBDX`: to set optical properties of a boundary surface. The call is activated with SDUM = SPEC–BDX
   vi) `FRGHNS`: to set a possible degree of surface roughness, in order to have both diffusive and specular reflectivity from a given material surface **(not yet implemented)**
   vii) `QUEFFC`: to introduce Quantum Efficiency. This is activated by setting the 0^{th} photon sensitivity parameter to a value < -99 with SDUM = SENSITIV.
2. The 9 material properties input by the user with SDUM = &1, &2 and &3 are derivatives of order 1 to 3 of the three basic quantities which are input with SDUM = *blank* or SDUM = METAL. The three basic quantities and their derivatives are used to perform three series expansions centred on the selected central wavelength (if SDUM = WV–LIMIT has been used for the material concerned) or on the selected central angular frequency (if SDUM = OM–LIMIT has been used). The variable x used in the expansion is adimensional:

$$x = \frac{\lambda - \lambda_{central}}{\lambda_{central}} \qquad \text{or} \qquad x = \frac{\omega - \omega_{central}}{\omega_{central}}$$

   The three basic quantities are:

If SDUM = *blank*:	If SDUM = METAL:
(a) refraction index n_{refr}	(a) (not implemented yet)
(b) absorption coefficient μ_{abs}	(b) (not implemented yet)
(c) diffusion coefficient μ_{diff}	(c) $(1-r)$ (r = reflectivity index)

   The 4^{th}, 5^{th}, 6^{th} material properties are: $\frac{\mathrm{d}n_{refr}}{\mathrm{d}x}$, $\frac{\mathrm{d}\mu_{abs}}{\mathrm{d}x}$, $\frac{\mathrm{d}\mu_{diff}}{\mathrm{d}x}$

   the 7^{th}, 8^{th}, 9^{th} material properties are: $\frac{\mathrm{d}^2 n_{refr}}{\mathrm{d}x^2}$, $\frac{\mathrm{d}^2 \mu_{abs}}{\mathrm{d}x^2}$, $\frac{\mathrm{d}^2 \mu_{diff}}{\mathrm{d}x^2}$

   the 10^{th}, 11^{th}, 12^{th} material properties are: $\frac{\mathrm{d}^3 n_{refr}}{\mathrm{d}x^3}$, $\frac{\mathrm{d}^3 \mu_{abs}}{\mathrm{d}x^3}$, $\frac{\mathrm{d}^3 \mu_{diff}}{\mathrm{d}x^3}$
3. SDUM = SENSITIV can be used to set the quantum efficiency as a function of photon energy *overall* through the problem and it is not material/region dependent. The reason is that it is applied "a priori" at photon generation time. If a quantum efficiency curve is to be introduced *at detection*, then a weighting user routine should be used, such as `FLUSCW` or `COMSCW`. It is advantageous to reduce computer time avoiding to generate and transport a photon which would have a significant probability to remain undetected.
   The optical photon sensitivity parameters are: $\epsilon(0)$, $\frac{\mathrm{d}\,\epsilon}{\mathrm{d}\,x}$, $\frac{\mathrm{d}^2\,\epsilon}{\mathrm{d}\,x^2}$, $\frac{\mathrm{d}^3\,\epsilon}{\mathrm{d}\,x^3}$ where x is:

$$x = \frac{\lambda - \lambda_{central}}{\lambda_{central}} \qquad \text{or} \qquad x = \frac{\omega - \omega_{central}}{\omega_{central}}$$

Example 1:

```
* Optical photon transport requested between 3.E15 and 7.E15 rad/s
* (4.77E5 and 1.11E6 GHz, or 314 to 628 nm) for materials 6,9,12,15 and 18
*...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+...
OPT-PROP       3.E15     6.E15     7.E15       6.0      18.0       3. OM-LIMIT
* User routine USRMED called when an optical photon is going to be transported
* in materials 6, 12 and 18
MAT-PROP         1.0       0.0       0.0       6.0      18.0       6. USERDIRE


Example 2:
* Material 11 has a reflectivity index = 0.32
*...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+...
OPT-PROP         0.0       0.0      0.32      11.0       0.0       0. METAL
```

Example 3:

```
*...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+...
* Optical photon transport requested between 300 and 600 nm for water.
* (material 9). The optical properties are for the Na D line (589 nm)
MATERIAL         1.0       1.0  .0000899       3.0       0.0       1. HYDROGEN
MATERIAL         8.0      16.0   0.00143       5.0       0.0       0. OXYGEN
MATERIAL         0.0       0.0       1.0      21.0       0.0       0. WATER
COMPOUND         2.0       3.0       1.0       5.0       0.0       0. WATER
OPT-PROP       3.E-5   5.89E-5     6.E-5      21.0       0.0       0. WV-LIMIT
* diffusion coefficient of water from Appl.Opt. 17, 3587 (1978)
OPT-PROP     1.33299    0.0013     1.220      21.0       0.0       0.
```

7.52 PAIRBREM

Controls simulation of pair production and bremsstrahlung by high-energy muons and charged hadrons

See also MUPHOTON

WHAT(1) = `0.0`: ignored
= `1.0`: pair production by muons and charged hadrons is activated
= `2.0`: bremsstrahlung by muons and charged hadrons is activated
= `3.0`: pair production and bremsstrahlung are both activated
= `-1.0`: pair production is inhibited
= `-2.0`: bremsstrahlung is inhibited
= `-3.0`: pair production and bremsstrahlung are both inhibited (see Note 1).
Default = `3.0` (both pair production and bremsstrahlung are activated).
However, if option DEFAULTS is present with SDUM = EET/TRANsmut, EM–CASCAde, NEUTRONS or SHIELDINg, the default is = `-3.0` (pair production and bremsstrahlung inhibited)

WHAT(2) $\geq$ `0.0`: e^+, e^- kinetic energy threshold (in GeV) for explicit pair production. A value of `0.0` is meaningful (it does mean zero energy) and is even recommended (see Note 2).
$<$ `0.0`: no explicit pair production (overrides a possible non-negative value set in a previous PAIRBREM card), but the energy loss due to pair production is taken into account in a continuous approximation (see Note 3).
Default = `-1.0` (no explicit pair production).
However, if DEFAULTS is not present, or is present with SDUM = CALORIMEtry, ICARUS, NEW–DEFAults or PRECISIOn, the default is = `0.0` (explicit pair production with zero threshold).

WHAT(3) $>$ `0.0`: photon energy threshold (GeV) for explicit bremsstrahlung production (see Note 2).
$\leq$ `0.0`: no explicit bremsstrahlung production is simulated (it overrides a possible positive value set in a previous PAIRBREM card), but the energy loss due to bremsstrahlung is taken into account in a continuous approximation (see Note 3).
Default = `-1.0` (no explicit bremsstrahlung production)
However, if DEFAULTS is not present, or is present with SDUM = NEW–DEFAults, the default is = `0.001` (explicit bremsstrahlung production with threshold 1 MeV).
If DEFAULTS has SDUM = CALORIMEtry, ICARUS or PRECISIOn, the default is = `0.0003` (explicit bremsstrahlung production with threshold 300 keV).

WHAT(4) = lower bound of the indices of the materials to which the choices expressed by WHAT(1)...WHAT(3) apply
("*From material WHAT(4)...*")
Default = `3.0`

WHAT(5) = upper bound of the indices of materials to which the choices expressed by WHAT(1)...WHAT(3) apply
("*...to material WHAT(5)...*")
Default = `WHAT(4)`

WHAT(6) = step length in assigning indices
("*...in steps of WHAT(6)*")
Default = `1.0`

SDUM : reserved for program development

Default (option PAIRBREM not given): if option DEFAULTS is not present, or is present with SDUM = NEW–DEFAults, both pair production and bremsstrahlung are activated in all materials, with explicit generation of secondaries of energy ≥ 0 and ≥ 1 MeV, respectively.
If DEFAULTS is present with SDUM = CALORIMEtry, ICARUS or PRECISIOn, both pair production and bremsstrahlung are activated in all materials, with explicit generation of secondaries of energy ≥ 0 and ≥ 300 keV, respectively.
In any other case, both pair production and bremsstrahlung are activated in all materials, without explicit generation of secondaries (continuous loss approximation)

Notes

1. Initialisation of bremsstrahlung and pair production by heavy charged particles is very demanding in computer time. On the other hand, these effects must be taken into account for a correct simulation at high energies. It is suggested to inhibit them in the phase of input preparation and debugging, but to activate them again in production runs (in long runs the time taken by initialisation is of course a smaller fraction of total time). In pure electron-photon problems and in low-energy hadron problems, the effects should be inhibited (WHAT(3) = -3.0)
2. When setting a threshold for pair and bremsstrahlung production by heavy particles, the following considerations should be taken into account:
   - photon production threshold (WHAT(3)) must of course not be lower than the photon transport cut-off as set by EMFCUT (p. 102) or by the chosen default. (In general it will be reasonable to set it equal to it)
   - on the contrary, the electron and positron production threshold (WHAT(2)) should in general be set = 0.0, whatever the electron transport cut-off, unless *photon* transport cut-off is set higher than 511 keV. In this way, if the positron is produced with an energy lower than electron transport cut-off it will be forced to annihilate at the point of production, but the two 511 keV annihilation photons will be generated correctly.
3. If option PAIRBREM is not activated, by default FLUKA treats both bremsstrahlung and pair production by heavy charged particles as continuous energy losses (i.e., without generating secondaries and depositing their energy at the point of production). This will reproduce correctly the average ranges but not the straggling and the dose distributions. A similar approach is found in A. Van Ginneken's code CASIM [122, 172, 173].
4. Virtual photonuclear interactions by high-energy muons and charged hadrons are controlled by option MUPHOTON (p. 156).
5. Here are the settings for pair and bremsstrahlung production by high-energy muons and charged hadrons, corresponding to available DEFAULTS options:

   CALORIMEtry, ICARUS, PRECISIOn:
   pair production is activated in all materials, with explicit generation of secondaries of any energy; bremsstrahlung is also activated in all materials, with explicit generation of photons having energy $\geq$ 300 keV.

   NEW–DEFAults, or DEFAULTS missing:
   pair production is activated in all materials, with explicit generation of secondaries of any energy; bremsstrahlung is also activated in all materials, with explicit generation of photons having energy $\geq$ 1 MeV.

   Any other SDUM value:
   both pair and bremsstrahlung production are activated in all materials, without explicit generation of secondaries (continuous loss approximation) .

Example 1:

```
* Explicit pair production and bremsstrahlung requested for all heavy charged
* particles in materials 4, 7 and 10, provided the energy of the secondary
* electrons and positrons is > 1.2 MeV. No threshold is requested for photon
* production. For muons, explicit nuclear interactions are also requested.
*...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+...
PAIRBREM         3.0       0.0    0.0012       4.0      10.0       3.0
MUPHOTON         1.0       0.0       0.0       4.0      10.0       3.0
```

Example 2:

```
* Energy loss due pair production and bremsstrahlung by heavy charged
* particles accounted for in materials 6, and 7, without explicit generation
* of secondaries
*...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+...
PAIRBREM         3.0      -1.0      -1.0       6.0       7.0
```

7.53 PART–THRes

Sets different energy transport cut-offs for hadrons, muons and neutrinos.

See also EMFCUT, LOW–BIAS, THRESHOLd

The meaning of **WHAT(1)** depends also on the value of **WHAT(5)**:

For WHAT(5) = 0.0:

WHAT(1) $<$ 0.0: kinetic energy cut-off (GeV)
$>$ 0.0: momentum cut-off (GeV/c)

For WHAT(5) $\geq$ 1.0:

WHAT(1) $<$ 0.0: γ cut-off (Lorentz factor $= E/mc^2$)
$>$ 0.0: η cut-off ($\eta = \beta\gamma = v/c \times E/mc^2$)

Default ((WHAT(1) = 0.0): the cut-off is 0 for neutrinos and 0.0196 GeV for neutrons. For any other hadrons, and for muons:

- if option DEFAULTS is missing, or is present with SDUM = NEW–DEFAults or SHIELDINg, the default cut-off kinetic energy is 0.01 GeV.
- if SDUM = HADROTHErapy, ICARUS or PRECISIOn, the default cut-off kinetic energy is 0.0001 GeV.
- If SDUM = CALORIMEtry, the default cut-off kinetic energy is $0.001\frac{m}{m_p}$ GeV (m = particle mass, m_p = proton mass).
- In any other case, the default cut-off is 0.050 GeV

(For low-energy neutrons, the threshold is set by option LOW–BIAS and for e^+e^- and photons by EMFCUT, see Notes 1, 2 and 4 below).

WHAT(2) = lower bound of the particle id-numbers to which the cut-off applies ("*From particle WHAT(2)...*")
Default = 1.0

WHAT(3) = upper bound of the particle id-numbers to which the cut-off applies ("*... to particle WHAT(3)...*")
Default = WHAT(2)

WHAT(4) = step length in assigning numbers ("*... in steps of WHAT(4)*")
Default = 1.0

WHAT(5) : depending on its value, cut-off values indicated by WHAT(1) are assigned to kinetic energy, momentum, γ or η (see WHAT(1))

WHAT(6) = 1.0: restricts the given cut-off to charged particles only
Default : the cut-off applies to all particles indicated by WHAT(2–4)

SDUM : not used

Default (option PART–THRes not given): thresholds as described above for WHAT(1) = 0.0.

Notes

1. When applied to neutrons, the cut-off energy defined by PART–THRes refers to the energy boundary between high-energy and low-energy neutrons, i.e., the upper limit of the first energy group in the multigroup transport scheme (see Chap. 10). The actual cut-off for low-energy neutrons must be set by option LOW–BIAS (p. 135).
2. If PART–THRes is used to set an energy cut-off for high-energy neutrons, and that cut-off is larger than the higher energy boundary of the first low-energy neutron group declared explicitly with LOW–NEUT (p. 140) or implicitly via DEFAULTS (p. 83) , the cut-off is forced to coincide with it. Be careful *not* to set the neutron cut-off *lower* than the higher energy boundary of the first neutron group: the results are unpredictable and there is no check in the program! See Chap. 10 for details on neutron energy groups.
3. If low-energy neutron transport is not requested (explicitly via LOW–NEUT or implicitly via DEFAULTS), the energy of neutrons below threshold is deposited on the spot.
4. Option PART–THR acts on all particles except e^+e^- and photons; when using the EMF option (p. 97) to transport electrons, positrons and photons, the transport cut-off energy is governed by EMFCUT (p. 102).
5. When the energy of a charged particle becomes lower than the cut-off defined by PART–THR, and if such cut-off is lower than 100 MeV, the particle is not stopped, but is ranged out to rest in an approximate way. Its kinetic energy is deposited uniformly over the residual range if the latter is contained within a single region; otherwise a new residual range is calculated at each boundary crossing and the residual kinetic energy is distributed accordingly. If applicable, such a particle eventually decays at rest or is captured. All other forms of transport are ignored except curved paths in magnetic fields (multiple scattering, delta ray production, inelastic or elastic collisions, and *including decay in flight*). Magnetic fields are taken into account but only very roughly, since the continuous slowing down of the particles is not simulated. Antiprotons and π^- are always ranged out to rest (without allowance for decay) and forced to annihilate on a nucleus.
6. If the cut-off is higher than 100 MeV, however, the particles are stopped in place without any further treatment. If this happens at a boundary crossing where the material of the region entered is vacuum, a printed message warns the user that energy is being deposited in vacuum.

Example:

```
* A threshold of 2 MeV (kinetic energy) is requested for heavy charged
* particles with id-numbers between 1 and 11 (protons, antiprotons and
* muons). A threshold of Gamma (= E/m) = 2 will apply for pions and kaons
* (numbers from 13 to 16). For all other particles, the defaults will apply.
*...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+...
PART-THR      -0.002       1.0      11.0       0.0       0.0       1.0
PART-THR        -2.0      13.0      16.0       1.0       1.0       0.0
```

7.54 PHOTONUC

Activates gamma interactions with nuclei.

See also EMF, LAM–BIAS, MUPHOTON

WHAT(1) : flag to switch on nuclear interactions of photons:
$\leq$ `-1.0`: resets to default (no photonuclear interactions)
= `0.0`: ignored
= `1.0`: photonuclear interactions are activated at all energies
= `2.0`: photonuclear interactions are activated only in the high energy range ($>$ 0.7 GeV)
= `3.0`: photonuclear interactions are activated only in the region of the Δ resonance
= `4.0`: only quasi-deuteron interactions are activated
= `5.0`: only Giant Dipole Resonance interactions are activated
$\geq$ `10.0`: interpreted as `ih + id*10 + iq*100 + ig*1000`

where `ih` = `1` to activate High-energy interactions
`id` = `1` to activate Delta Resonance
`iq` = `1` to activate Quasi-deuteron
`ig` = `1` to activate Giant Resonance

and each is = `0.0` otherwise
Default = `0.0` (no photonuclear interaction)

WHAT(2), WHAT(3): not used

WHAT(4) = lower bound of the indices of materials where the indicated photonuclear interactions are activated
("*From material WHAT(4)...*")
Default = `3.0`

WHAT(5) = upper bound of the indices of materials where the indicated photonuclear interactions are activated
("*... to particle WHAT(5)...*")
Default = `WHAT(4)`

WHAT(6) = step length in assigning indices
("*... in steps of WHAT(6)*")
Default = `1.0`

SDUM : not used

Default (option PHOTONUC not given): photon interactions with nuclei are not simulated

Notes

1. Muon photonuclear interactions (via virtual photons) are not handled by PHOTONUC but by MUPHOTON (p. 156).
2. Because photonuclear cross-sections are much smaller than photon cross-sections for electromagnetic interactions with atoms and electrons, analogue simulations of photonuclear interactions are very inefficient. Generally, it is recommended to use PHOTONUC in combination with LAM–BIAS (p. 131) to increase artificially the frequency of photonuclear interactions. See Notes 9 and 10 to option LAM–BIAS for more details.

Example 1:

```
* Giant Resonance and Quasideuteron photonuclear interactions are requested
* in material 18. The photon hadronic interaction length is artificially
* shortened by a factor 0.02 in order to improve statistics
*...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+...
PHOTONUC      1100.0       0.0       0.0      18.0       0.0       0.0
LAM-BIAS         0.0      0.02      18.0       7.0       0.0       0.0
```

Example 2:

```
* Photonuclear interactions are requested at all energies in materials
* 3, 7, 11 and 15. The photon hadronic interaction length is shortened
* by a factor 0.025
*...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+...
PHOTONUC         1.0       0.0       0.0       3.0      15.0       4.0
LAM-BIAS         0.0     0.025       0.0       7.0       0.0       0.0
```

7.55 PHYSICS

Allows to override the standard FLUKA *defaults for physics processes.*

See also DPMJET, EMFCUT, EVENTYPE, POLARIZAti, RQMD, THRESHOLd

This codeword concerns the following physics processes:

1. SDUM = DECAYS: decays of $\pi^{\pm}$, $\mu^{\pm}$, $K^{\pm}$ ($e\nu_e$, $\mu\nu_\mu$, $K^{\pm}_{\mu 3}$ and $K^{\pm}_{e3}$ channels) and K_{long} ($K^0_{\mu 3}$ and K^0_{e3} channels)
2. SDUM = CHARMDECay: flag for charmed hadron transport
3. SDUM = COALESCEnce: flag to activate the coalescence mechanism
4. SDUM = DPMTHREShold: lower energy threshold(s) for DPMJET
5. SDUM = EM–DISSOciation: ion electromagnetic dissociation
6. SDUM = EVAPORATion: evaporation
7. SDUM = IONSPLITting: activates the superposition model, i.e., ion splitting into nucleons

For SDUM = CHARMDECay:

WHAT(1) : flag for charmed hadron decays
- $\leq$ `-1.0`: resets to default (decay at production, no transport)
- = `0.0`: ignored
- = `1.0`: charmed hadrons are transported

WHAT(2) – **WHAT(6)**: not used

For SDUM = COALESCEnce:

WHAT(1) : coalescence flag
- $\leq$ `0.0`: false (no coalescence, default)
- = `0.0`: ignored
- > `0.0`: true, coalescence is activated

WHAT(2) – **WHAT(6)**: reserved to developers' use

For SDUM = DECAYS:

WHAT(1) : flag for particle decay
- $\leq$ `-1.0`: resets to default
- = `0.0`: ignored
- = `1.0`: maximum accuracy, polarisation accounted for in μ decays and in $\pi/K \rightarrow \mu$–ν_μ(e–ν_e) decays
- = `2.0`: maximum accuracy, polarisation not accounted for
- = `3.0`: phase space-like decays

`100.0` $\leq$ **WHAT(1)** $<$ `200.0`: leptonic decays only are allowed (implemented only for τ's). `WHAT(1) - 100` has the same meaning as above.

`200.0` $\leq$ **WHAT(1)** $<$ `300.0`: hadronic decays only are allowed (implemented only for τ's). `WHAT(1) - 200` has the same meaning as above.

Default = `1.0` (maximum accuracy and polarisation for both hadronic and leptonic decays)

WHAT(2), WHAT(3): not used

WHAT(4) = lower bound of the particle id-numbers to which the decay flag chosen by WHAT(1) applies ("*From particle WHAT(4)...*")
Default = 1.0

WHAT(5) = upper bound of the particle id-numbers to which the decay flag chosen by WHAT(1) applies ("*... to particle WHAT(5)...*")
Default = WHAT(4)

WHAT(6) = step length in assigning numbers ("*... in steps of WHAT(6)*")
Default = 1.0

For SDUM = DPMTHREShold:

WHAT(1) : minimum DPMJET kinetic energy for hadrons (GeV)
≤ 0.0: ignored

WHAT(2) : minimum DPMJET kinetic energy for ions (GeV/n)
≤ 0.0: ignored

WHAT(3) : minimum RQMD kinetic energy for ions (GeV/n)
≤ 0.0: ignored

WHAT(4) : smearing (±ΔE, GeV/n) for the RQMD-DPMJET switch energy
< 0.0: resets to default (0.0)

WHAT(5) : smearing (±ΔE, GeV/n) for the FLUKA-DPMJET switch energy for hA interactions
< 0.0: resets to default (0.0)

WHAT(6) : flag for restricting DPMJET h-A interactions to primary particles only
≤ -1.0: resets to default (false)
≤ 0.0: ignored
> 0.0: sets to true

Default (no PHYSICS option with SDUM = DPMTHREShold): DPMJET is not called for h-A interactions, but it is called for A-A interactions down to 5 GeV/n. RQMD is called between 5 and 0.1 GeV/n.
Warning: to activate interactions of primary and secondary heavy ions, the EVENTYPE card *must* be issued with SDUM = DPMJET
Warning: the FLUKA executable must be built with the DPMJET and RQMD libraries to perform A-A interactions (see `ldpmqmd` script in `$FLUPRO/flutil`).

For SDUM = EM–DISSOciation:

WHAT(1) : flag for activating ion electromagnetic dissociation
≤ -1.0: resets to default (no em-dissociation)
= 0.0: ignored
= 1.0: (default) no em-dissociation
= 2.0: projectile and target em-dissociation activated
= 3.0: projectile only em-dissociation activated
= 4.0: target only em-dissociation activated

WHAT(2) – **WHAT(6)**: not used

For SDUM = EVAPORATion:

WHAT(1) : flag for FLUKA evaporation model
- $\leq$ -1.0: resets to default (new model, no heavy fragment evaporation)
- = 0.0: ignored
- = 1.0: old evaporation model (**OBSOLETE:** kept for developers' use only)
- = 2.0: new evaporation model, no heavy fragment evaporation
- = 3.0: new evaporation model, with heavy fragment evaporation (CPU expensive: see Note 1 below)

Default = 2.0 (new model,no heavy fragment evaporation)

WHAT(2) – **WHAT(6)**: not used

For SDUM = IONSPLITting:

WHAT(1) : flag for activating ion splitting into nucleons
- $<$ 0.0: false, no ion splitting (default)
- = 0.0: ignored
- $>$ 0.0: true: ion splitting is activated

WHAT(2) : minimum energy for ions (GeV/n) above which splitting into nucleons will be performed (default: 0.1 GeV/n)
- $\leq$ 0.0: ignored

WHAT(3) : maximum energy for ions (GeV/n) below which splitting into nucleons will be performed (default: 5 GeV/n)
- $\leq$ 0.0: ignored

WHAT(4) – **WHAT(6)**: not used

Default (option PHYSICS not given): standard FLUKA treatment of physics processes

Note

1. In order to achieve accurate results for residual nuclei production the evaporation of heavy fragments *must* be activated. This, however, is not the default since it brings a heavy CPU burden, and is not needed for most applications.

Example:

```
* Only hadronic decays are allowed for tau+ and tau- (id-number 41 and 42)
*...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+...
PHYSICS        250.0       0.0       0.0      41.0      42.0       0. DECAYS
* Maximum accuracy requested for decay of pi+ and pi-(id-number 13 and 14),
* but without accounting for polarisation
* Phase space
PHYSICS          2.0       0.0       0.0      13.0      14.0       0. DECAYS
* New evaporation model requested
PHYSICS          2.0       0.0       0.0       0.0       0.0       0. EVAPORAT
```

7.56 PLOTGEOM

Calls the PLOTGEOM *geometry plotting package [89], to scan slices of the problem geometry and to produce auxiliary files for plotting*

See also GEOBEGIN, GEOEND

WHAT(1) = `0.0`: axes are plotted
≠ `0.0`: no axes

WHAT(2) = `0.0`: all region boundaries are plotted
≠ `0.0`: only boundaries between different materials are plotted

WHAT(3) = `0.0`: no numbering of regions or boundaries
≠ `0.0`: boundaries and regions are numbered **(not yet implemented)**

WHAT(4) > `0.0`: maximum length of each worm (see Note 1)
< `0.0`: worm compression is performed
Default = `2000.0`

WHAT(5) = `0.0`: no diagnostic printing
≠ `0.0`: write the scan history to logical output unit WHAT(6) (the filename will be `PLOTGEOM.OUT`)

WHAT(6) = number of the logical unit for PLOTGEOM input. If different from `0.0` or `5.0`, the PLOTGEOM input must be provided in a file `PLG.GFXINDAT`. PLOTGEOM input, which is not in standard FLUKA format, is described in Note 4.
Default = `LUNIN (= 5.0)` (i.e., PLOTGEOM input immediately follows)

SDUM = FORMAT: the PLOTGEOM store file will be a formatted one
= anything else: unformatted store file

Default (option PLOTGEOM not given): no plotting

Notes

1. The PLOTGEOM codeword links to FLUKA the PLOTGEOM program, which was written by R. Jaarsma and H. Rief of the Ispra Joint Nuclear Research Centre, and was adapted as a stand-alone program for the FLUKA Combinatorial Geometry by G.R. Stevenson of CERN. The present version, integrated with the dynamically allocated storage of the FLUKA code as an input option, has been improved from several points of view, mainly higher flexibility and smaller space requirements.
   The following documentation is extracted with some modifications from the original EURATOM report of Jaarsma and Rief [89].
   PLOTGEOM is a program for checking the geometry input data of Monte Carlo particle transport codes. From the points of intersection between the geometrical structure and a mesh of lines of flight, PLOTGEOM generates a picture, representing a cross-section of the geometry.
   The user specifies a two-dimensional cross-section of the geometry by giving its orientation with respect to the geometry coordinate axes and the desired coordinates of the corners of the picture (note that the x-y coordinate system of the "picture" is usually different from the one to which the geometry description refers).
   The program generates a horizontal grid of lines (parallel to the x-axis *of the picture*), covering the area of the desired picture. The constant distance between adjacent lines is 0.07 cm in the picture. The points of intersection with the medium boundaries are recorded. After having scanned one line, each intersection point $P2_j$ found is compared with each similar point $P1_k$ found on the line immediately preceding. If the distance between a couple of points $P1_k$, $P2_j$ is ≤ 0.035 cm, then the linepiece $P1_k$–$P2_j$ is called a segment. If more

than one of the points P2 on the current line satisfies the quoted condition for a given P1, then only the nearest one to that P1 is taken.
Now we define a "worm body" as being one segment or a string of segments, so that the endpoint of one segment is the begin point of the next segment.
If a worm body with a last point $P1_j$ already exists, the segment $P1_j$–$P2_k$ is connected to this worm body and the last point of that worm body becomes $P2_k$. Otherwise, the segment $P1_j$–$P2_k$ is the first one of a new worm body and the program looks for a "worm head" to be connected to $P1_j$. This "head" has to be in the neighbourhood of $P1_j$ between the two last scanned lines and is found by the subroutine `HEADTL` (`PGMSCN` in the FLUKA version), which applies the same principle for finding segments, but on a refined and variable grid.
If there is a worm body with a last point $P1_j$ and if on examining all P2 points no segment $P1_j$–$P2_k$ is found, then this body should be given a "tail". This tail is determined by the subroutine `HEADTL` (`PGMSCN`) in the same way as a head.
The "worms" (head, body, tail) thus created are stored on disk.
If the horizontal scanning has been finished the same procedure is repeated in the vertical direction (parallel to the y-axis of the picture).
Finally the worms are concatenated as far as possible: head to tail, head to head, tail to tail. The strings of worms formed in this way are plotted by means of any available graphics program (e.g., PAW).

2. A PLOTGEOM card can be issued only after the combinatorial geometry input has been read (either from an external file or from standard input). In other words, PLOTGEOM cannot be input before the GEOEND card. In addition, if WHAT(2) is different from `0.0`, PLOTGEOM can be invoked only after materials have already been assigned.
3. Since PLOTGEOM now makes use of the same dynamically allocated blank common storage of FLUKA it is convenient to issue the PLOTGEOM card just after geometry and material definitions, but before biasing and any other option which makes use of permanent and/or temporary storage in blank common. The purpose is twofold:
   (a) this maximises the storage available for PLOTGEOM for a given blank common dimension, and hence minimises the chances of having a too small storage
   (b) since PLOTGEOM frees again all the used storage after completion, the total memory required for the blank common is minimised
4. The input data required by PLOTGEOM to perform a slice scan must be given on the unit specified by WHAT(6) as follows:
   First line (format `A80`) : scan title
   Second line (format `6E10.5`): `X0, Y0, Z0, X1, Y1, Z1`
   Third line (format `6E10.5`): `TYX, TYY, TYZ, TXX, TXY, TXZ`
   Fourth line (format `3E10.5`): `EXPANY, EXPANX, DUMMY`
   The meaning of the variables is:
   `X0, Y0, Z0` = real coordinates of the bottom left-hand corner of the picture
   `X1, Y1, Z1` = real coordinates of the top right-hand corner of the picture
   `TYX, TYY, TYZ` = direction cosines of the y-axis of the plot
   `TXX, TXY, TXZ` = direction cosines of the x-axis of the plot
   `EXPANY, EXPANX` = expansion factors $\geq$ `0.1` for the y-axis, resp. the x-axis
   `DUMMY`: not used at present
   There is some redundancy in the position and direction input: indeed once `X0,Y0,Z0,X1,Y1,Z1` are given, only one of the axis is actually required. Therefore the three cosines of one of the two axes can be left = `0.0`.
   If $<$ `0.1`, `EXPANX, EXPANY` are reset to the default = `1.0`. Only their relative value matters.
5. The scan output file is written (formatted or not according to the value of SDUM) on unit `LUNPGS` (= `4`) with the default name of `PLOTGEOM.STORE`.
   The formatted version is self explanatory, while the unformatted one is organised as follows:
   1st record: one `CHARACTER*80` variable (title of the scan)
   2nd record: `14 REAL*4` variables: `X0,Y0,Z0, X1,Y1,Z1, TYX,TYY,TYZ, TXX,TXY,TXZ, XAXLEN,YAXLEN`
   where:
   `X0,Y0,Z0` and `X1,Y1,Z1` = coordinates of the bottom and top corners (from 2nd input line above)
   `TYX,TYY,TYZ` and `TXX,TXY,TXZ` = direction cosines of the axes (from 3rd input line above)
   `XAXLEN` and `YAXLEN` = lengths of x and y axes
   Then, repeated `M` times (`M` $\geq$ `2`, typically `M` = `2`):
   M_1 record: two `I*4` variables, representing:
   (a) the number of worms `NWORMS`
   (b) a dummy variable

Then, repeated `I = 1, NWORMS` times:

MI_1 record: three `I*4` variables `I, IDUM, LENGTH`
where the first one is the worm index, the second is dummy, the third is the worm `LENGTH`, namely the number of points in the worm

MI_2 record: `(X(J),Y(J),J=1,LENGTH)`
where `X` and `Y` are the abscissae and ordinates of a vector of `LENGTH` points to be joined with a line (= a "worm") in the plane where the origin (`0,0`) corresponds to `X0,Y0,Z0` and the x and y axis to `TXX,TXY,TXZ` and `TYX,TYY,TYZ`

In a compressed file there is just an extra record at the very beginning of the file: it contains a `CHARACTER*80` string equal to '`***COMPRESSED***COMPRESSED***`'

Example:

```
* Plot a vertical section of a geometry where x-axis points up, y-axis points
* to the right, and z-axis into the page. The PLOTGEOM file will be formatted.
*...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+...
PLOTGEOM         1.0       1.0       0.0       0.0       0.0       5. FORMAT
          Vertical section of the tunnel geometry at z = 35 m
    -120.0    -180.0    3500.0     120.0     180.0    3500.0
       1.0       0.0       0.0       0.0       1.0       0.0
       1.0       1.0       0.0
```

7.57 POLARIZAti

Defines the polarisation of a photon beam or source and activates transport of polarised photons.

|WHAT(1)| $\leq$ `1.0`: x-axis cosine of the beam polarisation vector (electric vector in case of photons)
$>$ `1.0`: resets the default (no polarisation)
This value can be overridden in user routine `SOURCE` (p. 314) by assigning a value to variable `UBMPOL`
Default = `-2.0` (no polarisation)

WHAT(2) = y-axis cosine of the beam polarisation vector
This value can be overridden in user routine `SOURCE` (p. 314) by assigning a value to variable `VBMPOL`
Default = `0.0`

WHAT(3) = z-axis cosine of the beam polarisation vector
This value can be overridden in user routine `SOURCE` (p. 314) by assigning a value to variable `WBMPOL`
Default = `0.0`

WHAT(4) : flag for relative direction of beam and polarisation
$\geq$ `1.0`: the polarisation is orthogonal to the direction of the primary photons
$<$ `1.0`: resets the default (the polarisation is not orthogonal to the direction of the primaries)
This value can be overridden in user routine `SOURCE` (p. 314) by assigning a value to the logical variable `LPPERP`
Default = `0.0` (the polarisation is not orthogonal to direction of the primaries)

WHAT(5) = polarisation fraction (see explanation in WHAT(6) below)
$<$ `0.0`: resets to default = 1.0
$>$ `1.0`: resets to default = 1.0
This value can be overridden in user routine `SOURCE` (p. 314) by assigning a value to variable `POLFRA`
Default = `1.0` (fully polarised in the direction described by WHAT(1)–WHAT(3))

WHAT(6) : flag for interpreting WHAT(5):
$\leq$ `0.0`: a fraction `WHAT(5)` of beam particles are linearly polarised in the direction described by WHAT(1)–(3) and the remaining fraction (`1.0 - WHAT(5)`) are not polarised
$\geq$ `1.0`: a fraction `WHAT(5)` of beam particles are linearly polarised in the direction described by WHAT(1)–(3) and the remaining fraction (`1.0 - WHAT(5)`) are polarised in the direction orthogonal to both the beam and that described by WHAT(1)–(3)
This value can be overridden in user routine `SOURCE` (p. 314) by assigning a value to the logical variable `LPFRAC`
Default = `0.0` (only a fraction `WHAT(5)` of the photons is polarised as indicated by WHAT(1)–WHAT(3), and the remaining fraction is not polarised)

SDUM : not used

Default (option POLARIZAti not given): photons are not assumed to be polarised

Notes

1. Polarisation direction defined by option POLARIZAti is meaningful only if the beam direction is along the positive z-axis, unless a command BEAMAXES is issued to establish a beam reference frame different from the geometry frame (see p. 7.4).

2. The program takes care of properly normalising the cosines unless they are badly unnormalised (in the latter case the code would reset to no polarisation). If WHAT(4) $\geq$ 1.0, the code makes sure that the two vectors are orthogonal within the minimum possible rounding errors.
3. What polarisation means is dependent on the physics implemented in the code: for the moment the only polarisation dependent effects are Compton, Rayleigh and photoelectric effect for photons, where of course the polarisation vector represents the electric field direction and must be normal to the beam direction.

Example:

```
* Synchrotron radiation beam with m_e/E mrad x,y divergence (produced by a 3 GeV
* electron beam). The actual spectrum is provided by a a user-written source
* (E_max = 500 keV). Photons are fully polarised in the horizontal (y) direction
* and the polarisation is orthogonal to the direction of the primary photons
*...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
DEFAULTS                                                              EM-CASCA
BEAM        -500.E-6       0.0 1.7033E-4       0.0       0.0       1.0PHOTON
SOURCE           0.0       0.0       0.0       0.0       0.0       0.0
POLARIZA         0.0       1.0       0.0       1.0       1.0       0.0
```

7.58 RADDECAY

requests simulation of radioactive decays and sets the corresponding biasing and transport conditions

See also DCYTIMES, DCYSCORE, IRRPROFIle, RESNUCLEi

WHAT(1) : flag for activating radioactive decays

= `1.0`: radioactive decays activated for requested cooling times

> `1.0`: radioactive decays activated in semi-analogue mode

= `0.0`: ignored

< `0.0`: reset to default

Default : no radioactive decays

WHAT(2) : flag for "patching" isomer production, while waiting for a better production model

> `0.0`: isomer production "patching" activated

< `0.0`: isomer production "patching" disabled

= `0.0`: ignored

Default : activated if non-analogue radioactive decays are requested, disabled otherwise

WHAT(3) : number of "replicas" of the decay of each individual residual

= `0.0`: ignored

< `0.0`: reset to default

Default = `1.0` for analogue decays, `3.0` otherwise

WHAT(4) switch for applying various biasing features only to prompt particles, or only to particles originated in radioactive decays, or to both.

> `0.0`: a 9-digit number `abcdefghi`, where each `a-i` digit is interpreted as follows (but see Note 4 below):

`0.0` = ignored

`1.0` = the corresponding biasing is applied to prompt radiation only

`2.0` = applied to decay radiation only

`3.0` = applied to both prompt and decay radiation

and the digit position is interpreted as follows:

`a` = hadron/muon interaction length or decay biasing, as defined by command LAM–BIAS

`b` = hadron/muon leading particle biasing (not defined at the moment)

`c` = hadron/muon importance and Weight Window biasing, as defined by commands BIASING and WW–FACTOr

`d` = $e^{\pm}/\gamma$ interaction length biasing, as defined by command EMF–BIAS

`e` = $e^{\pm}/\gamma$ leading particle biasing, as defined by command EMF–BIAS

`f` = $e^{\pm}/\gamma$ importance and Weight Window biasing, as defined by commands BIASING and WW–FACTOr

`g` = low-energy neutron biased downscattering, as defined by command LOW–DOWN, and non-analogue absorption, as defined by LOW–BIAS

`h` = no meaning for the time being

`i` = low-energy neutron importance and Weight Window biasing, as defined by commands BIASING, WW–FACTOr and WW–PROFIle

= `0.0`: ignored

< `0.0`: reset to default

Default : all biasing is applied to prompt showers only (equivalent to `111111111.`)

WHAT(5) : multiplication factors to be applied to $e^{\pm}/\gamma$ transport energy cut-offs, respectively for prompt and decay radiation

$>$ `0.0`: a 10-digit number `xxxxxyyyyy`, where the first and the last 5 digits are interpreted as follows (see Note 5 below):

`xxxxx` $\times$ `0.1` = transport energy cut-off multiplication factor for $\beta^{\pm}$ and γ decay radiation

`yyyyy` $\times$ `0.1` = transport energy cut-off multiplication factor for prompt $e^{\pm}$ and γ radiation

$=$ `0.0`: ignored

$<$ `0.0`: reset to default

Default : $e^{\pm}$ and γ transport energy cut-offs are unchanged: the multiplication factors are set = `1.0` for both prompt and decay radiation (equivalent to `0001000010.`)

WHAT(6), SDUM: not used

Default (option RADDECAY not given): no radioactive decay is activated, and no multiplication factors are applied to transport energy cut-offs

Notes

1. FLUKA allows for two different ways of simulating radioactive decay. In the *semi-analogue* mode, (WHAT(1) >1) each single radioactive nucleus is treated in a Monte Carlo way like all other unstable particles: a random decay time, random daughters, random radiation are selected and tracked. This allows for event-by-event analysis, with the time structure recorded in the particles age variable. It is called semi-analogue because the radiation spectra are inclusive (i.e., no correlated γ cascade is reproduced, etc.).
   In the "activation study" mode (WHAT(1) = 1) the time evolution is calculated analytically and all daughter nuclei and all associated radiation are considered, but at fixed times. (See Note 6 below).
   In both cases, the emitted particles are transported like all other secondaries, within the same run.
2. In the analytical evolution, each radioactive nucleus can be "decayed" several times, in order to improve statistics on, for instance, energy deposition, as set by WHAT(3).
3. Although FLUKA allows to simulate in a same run the transport of cascade particles and that of particles generated by decay of the produced residual nuclei, transport and biasing need in general to be set at very different levels. For instance, in a study of induced activity due to photonuclear reactions, it is recommended to set the photon transport threshold not lower than the photonuclear reaction threshold. However, gammas produced in the decay of those residual nuclei have in general lower energies and need to be transported with much lower energy cut-offs (see Note 5 below).
4. Biasing can be applied to radiation products. At present, for the biasing switch represented by WHAT(4), only the `d`, `e` and `f` choices are relevant since only $\beta^{\pm}$ and γ decays are considered for the time being.
5. Both multiplication factors imbedded in WHAT(5) must be $\geq$ `1.0`. If any of the multiplication factors is set to a value larger than `9999.0`, it is effectively considered as ∞, i.e., WHAT(5) = `0000099999.` will kill the electromagnetic cascade in the prompt part, while leaving it untouched in the decay part.
   WHAT(5) = `9999900000.` will do the opposite.
6. It is possible to perform on-line time evolution of decay radiation, and to score all standard quantities (energy deposition, residuals...) according to a user-defined irradiation profile (IRRPROFIle command, see p. 130) and one or more user-defined decay times (DCYTIMES command, p. 82). Radiation transport will be performed only once, and the evolution will be applied as a weight depending on the setting of the estimator, to be defined with the DCYSCORE command (p. 81).

Example:

```
* In this example, radioactive decays are activated for requested cooling
* times, with an approximated isomer production. Each radioactive nucleus
* produced will be duplicated.
*...+....1....+....2....+....3....+....4....+....5....+....6....+....7...
RADDECAY         1.0       1.0       2.0   111000.      200.
* Any biasing of electrons, positrons and photons is applied only to
* prompt particles in the electromagnetic shower, and not to beta and
* gamma particles from radioactive decay.
* The transport energy cut-offs set by EMFCUT (or by DEFAULTS) are
* applied as such to decay betas and gammas, but are multiplied by a
* factor 20 when applied to prompt particles.
```

7.59 RANDOMIZe

Sets the seeds for the double-precision random number generator RM64

WHAT(1) : logical file unit from which to read the seeds.
Default = `1.0` (reads the random number seeds from unit 1)

WHAT(2) = any number $< 2 \cdot 10^9$ (see Note 5).
Default = `1234598765`

WHAT(3) – **WHAT(6), SDUM**: not used

Default (option RANDOMIZe not given): standard seeds are used as implemented

Notes

1. The random number generator can be now initialised in one way only, namely to read from an external file (generated in a previous run) a vector of 97 seeds (the file contains also some auxiliary information). If that file is missing or empty or anyway invalid, the code will initialise the random number generator in its default state.
2. While the number of calls to the random number generator are printed on the standard output at the end of each primary history (or group of histories — see WHAT(5) of option START, p. 199), the 97 seeds are printed at the same time on a separate file. Skipping calls is therefore unnecessary in order to re-start a run at a particular point (e.g., where it stopped because of a crash). However, it still possible to skip a given number of calls by running the random number generator stand-alone.
3. It is *mandatory* to use only seeds output information as written by the program in earlier runs *on the same computer platform*. Otherwise the randomness of the number sequence would not be guaranteed.
4. FLRN64 is a portable random number generator in double precision, written in Fortran by P. Sala. It is based on an a new algorithm by Marsaglia and Tsang [102]. It gives random floating point numbers in the interval [0,1), with 53 significant bits of mantissa.
5. Different numbers input as WHAT(2) will initialise different and independent random number sequences, allowing the user to run several jobs in parallel for the same problem.
   The default is 1234598765.

Example 1:

```
* The seeds for the random number generator will be read from the file connected
* with logical unit 1
*...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
RANDOMIZE        1.0       0.0       0.0       0.0       0.0       0.0       0.0
```

Example 2:

```
* This run will be completely independent statistically from the previous one
*...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
RANDOMIZE        1.0  4042731.       0.0       0.0       0.0       0.0       0.0
```

7.60 RESNUCLEi

Scores residual nuclei produced in inelastic interactions on a region basis.

See also DCYSCORE, DCYTIMES, IRRPROFIle, RADDECAY)

WHAT(1) : type of products to be scored
- = 1.0: spallation products (all inelastic interactions except those induced by neutrons below the threshold for multigroup treatment)
- = 2.0: low-energy neutron products, i.e., those produced by neutrons below the threshold for multigroup treatment (provided the information is available, see Note 1).
- = 3.0: all residual nuclei are scored (if available, see above)
- = 0.0: resets the default (= 1.0)

Default = 1.0 (only spallation products are scored)

WHAT(2) = logical output unit
- > 0.0: formatted data are written on the WHAT(2) unit
- < 0.0: unformatted data are written on the |WHAT(2)| unit
  Values of |WHAT(2)| < 21.0 should be avoided
  (with the exception of WHAT(2) = +11.0 = standard output).
- = 0.0: resets the default (= 11.0)

Default = 11.0 (standard output unit)

WHAT(3) = maximum atomic number Z of the residual nuclei distribution
Default : according to the Z of the element(s) of the material assigned to the scoring region

WHAT(4) = maximum $M = N - Z - (NMZ)_{min}$ of the residual nuclei distribution (see Note 2)
Default : according to the mass and atomic number A, Z of the element(s) of the material assigned to the scoring region

WHAT(5) = scoring region number
Default = 1.0

WHAT(6) = volume of the region in cm^3
Default = 1.0

SDUM = any character string identifying the detector (max. 10 characters)

Notes

1. Elements or isotopes for which the FLUKA low-energy neutron cross-sections contain information on the production of residual nuclei are indicated by "Yes" in column 5 ("Residual nuclei") of Table 10.3 (p. 295) where the components of the neutron cross-section library are listed.
   The same information can be obtained by setting the printing flag in the LOW–NEUT option (WHAT(4) > 0.0). If such data are available for a given nuclide, the following message is printed on standard output:
   (RESIDUAL NUCLEI INFORMATIONS AVAILABLE)
2. To minimise storage, nuclei are indexed by Z (with $Z_{min} = 1$) and $NMZ = N - Z$ (with $(NMZ)_{min} = -5$). The parameter M is defined as $M = NMZ - (NMZ)_{min}$: therefore $M_{min} = 1$. The following relations can also be useful:
   $N - Z = M + (NMZ)_{min}$ $\qquad$ $N = M + Z + (NMZ)_{min}$
3. In order to achieve reasonable results for residual nuclei production the new evaporation module (see option PHYSICS, p. 176) *must* be activated. The old evaporation is still the default, mostly because of historical reasons and speed, but it does not produce meaningful results for residuals. The new evaporation, available since 1997, is far more sophisticated in this respect, while differences in emitted particle spectra and multiplicities are weak.
4. Protons/neutrons are never scored, ^{2}H, ^{3}H, ^{3}He, ^{4}He are scored at the end of their path, if transported (see option EVENTYPE, p. 116).
5. All residual nuclei are scored when they have been fully de-excited down to their ground or isomeric state. The scoring does not distinguish between ground state and isomeric state: they are scored as the same isotope.

6. Radioactive decay of the produced nuclei can be performed by FLUKA in the same run: see commands DCYTIMES (7.12), DCYSCORE (7.11), IRRPROFIle (7.36) and RADDECAY (7.58).
7. An example on how to read RESNUCLEi unformatted output is shown below. An explanation of the meaning of the different variables is given in the comments at the beginning of the program. The program lists the `Z` and `A` of the produced nuclei, followed by the corresponding amount per unit volume.
   A more complex program `USRSUW`, which allows to compute also standard deviations over several runs, is available with the normal FLUKA code distribution in directory `$FLUPRO/flutil`. A special version of the same program, `USRSUWEV`, provides in addition a calculation of induced activity and of its evolution in time. But the new available FLUKA feature, which allows to do the same kind of calculation on-line in the same run (see Note 6 above), is to be preferred because it is more accurate and allows to estimate statistical errors.

```
      PROGRAM RDRESN
*---------------------------------------------------------------------*
*     Up to MXRSNC user defined track or coll are allowed             *
*            izrhgh = maximum Z of the scoring (minimum Z: 1)         *
*            imrhgh = maximum M=N-Z-NMZ_min of the scoring            *
*                     (minimum M: 1). Note:                           *
*                      N-Z = M + NMZ_min, N = M + Z + NMZ_min         *
*            itursn = type of binning: 1 = spallation products,       *
*                     2 = low-energy neutrons products,               *
*                     3 = all products                                *
*            nrursn = region                                          *
*            vursnc = volume (cm**3) of the detector                  *
*            tiursn = scoring name                                    *
*---------------------------------------------------------------------*
      PARAMETER ( MXRSNC = 400 )
      CHARACTER*10 TIURSN
      CHARACTER RUNTIT*80, RUNTIM*32, FILNAM*80

      DIMENSION TIURSN(MXRSNC), ITURSN(MXRSNC), NRURSN(MXRSNC),
     &          VURSNC(MXRSNC), IMRHGH(MXRSNC), IZRHGH(MXRSNC)
      DIMENSION RNDATA(MXRSNC,100,260)

      WRITE(*,*)' Type the name of the input file:'
      READ (*,'(A)') FILNAM
      LQ = INDEX(FILNAM,' ') - 1
      OPEN (UNIT=1, FILE=FILNAM, STATUS='OLD', FORM='UNFORMATTED')
      OPEN (UNIT=2, FILE=FILNAM(1:LQ)//'.txt', STATUS='UNKNOWN')
*----------- read and write 1st record ---------------------------------
      READ (1) RUNTIT, RUNTIM, WEIPRI, NCASE
      WRITE(2,100) RUNTIT, RUNTIM, NCASE, WEIPRI
*------- loop on residual nuclei detector data in the present file -----
      DO 1 IRN = 1, MXRSNC
         READ (1, END=1000) NRN, TIURSN(NRN), ITURSN(NRN),
     &   NRURSN(NRN), VURSNC(NRN), IMRHGH(NRN), IZRHGH(NRN), K
         IF (ABS(ITURSN(NRN)) .LE. 1) THEN
            WRITE (2,200) NRN, TIURSN(NRN), NRURSN(NRN),
     &        VURSNC(NRN), IZRHGH(NRN), IMRHGH(NRN) + K, K + 1
         ELSE IF (ABS(ITURSN(NRN)) .LE. 2) THEN
            WRITE (2,300) NRN, TIURSN(NRN), NRURSN(NRN),
     &        VURSNC(NRN), IZRHGH(NRN), IMRHGH(NRN) + K, K + 1
         ELSE
            WRITE (2,400) NRN, TIURSN(NRN), NRURSN(NRN),
     &        VURSNC(NRN), IZRHGH(NRN), IMRHGH(NRN) + K, K + 1
         END IF
         WRITE(2,'(/,A)') '   Z   A   Residual nuclei'
         WRITE(2,'(A,/)') '         per cm**3 per primary'
         READ (1) ((RNDATA(NRN,I,J), I=1,IZRHGH(NRN)), J=1,IMRHGH(NRN))
         DO 2 I = 1, IZRHGH(NRN)
            DO 3 J = 1, IMRHGH(NRN)
               IF(RNDATA(NRN,I,J) .GT. 0.)
     &         WRITE(2,'(2I4,1P, G15.6)') I, J+K+2*I, RNDATA(NRN,I,J)
```

```
 3         CONTINUE
 2      CONTINUE
 1   CONTINUE
1000 CONTINUE
100  FORMAT(/,1X,'*****',2X,A80,2X,'*****',/,/,10X,A32,/,/,
    &        10X,'Total number of particles followed ',I9,', for a ',
    &        'total weight of ',1P,E15.8,/)
200  FORMAT (/,3X,'Res. nuclei n. ',I3,'  "',A10,
    &        '" , "high" energy products, region n. ',I5,
    &        /,6X,'detector volume: ',1P,E11.4,' cm**3',/
    &          6X,'Max. Z: ',I3,', Max. N-Z: ',I3,' Min. N-Z:',I3)
300  FORMAT (/,3X,'Res. nuclei n. ',I3,'  "',A10,
    &        '" , "low"  energy products, region n. ',I5,
    &        /,6X,'detector volume: ',1P,E11.4,' cm**3',/
    &          6X,'Max. Z: ',I3,', Max. N-Z: ',I3,' Min. N-Z:',I3)
400  FORMAT (/,3X,'Res. nuclei n. ',I3,'  "',A10,
    &        '" ,  all         products, region n. ',I5,
    &        /,6X,'detector volume: ',1P,E11.4,' cm**3',/
    &          6X,'Max. Z: ',I3,', Max. N-Z: ',I3,' Min. N-Z:',I3)
     END
```

Example:

```
* Calculate residual nuclei produced in an iron slab (region 6) and in a zinc
* vessel (region 10). Heavy recoils are transported (option EVENTYPE) and scored
* at the point where they stop. The new evaporation model is activated to ensure
* a better quality of the results. For iron, all residual nuclei are scored. For
* zinc, no data are available for low-energy neutrons, so only nuclei produced
* by spallation/evaporation are scored. Results are written (formatted) on
* logical unit 22 and 23, respectively.
*...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
MATERIAL        26.0    55.847      7.87       11.       0.0       0. IRON
MATERIAL        30.0     65.39     7.133       12.       0.0       0. ZINC
ASSIGNMAT       11.0       6.0       9.0       0.0   !  Four Fe slabs
ASSIGNMAT       12.0      10.0       0.0       0.0   !  Zn vessel
EVENTYPE         0.0       0.0       1.0       0.0       0.0       0. EVAP
PHYSICS          2.0       0.0       0.0       0.0       0.0       0. EVAPORAT
RESNUCLEI        3.0      22.0       0.0       0.0       6.0       0. FirstFe
RESNUCLEI        1.0      23.0       0.0       0.0      10.0       0. Znvessel
```

7.61 ROT–DEFIni

Defines rotations and translations to be applied to binnings.

See also EVENTBIN, ROTPRBIN, USRBIN

WHAT(1) : assigns a transformation index and the corresponding rotation axis
≤ `0.0`: the card is ignored
> `0.0`: interpreted as `i+j*100`
where `i` = index of the rotation
`j` = 1 rotation with respect to x axis
= 2 rotation with respect to y axis
= 3 rotation with respect to z axis
(see Note 2)
Default = `0.0` (no transformation defined)

WHAT(2) = polar angle of the rotation (θ = 0 ... 180 degrees)
Default : no default

WHAT(3) = azimuthal angle of the rotation (ϕ = -180 ... 180 degrees)
Default : no default

WHAT(4) = X_{offset} for the translation
Default : no default

WHAT(5) = Y_{offset} for the translation
Default : no default

WHAT(6) = Z_{offset} for the translation
Default : no default

SDUM : not used

Default (option ROT–DEFIni not given): no transformation is defined

Notes

1. FLUKA binnings (spatial meshes independent of the problem geometry, designed to score average or event-by-event quantities) are generally defined as Cartesian structures parallel to the coordinate axes, or as cylindrical structures parallel to the z-axis. However, it is possible to define binnings with any arbitrary position and direction in space, by means of transformations described by commands ROT–DEFIni and ROTPRBIN. Command ROT–DEFIni defines rotations/translations to be applied to binnings (requested by the user by means of EVENTBIN (p. 111) or USRBIN) (p. 218). Each transformation defined by ROT–DEFIni is assigned a number WHAT(1) which can be applied to one or more binnings. The correspondence between transformation index and binning number is assigned via option ROTPRBIN (p. 192).
2. The transformation matrices are:
`j = 1`:

$$\begin{vmatrix} X_{new} \\ Y_{new} \\ Z_{new} \end{vmatrix} = \begin{vmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{vmatrix} \cdot \begin{vmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & \sin\phi \\ 0 & -\sin\phi & \cos\phi \end{vmatrix} \cdot \begin{vmatrix} X_{old} + X_{offset} \\ Y_{old} + Y_{offset} \\ Z_{old} + Z_{offset} \end{vmatrix}$$

`j = 2`:

$$\begin{vmatrix} X_{new} \\ Y_{new} \\ Z_{new} \end{vmatrix} = \begin{vmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & \sin\theta \\ 0 & -\sin\theta & \cos\theta \end{vmatrix} \cdot \begin{vmatrix} \cos\phi & 0 & -\sin\phi \\ 0 & 1 & 0 \\ \sin\phi & 0 & \cos\phi \end{vmatrix} \cdot \begin{vmatrix} X_{old} + X_{offset} \\ Y_{old} + Y_{offset} \\ Z_{old} + Z_{offset} \end{vmatrix}$$

`j = 3`:

$$\begin{vmatrix} X_{\text{new}} \\ Y_{\text{new}} \\ Z_{\text{new}} \end{vmatrix} = \begin{vmatrix} \cos\theta & 0 & -\sin\theta \\ 0 & 1 & 0 \\ \sin\theta & 0 & \cos\theta \end{vmatrix} \begin{vmatrix} \cos\phi & \sin\phi & 0 \\ -\sin\phi & \cos\phi & 0 \\ 0 & 0 & 1 \end{vmatrix} \begin{vmatrix} X_{\text{old}} + X_{\text{offset}} \\ Y_{\text{old}} + Y_{\text{offset}} \\ Z_{\text{old}} + Z_{\text{offset}} \end{vmatrix}$$

$R_{ij} = T_{ik}P_{kj}$:

$$\begin{vmatrix} X_{\text{new}} \\ Y_{\text{new}} \\ Z_{\text{new}} \end{vmatrix} = \begin{vmatrix} \cos\phi\times\cos\theta & \sin\phi\times\cos\theta & -\sin\theta \\ -\sin\phi & \cos\phi & 0 \\ \cos\phi\times\sin\theta & \sin\phi\times\sin\theta & \cos\theta \end{vmatrix} \begin{vmatrix} X_{\text{old}} + X_{\text{offset}} \\ Y_{\text{old}} + Y_{\text{offset}} \\ Z_{\text{old}} + Z_{\text{offset}} \end{vmatrix}$$

and the inverse $R^{-1}_{ij} = P^{-1}_{ik}T^{-1}_{kj}$:

$$\begin{vmatrix} X_{\text{old}} \\ Y_{\text{old}} \\ Z_{\text{old}} \end{vmatrix} = \begin{vmatrix} \cos\phi & -\sin\phi & 0 \\ \sin\phi & \cos\phi & 0 \\ 0 & 0 & 1 \end{vmatrix} \begin{vmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{vmatrix} \begin{vmatrix} X_{\text{new}} \\ Y_{\text{new}} \\ Z_{\text{new}} \end{vmatrix} - \begin{vmatrix} X_{\text{offset}} \\ Y_{\text{offset}} \\ Z_{\text{offset}} \end{vmatrix}$$

$$\begin{vmatrix} X_{\text{old}} \\ Y_{\text{old}} \\ Z_{\text{old}} \end{vmatrix} = \begin{vmatrix} \cos\phi\times\cos\theta & -\sin\phi & \cos\phi\times\sin\theta \\ \sin\phi\times\cos\theta & \cos\phi & \sin\phi\times\sin\theta \end{vmatrix} \begin{vmatrix} X_{\text{new}} \\ Y_{\text{new}} \\ Z_{\text{new}} \end{vmatrix} - \begin{vmatrix} X_{\text{offset}} \\ Y_{\text{offset}} \\ Z_{\text{offset}} \end{vmatrix}$$

3. For example (assume zero offset and $[x, y, z]$ = old frame, $[x', y', z']$ = new frame):

$\theta = \pi/2, \phi = 0$:

`j = 1`: $x' = -y$, $y' = x$, $z' = z$

`j = 2`: $x' = x$, $y' = -z$, $z' = y$

`j = 3`: $x' = z$, $y' = y$, $z' = -x$

$\theta = 0, \phi = \pi/2$:

`j = 1`: $x' = x$, $y' = z$, $z' = -y$

`j = 2`: $x' = -z$, $y' = y$, $z' = x$

`j = 3`: $x' = y$, $y' = -x$, $z' = z$

That is, the vector which has position angles θ and ϕ with respect to the `j`$_{th}$ axis in the original system, will become the `j`$_{th}$ axis in the rotated system. For the special case $\theta = 0$ this implies a rotation of $-\phi$ in the original frame. In practice it is more convenient to think about the inverse rotation, the one which takes the `j`$_{th}$ versor into the versor with θ and ϕ.

4. Note that a transformation can be defined recursively, for example with two cards pointing to the same transformation. If P_{ij} is the rotation corresponding to the first card and T_{ij} the one corresponding to the second card, the overall rotation will be $R_{ij} = T_{ik}P_{kj}$

Example:

```
*...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+...
ROT-DEFI       201.0       0.0      90.0    -100.0      80.0    -500.0
USRBIN          11.0     201.0      70.0      30.0       0.0    1000.0tot-dose
USRBIN           0.0       0.0       0.0      10.0       1.0       6.0&
ROTPRBIN        -1.0       1.0                 1.0       1.0       1.0
*  Here the transformation is applied to a cylindrical binning.
*  Track-lengths are scored in the binning with its axis
*  parallel to the x-axis of the coordinate frame:
*    Xmin = 100.0, Xmax = 1100.0
*    Rmin =   0.0, Rmax =   30.0
*    ( Y , Z ) coordinate of the binning axis = ( -80.0 , 500.0 )
```

7.62 ROTPRBIN

Sets the amount of storage and the storage precision (single or double) for binnings.
Sets also the correspondence between rotations/translations and binnings.

See also EVENTBIN, ROT–DEFIni

WHAT(1) : sets the run-time storage precision of binnings:
`-1.0` < WHAT(1) ≤ `0.0`: ignored
≤ `-1.0`: resets the storage precision to default (double)
= `xxz01`: sets the storage precision to single (see Note 2)
= `xxz00`: sets the storage to only a fraction of the required memory, that is only xx.z percent of the required memory for this binning is actually allocated (see Note 3)
Default : double precision storage and full memory allocation

WHAT(2) : associates a rotation/translation matrix
≤ `-1.0`: resets the associated rotation/translation to identity (transformation index = 0)
≥ `1.0`: associates the binning(s) indicated by WHAT(4)...WHAT(6) to the rotation/translation defined with number `NINT(WHAT(2))` by a ROT–DEFIni card (see Note 4)

WHAT(3) : not used

WHAT(4) = lower index bound of binnings in which the requested storage precision and/or transformation must be applied
("*From binning WHAT(4)...*")
Default = `1.0`

WHAT(5) = upper index bound of binnings in which the requested storage precision and/or transformation must be applied
("*...to binning WHAT(5)...*")
Default = WHAT(4)

WHAT(6) = step length in assigning indices.
("*...in steps of WHAT(6)*")
Default = `1.0`

SDUM : not used

Default (option ROTPRBIN not given): binning data are stored in double precision, and no rotation/translation is applied

Notes

1. Command ROTPRBIN can be used for three different tasks, all related to binnings requested by the user by means of EVENTBIN (p. 111) or USRBIN (p. 218):
   (a) to define the precision used at run-time to store the accumulated scores of selected binnings
   (b) to allocate a reduced amount of memory as a storage for selected binnings
   (c) to set the correspondence between the index of a transformation (rotation/translation as defined by command ROT–DEFIni, see p. 192) and the index of selected binnings.
2. The USRBIN/EVENTBIN output values are always in single precision, regardless of the run-time storage precision used. Run-time storage precision, which is double by default, should never be changed for binnings defined by USRBIN to prevent severe loss of data (adding small amounts in the rounding can result in values no

longer increasing with the number of primary particles). However, this is unlikely to happen with EVENTBIN binnings, which are reset at the end of each history.

3. In many cases, binnings defined by EVENTBIN result in a number of sparse "hit" cells, while all other bins are empty (scoring zero). In such cases, it is convenient to allocate less storage than required by the whole binning structure. See also Note 2 to option EVENTBIN, p. 111.
4. Binning space transformations (rotations and translations) are those defined by a ROT–DEFIni card. That is, the variables used for scoring are the primed one (x', y', z') (see Notes 2 and 3 to option ROT–DEFIni).

Example 1:

```
*...+....1....+....2....+....3....+....4....+....5....+....6....+....7...+...8
ROTPRBIN      85500.       0.0       0.0       2.0       6.0       2.0
*  Allocate only 85.5% of the memory normally required for binnings 2, 4 and 6
```

Example 2:

```
*...+....1....+....2....+....3....+....4....+....5....+....6....+....7...+...8
ROTPRBIN      10001.       0.0       0.0       3.0       5.0       0.0
*  Set single storage precision for binnings 3, 4 and 5
```

7.63 RQMD

Defines some I/O parameters relevant to the heavy ion event generators RQMD.

See also BME, DPMJET, EVENTYPE, MYRQMD, PHYSICS

Option RQMD is useful only for development work: generally the user does not need this option — to request activation of ion interactions use the command EVENTYPE (p. 116) with SDUM = DPMJET.

WHAT(1) – **WHAT(3)**: not used

WHAT(4) : flag to enable separate input

Default = 0.0 (no separate input)

WHAT(5) : unit for standard output from RQMD generator

Default = 0.0 (no output)

WHAT(6) : unit for extra output from RQMD generator

Default = 0.0 (no extra output)

SDUM : not used

Default (option RQMD not given): all the above defaults apply

7.64 SCORE

Defines the (generalised) particles producing the stars to be scored in each region.
Requests scoring of energy deposition in each region.

See also EVENTDAT, THRESHOLd, USRBIN

WHAT(1) ... WHAT(4): id-numbers of particles or generalised particles (see 5.1).

Depending on the (generalised) particle type, different quantities are scored *in each region*:

- For hadrons, photons, muons: stars (see Notes 3, 4a)
- For generalised particles `208.0` and `211.0`: energy deposition (Notes 4b, 4c).
- For generalised particles `219.0`, `220.0` and `221.0`: fissions (Note 4d).
- For generalised particle `222.0`, neutron balance (Note 4e).
- For generalised particles `229.0` and `230.0`: unbiased energy deposition (Note 4f).

= `0.0`: no scoring per region of any of the quantities listed above

Default = `201.0, 0.0, 0.0, 0.0` (score stars produced by all particles)

WHAT(5), WHAT(6), SDUM: not used

Default (option SCORE not given): no scoring by region

Notes

1. The possible particle numbers are those listed in 5.1, i.e., `-6.0` to `62.0` and `201.0` to `233.0`. However, not all particles will give meaningful results: for instance particles `3.0` and `4.0` (electrons and positrons) cannot produce stars, fissions, etc.
   Selecting generalised particles `208.0` (energy) or `211.0` ("electromagnetic" energy, i.e., energy of $e^+e^-$and γ), one can score deposited energy (proportional to dose).
2. SCORE is one of the oldest FLUKA commands, which has been kept unchanged because of its simplicity and easiness of use. On the other hand, because it lacks the flexible memory allocation of all other scoring options, there is presently room for only 4 types of particles. Therefore, only the 4 first valid WHAT-parameters are retained.
3. A star is a hadronic inelastic interaction occurring at an energy higher than a threshold defined via the option THRESHOLd (or by default higher than the transport threshold of the interacting particle). Star scoring, traditionally used in most high-energy shielding codes, can therefore be considered as a form of crude collision estimator: multiplication of star density by the asymptotic value of the inelastic nuclear interaction length gives the fluence of hadrons having energy higher than the current threshold. However, this is meaningful only if the interaction length doesn't vary appreciably with energy; therefore it is recommended to set a scoring threshold = 50 MeV (using option THRESHOLd), since interaction lengths are practically constant above this energy. Besides, star densities calculated with a 50 MeV threshold are the basis of some established radiation protection techniques such as the ω-factors for estimating material activation (see [169], p. 106), and the prediction of single isotope yields from the ratio of partial to inelastic cross-section.

4. The SCORE card defines the following scoring:
   (a) scoring by region the density of stars produced by the selected particles (if applicable, i.e., if the particles are hadrons — but not low-energy neutrons — photons or muons, or families of them). Stars produced by primary particles can be scored with id-number `210.0`, all stars with `201.0`.
   Results will be in stars/cm^3 per primary particle if region volumes have been input by setting `IVOPT = 3` in the geometry title card (see 8.2.2), otherwise in stars per region per primary.
   (b) scoring by region the total energy density, if generalised particle `208.0` has been selected.
   Results will be in GeV/cm^3 per primary if volumes have been input, otherwise in GeV per region per primary. To obtain dose in Gy, multiply GeV/cm^3 by $1.602176462 \times 10^{-7}/\rho$ (ρ = density in g/cm^3). This conversion can be done also at run-time by means of the user routine `COMSCW` (see 12.2.2).
   (c) scoring by region the energy density deposited by electrons, positrons and photons, if generalised particle `211.0` has been selected. Results as in the previous case.
   (d) scoring by region of fissions (or fission density), if generalised particles `219.0`, `220.0` or `221.0` have been selected. The three id-numbers refer to all fissions, high-energy fissions and low-energy neutron fissions respectively.
   Results are in fissions/cm^3 per primary if volumes are input, otherwise in number of fissions per primary.
   (e) scoring by region of the neutron balance (outgoing minus incoming neutrons), if generalised particle `222.0` has been selected.
   Results are in net number of neutrons per cm^3 per primary if volumes are input, otherwise in net neutrons per primary.
   (f) generalised particles `229.0` and `230.0` are similar respectively to `208.0` and `211.0`, with the difference that the energy deposited is scored with weight 1, independent of the actual weight of the particle. Of course, the results will have no physical meaning, but in some circumstances they will provide useful information about the run itself (for instance in order to optimise biasing).
   (g) scoring as in all cases above but separately for each primary event, if the EVENTDAT option is used (see p. 113).
5. A more flexible way to score by region stars, deposited energy etc. is "region binning" by means of option USRBIN (7.77, Note 8). However, note that in that case the results are not normalised per unit volume.
6. In FLUKA, stars *do not include* spallations due to annihilating particles.
7. SCORE *does not* define scoring done via USRBDX, USRBIN, USRCOLL and USRTRACK.

Example 1:

```
* Score stars produced in each region by protons, high-energy neutrons and pions.
* Score also total energy deposition in each region.
*...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
SCORE            1.0       8.0     209.0     208.0
```

Example 2:

```
* Score stars produced by primary particles (i.e., first interactions) in each
* region. Score also in each region stars produced by photons (photonuclear
* reactions) and energy deposited by electromagnetic showers.
*...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
SCORE          210.0       7.0     211.0
```

Example 3:

```
* Score fissions produced in each region by high- and low-energy particles.
* Score also the net neutron production in each region, and the kaon stars.
*...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
SCORE          220.0     221.0     222.0     215.0
```

7.65 SOURCE

Invokes the use of a user-defined source routine `SOURCE` *to sample the primary particles.*

See also BEAM, BEAMPOS, POLARIZAti, USRICALL

This option allows to input up to 12 double precision parameters and one character string chosen by the user, to be passed to the user routine `SOURCE`. To pass more than 6 parameters, two successive SOURCE cards are required.

First card:

WHAT(1) ... WHAT(6): user parameters to be passed to the `SOURCE` routine as a double precision subarray `WHASOU(1)...WHASOU(6)` (via `COMMON SOURCM`).

SDUM = any user dependent character string (not containing "`&`"), to be passed to the `SOURCE` routine as a character variable `SDUSOU` (via `COMMON CHEPSR`).

Continuation card (if present):

WHAT(1) ... WHAT(6): user parameters to be passed to the `SOURCE` routine as a double precision subarray `WHASOU(7)...WHASOU(12)`.

SDUM = "`&`" in any position in column 71 to 78 (or in the last field if free format is used)

Default (option SOURCE not given): subroutine `SOURCE` is not called

Notes

1. In many simple cases, the primary particle properties can be defined by just two input cards: BEAM (p. 64) and BEAMPOS (p. 69). The two options define the type of the particle, its energy/momentum (monoenergetic or simply distributed) and its starting position and direction (also sampled from a few simple distributions centred around the z-axis). A third option, POLARIZAti (p. 182), can be used to complete the description of the primaries in case they are totally or partially polarised.
   However, there are more complex situations where the type of primary particles and their phase space coordinates must be sampled from different types of distributions, read from files, or generated by a rule decided by the user. To handle these cases, it is possible to call a user routine `SOURCE` which can override totally or in part the definitions given in input. The call is activated by option SOURCE. A default version of the routine, which leaves any other input definition unchanged, is present in the FLUKA library. Instructions on how to write, compile and link `SOURCE` are given in 12.2.19.
2. Even when overridden by `SOURCE`, the momentum or kinetic energy defined by WHAT(1) of option BEAM is meaningful, since it is taken as maximum energy for several scoring facilities and for cross-section tabulations. Therefore, it is recommended to input in any case a BEAM card with the maximum energy expected in the current run.
3. The user has the possibility to write a flexible `SOURCE` routine which can handle different cases without being recompiled and linked. The 12 `WHASOU` optional double precision parameters and the `SDUSOU` character string can be combined to provide a multitude of possible options.
4. Initialisations (for instance reading data from files, or spectrum normalisation needed for sampling) can be done in `SOURCE` itself, as explained in 12.2.19, or in another user routine `USRINI`, which is called every time a card USRICALL (p. 229) is found in input (see 12.2.26).
   Note that more than one primary particle can be sampled in a same call to `SOURCE` and loaded into stack for later transport. A third user routine, `USREIN` (see 12.2.24), which is called just after `SOURCE` and before the sampled primary (or primaries) are transported, allows the user to do an initialisation at the beginning of each *event*. An event is defined as all the histories of primaries sampled in a single call to `SOURCE` and of their descendants. Routine `USREOU` is called instead at the end of each event (see 12.2.25).

5. In old versions of FLUKA, the call to SOURCE was requested by means of a flag in card START. This feature has been discontinued.

Example 1:

```
* A user-written SOURCE routine is called without passing any parameter.
*...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
SOURCE
```

Example 2:

```
* Here the user passes to the SOURCE routine 7 numerical values and one
* character string. These can be used as free parameters inside the routine,
* or as flags/switches to select between different options
*...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....8
SOURCE         12.58       1.0      -14.  0.987651      100.      365.FLAG18
SOURCE         999.2                                                   &
```

7.66 START

Defines the termination conditions, gets a primary from a beam or from a source and starts the transport.

See also RANDOMIZe, STOP

WHAT(1) = maximum number of primary histories simulated in the run
Default = 5000.0

WHAT(2) : not used

WHAT(3) = time left for termination and printout (in seconds)
Default = 80.0

WHAT(4) : not used

WHAT(5) = 0.0: a line reporting the number of calls to the random number generator (in hexadecimal form) is printed at the beginning of each history only for the first ones, and then with decreasing frequency

> 0.0: the number of calls is printed at the beginning of each history.

WHAT(6) = time reserved for an interactive run (in seconds)
Default = 100.0

SDUM : not used

Default (option START not given): the other input cards are read and an echo is printed on the standard output, but no actual simulation is performed. However two input cards, both related to geometry, have an effect even if no START card is present: GEOEND (p. 123) (with SDUM = DEBUG) and PLOTGEOM (p. 179). In all cases in which particles are transported, START must *always* be present.

Notes

1. The interactive time limit indicated by WHAT(6) can be used only on some systems which can provide a signal when the time limit is approaching. On personal workstations, generally no time limit is enforced.
2. It is also possible to terminate a FLUKA run before the pre-set time has expired or the total number of histories has been completed. To this effect, the user may create a "stopping file" (its content is irrelevant, since only its existence is checked by the program).
   When the program detects the existence of such a file, the time left is set to zero, the run is terminated at the end of the current history, and the stopping file itself is erased.
   The name of the stopping file is `FLUKA$STOP` on VAX, and *inputname*`.stop` on UNIX, where *inputname* is the name of the input file without the extension `.inp`. The name of the input file is provided by the user via the option COMMENT (p. 76) (SDUM = INPUT), otherwise it is assumed to be "`fluka`" by default.

Example:

```
*...+....1....+....2....+....3....+....4....+....5....+....6....+....7...+...8
START        70000.0
*  Request a run of 70000 primary particles
```

7.67 STEPSIZE

Sets the minimum and maximum step size on a region-by-region basis for transport of all charged particles (hadrons, muons and electrons)

See also EMFFIX, FLUKAFIX, MULSOPT, MGNFIELD

WHAT(1) $\geq$ `0.0`: minimum step size in cm (overrides region by region the overall minimum step set by WHAT(2) of MGNFIELD (*if larger than it*)
$<$ `0.0`: whatever happens, the location of boundary intersections will be found with an uncertainty $\leq$ |WHAT(1)|. Of course, for a given boundary, this value should be applied to both regions defining the boundary.
Default : no default

WHAT(2) = maximum step size in cm
Default = `100000.0` cm for a region without magnetic field
= `10.0` cm for a region with magnetic field
= twice the minimum step if the latter has been defined larger than the maximum

WHAT(3) = lower bound of the region indices in which the indicated step size is to be applied ("*From region WHAT(3)...*")
Default = `2.0`

WHAT(4) = upper bound of the region indices in which the indicated step size is to be applied ("*... to region WHAT(4)...*")
Default = WHAT(3)

WHAT(5) = step length in assigning indices ("*... in steps of WHAT(5)*")
Default = `1.0`

WHAT(6) and **SDUM**: not used

Default (option STEPSIZE not given): the above defaults apply in all regions (10 cm with magnetic field, 10^5 cm without)

Notes

1. This option differs from EMFFIX (p. 107) and FLUKAFIX (p. 119) for the following main reasons:
   (a) it is given by region rather than by material
   (b) therefore, it is effective also in vacuum regions
   (c) the maximum step is determined in an absolute way (i.e., in cm) rather than in terms of maximum energy loss
   (d) it allows to set not only a maximum but also a minimum step length. This may be necessary in order to avoid the forcing of extremely small steps when a low-energy charged particle spirals in a magnetic field. Option MGNFIELD (p. 151, Note 6) offers a similar possibility, but not tuned by region.
2. Option STEPSIZE may be essential in and around regions of very small dimensions or in regions where a magnetic field is present (and is rarely required otherwise).
3. The maximum step size for a given region can be decided from the following considerations:
   – in a region with magnetic field it should not be larger than the minimum dimension of the region itself and of its neighbour regions. Obviously, it should also be larger than the minimum step possibly imposed by MGNFIELD or by STEPSIZE itself.
   – in a non-vacuum region, it should not be larger than about one-third of its minimum dimension, in order to allow the multiple-scattering algorithm to work in optimal conditions.

Example:

```
*...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+...
ASSIGNMAT        2.0      15.0      30.0       5.0       1.0       0.0
* A magnetic field is present in vacuum regions 15, 20, 25 and 30.
MGNFIELD        20.0       0.2      0.10       0.0       0.0       0.0
STEPSIZE       -0.05       0.0      20.0      25.0       0.0       0.0
STEPSIZE         0.3       0.0      15.0       0.0       0.0       0.0
* The maximum deviation angle due to magnetic field in any step is set
* = 20 degrees, and boundary crossings must be identified with an error
* not larger than 2 mm. If the max. angle constraint forces the step to
* be shorter than 10 cm, the step will be set = 10 cm in region 20, 25, 30,
* but will be set = 3 mm in region 15 (WHAT(1) of STEPSIZE overrides the
* general setting due to WHAT(3) of MGNFIELD). Whatever the size of the
* step, however, the accuracy of the boundary crossing position must be
* equal or better than 0.5 mm in regions 20 and 25 (probably contiguous,
* since the same accuracy must be set for regions on either side of a
* boundary). The value of the magnetic field will be provided at each
* step by the user routine MAGFLD.
```

7.68 STERNHEIme

Allows to input Sternheimer density effect parameters

See also MAT–PROP

WHAT(1) = Sternheimer `-C` ($\bar{C}$) parameter

WHAT(2) = Sternheimer `X0` (X_0) parameter

WHAT(3) = Sternheimer `X1` (X_1) parameter

WHAT(4) = Sternheimer `AFACT` (a) parameter

WHAT(5) = Sternheimer `SK` (m) parameter

WHAT(6) = Sternheimer `DELTA0` (δ_0) parameter (only for single elements)

SDUM = index of the material to which the above Sternheimer parameters apply. Exceptionally, here SDUM must be an integer number, in free format, rather than a character string.

Default (option STERNHEImе not given): density effect parameters are computed according to the Sternheimer-Peierls general formula

Notes

1. For gases the parameters are supposed to be given at 1.0 atm (NTP); the code takes care to scale them to the actual pressure as given by the MAT–PROP card (p. 144).
2. MAT–PROP can be used also to override the value of the average ionisation potential used by the program. Recommended Sternheimer parameters and ionisation potentials are automatically provided by the program for elemental materials. For compounds and mixtures, see [164].
3. STERNHEImе is one of the two FLUKA options where SDUM is used to input numerical data. (Actually, the material number is first read as a string and then an internal reading is performed on the string to get the number).

Example:

```
*...+....1....+....2....+....3....+....4....+....5....+....6....+....7...+...8
MATERIAL         29.    63.546      8.96       12.       0.0       0. COPPER
STERNHEIme    4.4190   -0.0254    3.2792   0.14339    2.9044      0.08  12
*  Use the copper Sternheimer parameters published in At. Data Nucl. Data
*  Tab. 30, 261-271 (1984)
```

7.69 STOP

Stops the execution of the program

See also START

WHAT(1) – **WHAT(6)** and **SDUM**: not used

Default (option STOP not given): no effect (the program stops at the end of the run when the conditions set in the START command (p. 199) are satisfied).

Notes

1. Inserted at any point in a FLUKA input sequence before the START command, a card STOP interrupts input reading and de-activates all the following cards. It can thus help in debugging input. After START, its presence is optional and has no effect.
2. When running the geometry debugger or plotting a slice of the geometry, it is often convenient to place a STOP command just after the GEOEND cards or after the PLOTGEOM (p. 179) input. Otherwise, once the debugging or plotting has been completed, FLUKA would continue reading input cards and eventually would start particle transport as soon as a card START is found.

Example:

```
*...+....1....+....2....+....3....+....4....+....5....+....6....+....7...+...8
GEOEND          150.       75.      220.       30.        0.     -220.DEBUG
GEOEND          120.        1.      110.        0.        0.        0. &
STOP
*   Debugs the geometry and stops without starting a simulation
```

7.70 TCQUENCH

Sets time cut-offs and/or quenching factors when scoring using the USRBIN or the EVENTBIN options.

See also TIME–CUT

WHAT(1) $>$ 0.0: time cut-off for scoring (seconds)
$<$ 0.0: resets any previously requested time cut-off to default (∞)
$=$ 0.0: ignored

WHAT(2) $>$ 0.0: first Birks law coefficient in g/(MeV·cm^2)
$=$ 0.0: ignored
$<$ 0.0: resets both Birks law coefficients to default $=$ 0.0

WHAT(3) $>$ 0.0: second Birks law coefficient in g/(MeV·cm^2)
Default $=$ 0.0

WHAT(4) = lower index bound of binnings in which the requested scoring time cut-off and/or Birks law coefficients must be applied
("*From binning WHAT(4)...*")
Default $=$ 1.0

WHAT(5) = upper index bound of binnings in which the requested scoring time cut-off and/or Birks law coefficients must be applied
("*... to binning WHAT(5)...*")
Default $=$ WHAT(4)

WHAT(6) = step length in assigning indices.
("*... in steps of WHAT(6)*")
Default $=$ 1.0

SDUM : not used

Default (option TCQUENCH not given): no time cut-off for scoring and no quenching of dose binning

Notes

1. Binnings are numbered sequentially in the order they are input via the USRBIN or EVENTBIN options. Of course, for quenching to be applied the quantity binned must be energy (generalised particle 208 or 211). The energy deposited in a charged particle step is "quenched" according to Birks law, i.e., it is weighted with a factor dependent on stopping power $S = \mathrm{d}E/\mathrm{d}x$:
$$\mathrm{d}E' = \frac{\mathrm{d}E}{1 + BS + CS^2}$$
with B = first Birks parameter and C = Chou or second Birks parameter [31].
2. The time cut-off is useful in order to score only within a time gate between $t = 0$ and $t = t_{\mathrm{cut-off}}$
3. The *scoring* time cut-off should not be confused with the time cut-off *for transport* (see TIME–CUT, p. 206). Particles outside the time gate defined by TCQUENCH are still transported and can contribute to scoring in regions and in binnings having a different time scoring cut-off.

Example:

```
*...+....1....+....2....+....3....+....4....+....5....+....6....+....7...+...8
TCQUENCH        20.0   7.35E-3   1.45E-5       4.0      10.0       3.0
*  Set a 20 sec time scoring cut-off for binnings 4, 7 and 10, and apply to
*  them the NE213 scintillator Birks parameters published by Craun and
*  Smith, Nucl. Instr. Meth. 80, 239 (1970)
```

7.71 THRESHOLd

Defines the energy threshold for star density scoring.
Sets thresholds for elastic and inelastic hadron reactions.

See also PART–THR

WHAT(1) and **WHAT(2)**: not used

WHAT(3) = threshold kinetic energy for hadron elastic scattering
Default: same as for particle transport (set by option PART–THR (p. 172) or by default (i.e., the minimum one for which the present FLUKA physics can work, typically 0.02/0.05 GeV depending on the hadron)

WHAT(4) = threshold kinetic energy for hadron inelastic reactions
Default: same as for particle transport, as defined by option PART–THR or by default. The default thresholds are: 0.0 for protons and pions, 0.02 GeV for high energy neutrons, 0.0 for particles which annihilate at rest — but the latter will not undergo any inelastic interaction between the interaction threshold (in the range 0.0 and 0.050 GeV depending on the hadron), and 0.0 — and between 0.0 and 0.050 GeV for all other hadrons.

WHAT(5) : not used

WHAT(6) = threshold kinetic energy for star scoring
Default: the same values as set for inelastic reactions (see WHAT(4) above)

SDUM : not used

Default (option THRESHOLd not given): the threshold for star scoring is set at 20 MeV for protons and neutrons, and at 50 MeV for all other hadrons

Notes

1. The possibility to change the threshold for elastic scattering or inelastic collisions (WHAT(3) and WHAT(4)) is not to be used in normal transport problems, but is made available to investigate the relative importance of different processes on the studied problem.
2. For reasons explained in Note 3 to command SCORE (7.64), it is recommended to set the threshold for star scoring (WHAT(6)) equal to 0.050 GeV overriding the default of 0.020 GeV. A 0.050 GeV cut-off was used in the past to establish so-called ω-factors which are still currently used to estimate induced activity (see [86,87,127]). Stars defined in this way do not include those produced by annihilating particles.
3. The threshold for star scoring requested by option THRESHOLd applies *only* to the output related to options SCORE and USRBIN (7.77). The number of stars reported in the summary statistics at the end of the standard output is always based on the actual energy thresholds for nonelastic interactions (except for neutrons for which stars are reported above 20 MeV).

Example 1:

```
*...+....1....+....2....+....3....+....4....+....5....+....6....+....7...+...8
THRESHOLd        0.0       0.0       2.0       0.0       0.0       0.0
*  Switch off elastic scattering of hadrons below 2 GeV
```

Example 2:

```
*...+....1....+....2....+....3....+....4....+....5....+....6....+....7...+...8
THRESHOLd        0.0       0.0       0.0      0.05       0.0       0.0
*  Score stars only above 50 MeV
```

7.72 TIME–CUT

Sets transport time cut-offs for different particles

See also TCQUENCH

WHAT(1) = transport time cut-off (ns)
Default : no cut off

WHAT(2) = transport time cut-off delay (ns)
Default = 0.0

WHAT(3) = material number for the "start" signal from which time is measured (useful for calorimetry).
Not implemented at present
Default : "start" at creation time

WHAT(4) = lower bound of the particle numbers for which the transport time cut-off and/or the start signal is to be applied
("*From particle WHAT(4)...*")
Default = 1.0

WHAT(5) = upper bound of the particle numbers for which the transport time cut-off and/or the start signal is to be applied
("*... to particle WHAT(5)...*")
Default = WHAT(4) if WHAT(4) < 0.0, all particles otherwise

WHAT(6) = step length in assigning particle numbers
("*... in steps of WHAT(6)*")
Default = 1.0

SDUM : not used

Default : (option TIME–CUT not given): no time cut-off for particle transport

Notes

1. The *transport* time cut-off defined by TIME–CUT should not be confused with the time cut-off *for scoring* defined by TCQUENCH (see Note 3 to option TCQUENCH, p. 204).
2. Particles outside the time gate defined by TIME–CUT are discarded (a summary is printed at the end of the standard output).

Example:

```
*...+....1....+....2....+....3....+....4....+....5....+....6....+....7...+...8
TIME-CUT      3000.0       0.0       0.0       8.0       0.0       0.0
*  Stop transporting neutrons after 3000 nsec
```

7.73 TITLE

Defines the title of the run

See also COMMENT

WHAT(1) – **WHAT(6)** and **SDUM** are not used.

Default : (option TITLE not given): the title is left blank

Notes

1. The title of the run must be given on the following card. Only one title may exist: if more than one is given, only the last one is retained. The title is printed at the top of the standard output and of each estimator output.
2. Giving a title is not mandatory, but it is recommended in order to identify the current run. The title is printed on all estimator files.

Example:

```
*...+....1....+....2....+....3....+....4....+....5....+....6....+....7...+...8
TITLE
Neutron background: 5 GeV electrons on a Cu slab 10 cm thick, first test
```

7.74 USERDUMP

Defines a collision "tape" (or a phase space file) to be written.

WHAT(1) $\geq$ `100.0`: A complete dump is written (unformatted) of one or more of the following:
- all source particles
- all trajectories
- all local (pointlike) energy deposition events
- all continuous energy deposition events
- user-defined events

$=$ `0.0`: ignored
$<$ `0.0`: the default is reset, i.e., no dump is written
$>$ `0.0` and $<$ `100.0`: not allowed! Originally used to request another form of collision tape. Presently only the "new" form of collision tape is possible (the old one being incompatible with the present version of FLUKA)

WHAT(2) = number of the unformatted output unit. Values of WHAT(2) $<$ `21.0` must be avoided because of possible conflicts with FLUKA pre-defined units.

Default : `49.0`

WHAT(3) $\leq$ `0.0`: source particles, trajectories, continuous and local energy losses are all dumped
$=$ `1.0`: only source particles are dumped
$=$ `2.0`: only trajectories and continuous energy losses are dumped
$=$ `3.0`: only local energy losses are dumped (e.g., heavy recoil kerma, cut-off energy). Proton recoils are not included (since recoil protons are transported by FLUKA)
$=$ `4.0`: source particles, trajectories and continuous energy losses are dumped
$=$ `5.0`: source particles and local energy losses are dumped
$=$ `6.0`: trajectories and all energy losses (continuous and local) are dumped
$\geq$ `7.0`: source particles, trajectories, continuous and local energy losses are not dumped (but user-defined dumps required by WHAT(4) are unaffected)

Default: source particles, trajectories, continuous and local energy losses are all dumped, provided WHAT(1) $>$ `1.0`

WHAT(4) $\geq$ `1.0`: user-defined dumps after collisions are activated
$=$ `0.0`: ignored
$<$ `0.0`: resets to default (user dependent dumps after collisions are de-activated)

WHAT(5) – **WHAT(6)**: not used

SDUM = name of the output file (max. 10 characters)

Default (option USERDUMP not given): no dump will be written

Note

1. The format of the collision tape, the code number of the events and the variables written for the different type of events, are described in Chap. 11. (Be careful, the amount of output can be enormous.)
2. The default options described above can be changed by modifying the user routine `MGDRAW` (see description in 12.2.13)

Example:

```
*...+....1....+....2....+....3....+....4....+....5....+....6....+....7...+...8
USERDUMP        200.      37.0       2.0                               TRAKFILE
* It is requested to write a file TRAKFILE, containing all trajectories and
* continuous energy losses, and pre-connected to the logical output unit 37.
```

7.75 USERWEIG

Defines the extra weighting applied to yields scored via the USRYIELD option, energy and star densities obtained via SCORE and USRBIN, energy deposition and and star production obtained via EVENTBIN, production of residual nuclei obtained via RESNUCLEi, currents calculated by means of USRBDX, and fluences calculated by means of USRBDX, USRTRACK, USRCOLL and USRBIN.

WHAT(1) and **WHAT(2)**: not used

WHAT(3) $>$ `0.0`: yields obtained via USRYIELD and fluences or currents calculated with USRBDX, USRTRACK, USRCOLL, USRBIN are multiplied by a user-supplied function `FLUSCW` at scoring time (see p. 306).

`1.0` $\leq$ WHAT(3) $\leq$ `2.0`: `FLUSCW` is called before any check on the current detector (see Note 5)

$>$ `2.0`: `FLUSCW` is called only after checking that the current detector applies (see Note 5)

$=$ `2.0` or `4.0`: The routine `FLDSCP` is also called, applying a shift to the current binned track

$<$ `0.0`: resets the default: no weighting

$=$ `0.0`: ignored

Default $=$ `-1.0` (no weighting)

WHAT(4) : not used

WHAT(5) $>$ `0.0`: the `USRRNC` user function is called every time a residual nucleus is generated. The residual nucleus score is multiplied by the returned value

WHAT(6) $>$ `0.0`: energy and star densities obtained via SCORE (p. 195) and USRBIN (p. 218), as well as energy deposition and star production obtained via EVENTBIN (p. 111) are multiplied by a user-supplied function `COMSCW` at scoring time (see p. 304).

`1.0` $\leq$ WHAT(6) $\leq$ `2.0`: `COMSCW` is called before any check on the current estimator (see Note 5)

$>$ `2.0`: `COMSCW` is called only after checking that the current detector applies (see Note 5)

$=$ `2.0` or `4.0`: The routine `ENDSCP` is also called, applying a shift to the current binned energy loss

$<$ `0.0`: resets the default: no weighting

$=$ `0.0`: ignored

Default $=$ `-1.0` (no weighting)

SDUM not used

Default (option USERWEIG not given): no extra weighting is applied to any scored quantity

Notes

1. These weights are really extra, i.e., the results are multiplied by these weights at scoring time, but printed titles, headings and normalisations are not necessarily valid. It is the user's responsibility to interpret correctly the output. Actually, it is recommended to insert into standard output a user-written notice informing about the extra weighting
2. Setting the incident particle weight to a value different from `1.0` (in the BEAM card, p. 64) will not affect the results, since the latter are always normalised to unit primary weight.
3. Note that USRBIN (p. 218) can be used to calculate star or energy density, and in this case function `COMSCW` has to be used. But when using USRBIN to calculate track-length fluences, the function to be used is `FLUSCW`.
4. Note that functions `FLUSCW`, `COMSCW` and `USRRNC` can contain user-written logic to tune the multiplication factor (which can have even a value = `0.0` or `1.0`!) according to position in space, direction, energy, material, type of particle, time, binning number etc. This allows to score only under certain conditions, or in any case to extend considerably the capability of the code. Similar possibilities exist for the offset provided by routines `FLDSCP` and `ENDSCP`.
5. For some applications, a call to the user routines `FLUSCW` or `COMSCW` is desired independently of whether the current estimator applies. But in general this is not the case and it is convenient to check first that a score actually is taking place, saving a large number of function calls. Different values of WHAT(3) and WHAT(6) allow the use to choose one of the two possibilities.
6. User-written functions `FLUSCW`, `COMSCW`, `FLDSCP`, `ENDSCP` and `USRRNC` are described in Chap. 12.

Examples:

```
*...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....
USERWEIG          0.        0.        0.        0.        0.       1.
*  Dose and star densities will be multiplied by a value returned by
*  function COMSCW according to the logic written by the user.
*  No check on the detector is done before calling the function.

USERWEIG          0.        0.        4.        0.        0.       0.
*  Fluences and currents will be multiplied by a value returned by
*  function FLUSCW according to the logic written by the user.
*  The function will be called only for detectors to which the present
*  score applies.

USERWEIG          0.        0.        0.        0.        1.       0.
*  Residual nuclei scores will be multiplied by function USRRNC according to
*  the logic written by the user.
```

7.76 USRBDX

Defines a detector for a boundary crossing fluence or current estimator

See also USRBIN, USRCOLL, USRTRACK, USRYIELD

The full definition of the detector may require two successive cards (the second card, identified by the character "&" in any column from 71 to 78 (or in the last field in case of free input format), must be given unless the corresponding defaults are acceptable to the user)

First card:

WHAT(1) = $i_1+i_2 \times 10 + i_3 \times 100$, where i_1, i_2, i_3 have the following meaning:

i_1 = +1.0: linear binning in energy and solid angle
= -1.0: logarithmic binning in energy, linear in solid angle
= +2.0: logarithmic binning in solid angle, linear in energy
= -2.0: logarithmic binning in energy and solid angle
i_2 = 0.0: one way scoring
= +1.0: two-way scoring
i_3 = 0.0: current scoring
= +1.0: fluence scoring (inverse cosine-weighted)

Default = 1.0 (one-way current, linear in energy and solid angle)

WHAT(2) = (generalised) particle type to be scored
Default = 201.0 (all particles)

WHAT(3) = logical output unit
> 0.0: formatted data are written on the WHAT(3) unit
< 0.0: unformatted data are written on the |WHAT(3)| unit
Values of |WHAT(3)| < 21.0 should be avoided (with the exception of +11 = standard output unit).
Default = 11.0 (standard output unit)

WHAT(4) = first region defining the boundary (in case of one-way scoring this is the upstream region)
Default = 1.0

WHAT(5) = second region defining the boundary (in case of one-way scoring this is the downstream region)
Default = 2.0

WHAT(6) = area of the detector in cm^2
Default = 1.0

SDUM = any character string (not containing "&") identifying the detector (max. 10 characters)

Continuation card:

WHAT(1) = maximum kinetic energy for scoring (GeV)
Default = beam particle total energy as set by the BEAM option (if no BEAM card is given, the energy corresponding to 200 GeV/c momentum will be used)

WHAT(2) = minimum kinetic energy for scoring (GeV)
Default = 0.0 if linear energy binning, 0.001 GeV otherwise

WHAT(3) = number of energy intervals for scoring
Default = `10.0`

WHAT(4) = maximum solid angle for scoring in sr
Default = 2π for one-way estimators, 4π for two-way

WHAT(5) : If linear angular binning: minimum solid angle for scoring (sr)
Default = `0.0`
If logarithmic angular binning: solid angle of the first bin (sr)
Default = `0.001`

WHAT(6) = number of angular bins
Default = `1.0` for linear angular binning, `3.0` otherwise

SDUM = "`&`" in any position in column 71 to 78 (or in the last field if free format is used)

Default (option USRBDX not given): no boundary crossing estimator

Notes

1. **IMPORTANT!** The results of a USRBDX boundary crossing estimator are always given as *double differential* distributions of fluence (or current) in energy and solid angle, in units of cm^{-2} GeV^{-1} sr^{-1} per incident primary, *even when only 1 interval (bin) has been requested*, which is often the case for angular distributions.
   Thus, for example, when requesting a fluence or current energy spectrum, with no angular distribution, to obtain *integral binned* results (fluence or current in cm^{-2} *per energy bin* per primary) one must multiply the value of each energy bin by the width of the bin (even for logarithmic binning), *and by* 2π or 4π (depending on whether one-way or two-way scoring has been requested).
2. Angular distributions must be intended as distributions in $\cos\theta$, where θ is the angle between the particle trajectory and the normal to the boundary at the point of crossing.
   When logarithmic scoring is requested for angular distributions, all intervals have the same logarithmic width (equal ratio between upper and lower limit of the interval), *except the first one*. The limits of the first angular interval are $\theta = 0$ and the value indicated by the user with WHAT(5) in the second USRBDX card.
3. If the generalised particle is `208.0` or `211.0`, the quantity scored is differential energy fluence (if cosine-weighted) or differential energy current (energy crossing the surface). In both cases the quantity will be expressed in GeV per cm^2 per energy unit per steradian per primary. That can sometimes lead to confusion since GeV cm^{-2} GeV^{-1} sr^{-1} = cm^{-2} sr^{-1}, where energy does not appear. Note that integrating over energy and solid angle one gets $\mathrm{GeV/cm}^2$.
4. The maximum number of boundary crossing detectors that the user can define is 1100. This value can be changed by modifying the parameter `MXUSBX` in member `USRBDX` of the `flukaadd` library or directory and then re-compiling and linking FLUKA.
5. The logical output unit for the estimator results (WHAT(3) of the first USRBDX card) can be any one of the following:
   - the standard output unit `11`: estimator results will be written on the same file as the standard FLUKA output.
   - a pre-connected unit (via a symbolic link on most UNIX systems, `ASSIGN` under VMS, or equivalent commands on other systems)
   - a file opened with the FLUKA command OPEN
   - a file opened with a Fortran `OPEN` statement in a user-written initialisation routine such as `USRINI` or `SOURCE` (see Chap. 12.2.26, 12.2.19)
   - a dynamically opened file, with a default name assigned by the Fortran compiler (typically `fort.xx` or `ftn.xx`, with `xx` equal to the chosen logical output unit number).

   The results of several USRBDX detectors in a same FLUKA run can be written on the same file, but of course only if they are all in the same mode (all formatted, or all unformatted).
   It is also possible in principle to write on the same file the results of different kinds of estimators (USRTRACK, USRBIN, etc.) but this is not recommended, especially in the case of an unformatted file, because it would make very difficult any reading and analysis.

6. When scoring neutron fluence or current, and the requested energy bin structure overlaps with that of the low-energy neutron groups, bin boundaries are forced to coincide with group boundaries and no bin can be smaller than the corresponding group.
   Actually, the program uses the requested energy limits and number of bins to estimate the desired bin width. The number of bins above the upper limit of the first low-energy neutron group is recalculated according to such width.
   Note that the energy limits of the thermal neutron group are 10^{-14} GeV (10^{-5} eV) and 4.14×10^{-10} GeV (0.414 eV) for the ENEA data sets. All group energy boundaries are listed in Table 10.1 on p. 294.
7. An example on how to read USRBDX unformatted output is given below. An explanation of the meaning of the different variables is given in the comments at the beginning of the program. The program lists the bin energy boundaries (in increasing order) for each angular interval, and the corresponding differential and cumulative fluence (energy-integrated within each angle interval, and integrated over both energy and angle).
   A more complex program `USXSUW`, which allows to compute also standard deviations over several runs, is available with the normal FLUKA code distribution in directory `$FLUPRO/flutil`.

```
      PROGRAM RDBDX
*----------------------------------------------------------------------
*  Up to MXUSBX user-defined bdrx detectors are allowed
*
*    ABXHGH = maximum angle (steradian) (WHAT(4) of 2nd card)
*    ABXLOW = minimum angle (steradian) (WHAT(5) of 2nd card)
*    AUSBDX = area (cm**2) of the detector (WHAT(6) of first card)
*    DABXBN = angular bin width (steradian) if linear in angle,
*             otherwise ratio between upper and lower edge of angular
*             intervals
*    DEBXBN = energy bin width (GeV) if linear in energy or referring
*             to a low-energy neutron energy group, otherwise
*             ratio between upper and lower edge of energy intervals
*    EBXHGH = maximum energy (GeV) (WHAT(1) of 2nd card)
*    EBXLOW = minimum energy (GeV) (WHAT(2) of 2nd card)
*    ENGMAX = upper energies (GeV) of the neutron groups
*    IDUSBX = (generalized) particle scored (WHAT(2) of first card)
*    IGMUSX = maximum low-energy neutron group to be scored
*    ITUSBX = type of binning (i1 in WHAT(1) of first card):
*              1 = linear energy, linear angle
*              2 = linear energy, logarithmic angle,
*             -1 = logarithmic energy, linear angle,
*             -2 = logarithmic energy, logarithmic angle
*    LFUSBX = current if .F., fluence if .T. (i3 in WHAT(1) of card 1)
*    LLNUSX = no low-energy neutron scoring if false, yes if true
*    LWUSBX = one way if false, 2-ways if true (i2 in WHAT(1) of card 1)
*    MX     = id-number of the boundary crossing detector
*    NABXBN = number of angular intervals (WHAT(6) of 2nd card)
*    NCASE  = number of beam particles followed
*    NEBXBN = number of energy intervals
*    NR1USX = first region (WHAT(4) of first card)
*    NR2USX = second region (WHAT(5) of first card)
*    RUNTIM = date and time of the run (as printed on standard output)
*    RUNTIT = title of the run (as given by card TITLE)
*    SCORED = result array
*    TITUSX = detector name (SDUM in first USBDRX card)
*    WEIPRI = total weight of primaries
*    NHIGH  = number of energy bins above low-energy neutron limit
*    CUMULE = energy-cumulative fluence/current
*    CUMULA = energy-angle-cumulative fluence/current
*    ELIMIT = lower energy of the first bin above or straddling the
*             n-group limit
*
*----------------------------------------------------------------------
      PARAMETER ( MXUSBX = 1500 )      ! max. number of usbdrx detectors
      PARAMETER ( MXENER = 1000)       ! storage for results
      PARAMETER ( MXANGL = 100)        !
```

```
      PARAMETER ( MXSCOR = MXENER*MXANGL )
      PARAMETER ( NMXGRP = 100 )       ! # of low-energy neutron groups

      LOGICAL LFUSBX, LWUSBX, LLNUSX
      CHARACTER RUNTIT*80,RUNTIM*32,TITUSX*10,FILNAM*80

      DIMENSION EBXLOW(MXUSBX), EBXHGH(MXUSBX), ABXLOW(MXUSBX),
     &          ABXHGH(MXUSBX), DEBXBN(MXUSBX), DABXBN(MXUSBX),
     &          AUSBDX(MXUSBX), NEBXBN(MXUSBX), NABXBN(MXUSBX),
     &          NR1USX(MXUSBX), NR2USX(MXUSBX), ITUSBX(MXUSBX),
     &          IDUSBX(MXUSBX), IGMUSX(MXUSBX), LFUSBX(MXUSBX),
     &          LWUSBX(MXUSBX), LLNUSX(MXUSBX), TITUSX(MXUSBX),
     &          ENGMAX (NMXGRP+1), SCORED(MXSCOR), ELIMIT(MXUSBX),
     &          MX(MXUSBX), NHIGH(MXUSBX)

      DOUBLE PRECISION CUMUL, ANGINT, EN1, EN2, ELIMIT, DIFF

      WRITE(*,*)' Type the name of the input file:'
      READ (*,'(A)') FILNAM
      LQ = INDEX(FILNAM,' ') - 1
      OPEN (UNIT=1, FILE=FILNAM, STATUS='OLD', FORM='UNFORMATTED')
      OPEN (UNIT=2, FILE=FILNAM(1:LQ)//'.txt', STATUS='NEW')
*----------- read and write 1st record ---------------------------------
      READ (1) RUNTIT,RUNTIM,WEIPRI,NCASE
      WRITE(2,100) RUNTIT, RUNTIM, NCASE, WEIPRI
 100  FORMAT(/,1X,'*****',2X,A80,2X,'*****',/,/,10X,A32,/,/,
     &        10X,'Total number of particles followed ',I9,', for a ',
     &        'total weight of ',1P,E15.8,/)
*----------- loop on bdryx detector data in the present file -----------
      DO 1 IX = 1, MXUSBX
         NX = IX
*        ---------------- read and write 2nd record --------------------
         READ (1, END=1000) MX(NX), TITUSX(NX), ITUSBX(NX), IDUSBX(NX),
     &      NR1USX(NX), NR2USX(NX), AUSBDX(NX), LWUSBX(NX), LFUSBX(NX),
     &      LLNUSX(NX), EBXLOW(NX), EBXHGH(NX), NEBXBN(NX), DEBXBN(NX),
     &      ABXLOW(NX), ABXHGH(NX), NABXBN(NX), DABXBN(NX)
         WRITE(2,101) MX(NX),
     &     TITUSX(NX), IDUSBX(NX), NR1USX(NX), NR2USX(NX), AUSBDX(NX)
*        ------------ if low-en. neutrons, read group energies ---------
         IF ( LLNUSX (NX) ) THEN
            READ (1) IGMUSX(NX), (ENGMAX(IG), IG = 1, IGMUSX(NX)+1)
            WRITE (2,102) IGMUSX(NX)
         ELSE
            IGMUSX(NX) = 0
         END IF
*        ---------------------- 1 or 2-way -----------------------------
         IF ( LWUSBX (NX) ) THEN
            WRITE (2,'(6X,A)') 'this is a two ways estimator'
         ELSE
            WRITE (2,'(6X,A)') 'this is a one way only estimator'
         END IF
*        ---------------------- fluence or current ---------------------
         IF ( LFUSBX (NX) ) THEN
            WRITE (2,'(6X,A)') 'this is a fluence-like estimator'
         ELSE
            WRITE (2,'(6X,A)') 'this is a current-like estimator'
         END IF
*        -------------------- linear or log in energy ------------------
         IF ( ITUSBX (NX) .GT. 0 ) THEN
            WRITE (2,103) EBXLOW(NX), EBXHGH(NX), NEBXBN(NX), DEBXBN(NX)
         ELSE
            WRITE (2,104) EBXLOW(NX), EBXHGH(NX), NEBXBN(NX), DEBXBN(NX)
```

```
         END IF
*        -------------------- linear or log in angle -------------------
         IF ( ABS (ITUSBX (NX)) .LE. 1 ) THEN
            WRITE (2,105) ABXLOW(NX), ABXHGH(NX), NABXBN(NX), DABXBN(NX)
         ELSE
            WRITE (2,106) ABXLOW(NX), ABXHGH(NX), NABXBN(NX), DABXBN(NX)
         END IF
*        interv = total number of energy intervals
*        (intervals above the limit for n-groups + intervals below)
         INTERV = NEBXBN(NX) + IGMUSX(NX)
*------------ read the scoring results as a 1-dimensional array --------
         READ (1) (SCORED(J), J = 1, INTERV * NABXBN(NX))
*------------ loop on angles -----------------------------------------
         CUMUL = 0.D0
         DO 2 IA = 1, NABXBN(NX)
            IF ( ABS (ITUSBX (NX)) .LE. 1 ) THEN
*              linear in angle
               WRITE(2,107) IA, ABXLOW(NX) + (IA-1)*DABXBN(NX),
     &                      ABXLOW(NX) + IA*DABXBN(NX)
            ELSE
*              logarithmic in angle
               IF(IA .EQ. 1) THEN
                  WRITE (2,108) ABXLOW(NX)
               ELSE
                  WRITE (2,107) IA, ABXLOW(NX) * DABXBN(NX)**(IA-1),
     &                            ABXLOW(NX) * DABXBN(NX)**IA
               END IF
            END IF
            WRITE (2,'("                              ",
     &        "       Double Differential   Angle-Integrated",
     &        "          Cumulative" )')
            IF ( LFUSBX (NX) ) THEN
               WRITE (2,'("                                ",
     &                   "    Fluence (dPhi/dE/dOmega)      dPhi/dE ",
     &                   "              Fluence")')
            ELSE
               WRITE (2,'("                                ",
     &                    "     Current (dJ/dE/dOmega)      dJ/dE ",
     &                    "             Current")')
            END IF
            WRITE (2,'("Lower energy      Upper energy",
     &                 "    cm**-2 GeV**-1 sr-1       cm**-2 GeV-1",
     &                 "             cm**-2"/)')
            ELIMIT(NX) = EBXLOW(NX)
*           -------------------- low-energy neutrons -------------------
            IF ( LLNUSX (NX) ) THEN
*              low-energy neutron data, if present, are stored backwards
               IG  = IGMUSX(NX)
               EN1 = ENGMAX(IG+1)
*              ------------- loop on low-energy groups -----------------
               DO 3 JG = IA*(NEBXBN(NX)+IGMUSX(NX)),
     &                   IA*(NEBXBN(NX)+IGMUSX(NX)) - IGMUSX(NX) + 1, -1
                  EN2 = ENGMAX(IG)
                  ANGINT = SCORED(JG) * DABXBN(NX)
                  CUMUL = CUMUL + ANGINT * (EN2 - EN1)
                  WRITE(2,109) EN1, EN2, SCORED(JG), ANGINT, CUMUL
                  IG = IG - 1
                  EN1 = EN2
 3             CONTINUE                 ! end loop on low-energy groups
*              find lower limit of first bin above or straddling the
*              n-group limit. Nhigh: counts the high energy bins
               NHIGH(NX) = 0
```

```
*              for the time being, set energy boundary at n-group limit
               ELIMIT(NX) = EN1
               EN1 = EBXHGH(NX)
               DO 4 IE = 1, NEBXBN(NX)
                  IF(ITUSBX(NX) .GT. 0) THEN
                     EN2 = EN1 - DEBXBN(NX)
                  ELSE
                     EN2 = EN1 / DEBXBN(NX)
                  END IF
                  EN1 = EN2
                  NHIGH(NX) = NHIGH(NX) + 1
                  IF(EN2 .LE. ELIMIT(NX)) GO TO 5
 4             CONTINUE
 5             CONTINUE
            ELSE
               EN1 = EBXLOW(NX)
               NHIGH(NX) = NEBXBN(NX)
            END IF
*           ------------------------------------------------------------
*           first bin above or straddling the n-group limit
            IF(ITUSBX(NX) .GT. 0) THEN
               EN2 = EN1 + DEBXBN(NX)
            ELSE
               EN2 = EN1 * DEBXBN(NX)
            END IF
            DIFF = SCORED(IA * (NEBXBN(NX)) - NHIGH(NX) + 1)
            ANGINT = DIFF * DABXBN(NX)
            CUMUL  = CUMUL + ANGINT * (EN2 - ELIMIT(NX))
            WRITE(2,109) ELIMIT(NX), EN2, DIFF, ANGINT, CUMUL
            EN1 = EN2
*           -------- loop on energies above the n-group limit ----------
            DO 6 IE = 2, NHIGH(NX)
               IF(ITUSBX(NX) .GT. 0) THEN
                  EN2 = EN1 + DEBXBN(NX)
               ELSE
                  EN2 = EN1 * DEBXBN(NX)
               END IF
               DIFF = SCORED(IA * (NEBXBN(NX)) - NHIGH(NX) + IE)
               ANGINT = DIFF * DABXBN(NX)
               CUMUL = CUMUL + ANGINT * (EN2 - EN1)
               WRITE(2,109) EN1, EN2, DIFF, ANGINT, CUMUL
               EN1 = EN2
  6         CONTINUE            ! end loop on energies above limit
*           Case of generalized particles .ne. 8 but including neutrons
            IF (LLNUSX(NX) .AND. IDUSBX(NX) .NE. 8 .AND.
     &                           NEBXBN(NX) .GT. NHIGH(NX)) THEN
              WRITE(2,110) ELIMIT(NX)
              EN1 = EBXLOW(NX)
              DO 7 IE = 1, NEBXBN(NX) - NHIGH(NX)
                 IF(ITUSBX(NX) .GT. 0) THEN
                    EN2 = EN1 + DEBXBN(NX)
                 ELSE
                    EN2 = EN1 * DEBXBN(NX)
                 END IF
                     DIFF = SCORED(IA * (NEBXBN(NX)) - NHIGH(NX) + IE)
                 DIFF = SCORED((IA-1) * (NEBXBN(NX)+IGMUSX(NX)) + IE)
                 ANGINT = DIFF * DABXBN(NX)
                 CUMUL = CUMUL + ANGINT * (EN2 - EN1)
                 WRITE(2,109) EN1, EN2, DIFF, CUMULE, CUMULA
                 EN1 = EN2
 7            CONTINUE
            END IF
```

```
*          ------------------------------------------------------------
 2       CONTINUE               ! end loop on angles
*        ------------------------------------------------------------------
 1    CONTINUE                  ! end loop on detectors
*     ---------------------------------------------------------------------
 1000 CONTINUE

 101  FORMAT(/,3X,'Bdrx n. ',I3,'  "',A10,
     &    '" , generalized particle n. ',I4,', from region n. ',I6,
     &    ' to region n. ',I6,/,6X,'detector area: ',1P,E15.8,' cm**2')
 102  FORMAT(6X,'low-energy neutrons scored from group 1 to group ', I5)
 103  FORMAT(6X,'linear energy binning from  ',1P,E11.4,' to ',
     &    E11.4,' GeV, ',0P,I6,' bins (',1P,E11.4,' GeV wide)')
 104  FORMAT(6X,'logar. energy binning from ',1P,E11.4,' to ',
     &    E11.4,' GeV, ',0P,I6,' bins (ratio :',1P,E11.4,')')
 105  FORMAT(6X,'linear angular binning from ',1P,E11.4,' to ',
     &    E11.4,' sr , ',0P,I6,' bins (',1P,E11.4,' sr wide )'/)
 106  FORMAT(6X,'logar. angular binning, first bin = ',1P,E11.4,
     &    ' sr, upper limit ',E11.4,' sr, ',0P,I6,' bins ( ratio :',
     &    1P,E11.4,')'/)
 107  FORMAT(/' Angle ',I4,' between ',1P,E11.4,' and ',E11.4,' sr'/)
 108  FORMAT(/' Angle    1 between 0 and ',1P,E11.4, ' sr'/)
 109  FORMAT(1P, 2(E11.4, 7X), 3(E14.7, 8X))
 110  FORMAT(/,20X,"Particles other than neutrons below E = ",1P,E11.4/)
      END
```

Example:

```
*...+....1....+....2....+....3....+....4....+....5....+....6....+....7...+...8
USRBDX         101.0       9.0      21.0       3.0       4.0    400.0 AntiNeu
USRBDX           5.0       0.0     200.0       0.0       0.0      0.0 &
*  Calculate fluence spectrum from 0 to 5 GeV, in 200 linear energy intervals,
*  of antineutrons passing from region 3 to region 4 (and not from 4 to 3).
*  Write formatted results on unit 21. The area of the boundary is 400 cm2.
*  A single angular interval is requested (from 0 to 2pi)
```

7.77 USRBIN

Scores distributions of several quantities in a regular spatial structure (binning) independent from the geometry

See also SCORE (scoring by region), EVENTBIN (event-by-event scoring) and USRBDX, USRCOLL, USRTRACK, USRYIELD (fluence estimators)

The full definition of the detector may require two successive cards. The second card, identified by the character "`&`" in any column from 71 to 78 (or in the last field in case of free format input), must be given unless the corresponding defaults are acceptable to the user.

First card:

WHAT(1) : code indicating the type of binning selected. Each type is characterised by a number of properties:

- structure of the mesh (spatial: R–Z, R–Φ–Z, Cartesian, or special — by region, or user-defined)
- quantity scored (energy, star, fission, neutron balance or track-length density)
- method used for scoring (old algorithm where the energy lost in a step by a charged particle is deposited in the middle of the step, or new algorithm where the energy lost is apportioned among different bins according to the relevant step fraction — see more in Note 7)
- mesh symmetry (no symmetry, or specular symmetry around one of the coordinate planes, or around the origin point)

= `0.0`: Mesh: Cartesian, no symmetry
Quantity scored:

- if WHAT(2) = `208.0, 211.0, 229.0 or 230.0`: energy density (deposited with the old algorithm at midstep, see Note 7)
- if WHAT(2) = `219.0, 220.0` or `221.0`: fission density
- if WHAT(2) = `222.0`: neutron balance density
- otherwise, density of stars produced by particles (or families of particles) with particle code = WHAT(2)

= `1.0`: Mesh: R–Z or R–Φ–Z, no symmetry. Φ is the azimuthal angle around the Z axis, measured from $-\pi$ to $+\pi$ relative to the X axis.
Quantity scored: same as for WHAT(1) = `0.0`

= `2.0`: Mesh: by region (1 bin corresponds to n regions, with $n = 1$ to 3)
Quantity scored: same as for WHAT(1) = `0.0`

= `3.0`: Mesh: Cartesian, with symmetry ± X (i.e., `|x|` is used for scoring)
Quantity scored: same as for WHAT(1) = `0.0`

= `4.0`: Mesh: Cartesian, with symmetry ± Y (i.e., `|y|` is used for scoring)
Quantity scored: same as for WHAT(1) = `0.0`

= `5.0`: Mesh: Cartesian, with symmetry ± Z (i.e., `|z|` is used for scoring)
Quantity scored: same as for WHAT(1) = `0.0`

= `6.0`: Mesh: Cartesian, with symmetry around the origin (i.e., `|x|, |y|` and `|z|` are used for scoring)
Quantity scored: same as for WHAT(1) = `0.0`

= `7.0`: Mesh: R–Z or R–Φ–Z, with symmetry ± Z (i.e., `|z|` is used for scoring)
Quantity scored: same as for WHAT(1) = `0.0`

= 8.0: Special user-defined 3D binning. Two variables are discrete (e.g., region number), the third one is continuous, but not necessarily a space coordinate. See 12.2.9 for instructions on how to write, compile and link the user routines.

Variable	Type	Default	Override routine
1^{st}	integer	region number	MUSRBR
2^{nd}	integer	lattice cell number	LUSRBL
3^{rd}	continuous	pseudorapidity	FUSRBV

= 10.0: Mesh: Cartesian, no symmetry
Quantity scored: if WHAT(2) = 208.0, 211.0, 229.0 or 230.0: energy density (apportioned with the new algorithm along the step as explained in Note 7).
If WHAT(2) = 219.0, 220.0 or 221.0: fission density.
Otherwise: fluence (track-length density) of particles (or families of particles) with particle code = WHAT(2)

= 11.0: Mesh: R–Z or R–Φ–Z, no symmetry
Quantity scored: same as for WHAT(1) = 10.0

= 13.0: Mesh: Cartesian, with symmetry ± X (|x| used for scoring)
Quantity scored: same as for WHAT(1) = 10.0

= 14.0: Mesh: Cartesian, with symmetry ± Y (|y| used for scoring)
Quantity scored: same as for WHAT(1) = 10.0

= 15.0: Mesh: Cartesian, with symmetry ± Z (|z| used for scoring)
Quantity scored: same as for WHAT(1) = 10.0

= 16.0: Mesh: Cartesian, with symmetry around the origin (|x|,|y|, |z| used for scoring)
Quantity scored: same as for WHAT(1) = 10.0

= 17.0: Mesh: R–Z or R–Φ–Z, with symmetry ± Z (|z| used for scoring)
Quantity scored: same as for WHAT(1) = 10.0

Default = 0.0 (Cartesian scoring without symmetry, star density or energy density deposited at midstep with the old algorithm)

WHAT(2) : particle (or particle family) type to be scored
If WHAT(2) = 208.0, 211.0, 229.0 or 230.0: if WHAT(1) $<$ 10.0, the binning will score energy deposition with the old algorithm.
If WHAT(1) $\geq$ 10.0, the new deposition algorithm will be used (more accurate, see Note 7).
Any other particle (or family of particles) requested will score:

- if WHAT(1) $<$ 10.0, density of stars produced by particles (or family of particles) with particle code = WHAT(2). Of course, this choice is meaningful only for particles which can produce stars (hadrons, photons and muons).
- if WHAT(1) $\geq$ 10.0, fluence of particles (or family of particles) with particle code = WHAT(2).

Note that it is not possible to score energy fluence with this option alone (it is possible, however, by writing a special version of the user routine FLUSCW — see 12.2.6)

Default = 208.0 (total energy density)

WHAT(3) = logical output unit:
$>$ 0.0: formatted data are written on WHAT(3) unit
$<$ 0.0: unformatted data are written on |WHAT(3)| unit
Values of |WHAT(3)| $<$ 21.0 should be avoided (with the exception of +11).

Default = 11.0 (standard output unit)

WHAT(4) = For Cartesian binning: X_{max}
For R–Z and R–Φ–Z binning: R_{max}
For region binning: last region of the first region set
For special binnings, upper limit of the first user-defined variable (last region if the default version of the `MUSRBR` routine is not overridden)

Default: No default

WHAT(5) = For Cartesian binning: Y_{max}
For R–Z binning: Y coordinate of the binning axis.
For region binning: last region of the second region set
For special binnings, upper limit of the second user-defined variable (last lattice cell if the default version of the `LUSRBL` routine is not overridden)

Default: No default

WHAT(6) = For R–Z, R–Φ–Z and Cartesian binnings: Z_{max}
For region binnings, last region of the third region set
For special binnings, upper limit of the third user-defined variable (η_{max} if the default version of the `FUSRBV` routine is not overridden)

Default: No default

SDUM = any character string (not containing "`&`") identifying the binning (max. 10 characters)

Continuation card: (not needed if the defaults are acceptable)

WHAT(1) = For Cartesian binning: X_{min} (if X symmetry is requested, X_{min} cannot be negative)
For R–Z and R–Φ–Z binning: R_{min}
For region binnings, first region of the first region set. Default: equal to last region (= `WHAT(4)` in the first `USRBIN` card)
For special binnings, lower limit of the first user-defined variable (first region if the default version of the `MUSRBR` routine is not overridden)
Default = `0.0`

WHAT(2) = For Cartesian binning: Y_{min} (if Y symmetry is requested, Y_{min} cannot be negative)
For R–Z binning: X coordinate of the binning axis.
For region binnings, first region of the second region set. Default: equal to last region (= `WHAT(5)` in the first `USRBIN` card)
For special binnings, lower limit of the second user-defined variable (first lattice cell if the default version of the `LUSRBL` routine is not overridden)
Default = `0.0`

WHAT(3) = For Cartesian, R–Z and R–Φ–Z binnings: Z_{min} (if Z symmetry is requested, Z_{min} cannot be negative)
For region binnings, first region of the third region set. Default: equal to last region (= `WHAT(6)` in the first `USRBIN` card)
For special binnings, lower limit of the third user-defined variable (η_{min} if the default version of the `FUSRBV` routine is not overridden)
Default = `0.0`

WHAT(4) = For Cartesian binning: number of X bins. (Default: `30.0`)
For R–Z and R–Φ–Z binning: number of R bins (default: `50.0`)
For region binnings, step increment for going from the first to the last region of the first region set. (Default: `1.0`)
For special binnings, step increment for going from the first to the last "region" (or similar). (Default: `1.0`)

WHAT(5) = For Cartesian binning: number of Y bins. (Default: `30.0`).
For R–Φ–Z: number of Φ bins. (Default is R–Φ–Z = R–Z, i.e., `1` Φ bin).
For region binnings, step increment for going from the first to the last region of the second region set. (Default: `1.0`).
For special binnings, step increment for going from the first to the last "lattice cell" (or similar). (Default: `1.0`).

WHAT(6) = For Cartesian, R–Z and R–Φ–Z binnings: number of Z bins (default: `10.0` for Cartesian, `50.0` for R–Z and R–Φ–Z)
For region binnings, step increment for going from the first to the last region of the third region set. (Default: `1.0`).
For special binnings, number of intervals for the third variable ("η", or similar). (Default: `1.0`).

SDUM = "`&`" in any position in column 71 to 78 (or in the last field if free format is used)

Default (option USRBIN not given): no binning

Notes

1. A *binning* is a regular spatial mesh completely independent from the regions defined by the problem's geometry. On user's request, FLUKA can calculate the distribution of several different quantities over one or more binning structures, separated or even overlapping. The following quantities can be "binned":
   - energy density (dose), total or deposited by $e^+e^-\gamma$ only
   - star density (hadronic inelastic interactions)
   - particle track-length density (fluence)
   - density of high-energy and low-energy fissions
   - neutron balance density (algebraic sum of outgoing neutrons minus incoming neutrons for all interactions)
   - unbiased energy density (physically meaningless but useful for setting biasing parameters and debugging)

   The available binning shapes are Cartesian (3-D rectangular, with planes perpendicular to the coordinate axes), R–Z (2-D cylindrical, with the cylinder axis parallel to the z-axis), and R–Φ–Z (3-D cylindrical).
2. It is possible to define also binnings with an arbitrary orientation in space, by means of options ROT–DEFIni (p. 190) and ROTPRBIN (p. 192).
3. A star is a hadronic inelastic interaction at an energy higher than a threshold defined via the option THRESHOLd (p. 205), or by default higher than the transport threshold of the interacting particle. Star scoring (traditionally used in most high-energy shielding codes) can therefore be considered as a form of crude collision estimator: multiplication of star density by the asymptotic value of the inelastic nuclear interaction length gives the fluence of hadrons having energy higher than the current threshold. However, this is meaningful only if the interaction length doesn't vary appreciably with energy; therefore it is recommended to set a scoring threshold = 50 MeV (using option THRESHOLd), since interaction lengths are practically constant above this energy. Besides, star densities calculated with a 50 MeV threshold are the basis of some established radiation protection techniques such as the ω-factors for estimating material activation (see [169], p. 106), and the prediction of single isotope yields from the ratio of partial to inelastic cross-section.
4. Selecting star scoring is meaningful for hadrons, photons and muons (if their energy is sufficiently high). Any other particle will not produce any star. And in FLUKA, stars do not include spallations due to annihilating particles.
   The results will be expressed in stars per cm^3 per unit primary weight.
5. Energy deposition will be expressed in GeV per cm^3 per unit primary weight. To obtain doses (in Gy per unit primary weight), the results must be multiplied by $1.602176462 \times 10^{-7}/\rho$, where ρ is the material density in g/cm^3. The multiplication may be done off-line by an analysis program, or on-line at the time of scoring by linking a user-written routine `COMSCW` (Chap. 12.2.2). The latter choice allows to calculate the correct average dose even in bins straddling the boundary between two regions of different density.
6. The results from USRBIN are normalised per unit volume and per unit primary weight, except for region binnings and special user-defined binnings which are normalised per unit primary weight only. In case symmetries are requested proper rescaled volumes are taken into account for normalisation (that is, an extra factor 2 is applied to the volume if symmetry around one plane is required, 8 if symmetry around the origin is required)

7. When scoring energy deposition (generalised particles 208.0 and 211.0), it is recommended to set in the first USRBIN card WHAT(1) = 10.0, 11.0,...17.0 (rather than 0.0, 1.0,...7.0).
   The difference between the two settings is the following. With WHAT(1) = 0.0, 1.0,...7.0, the energy lost in a charged particle step is deposited in the bin corresponding to the midpoint of the step: this is the old FLUKA algorithm, which is rather inefficient when the step length is larger than the bin size.
   The new algorithm, selected by setting WHAT(1) = 10.0, 11.0,...17.0, deposits in every bin traversed by the step a fraction of energy proportional to the respective chord (track-length apportioning). Statistical convergence is much faster.
8. When scoring region binning and more than one set of regions is defined, each of the sets (2 or 3) must consist of the same number of regions. The first bin will contain the sum of what is contained in the first regions of each set, the second bin the sum of the scores of the second regions, etc.
9. The maximum number of binnings that the user can define is 400. This value can be changed by modifying the parameter MXUSBN in member USRBIN of the flukapro library or directory and then re-compiling and linking FLUKA.
10. The logical output unit for the estimator results (WHAT(3) of the first USRBIN card) can be any one of the following:
    - the standard output unit 11: estimator results will be written on the same file as the standard FLUKA output.
    - a pre-connected unit (via a symbolic link on most UNIX systems, ASSIGN under VMS, or equivalent commands on other systems)
    - a file opened with the FLUKA command OPEN
    - a file opened with a Fortran OPEN statement in a user-written initialisation routine such as USRINI or SOURCE (see Chap. 12.2.26, 12.2.19)
    - a dynamically opened file, with a default name assigned by the Fortran compiler (typically fort.xx or ftn.xx, with xx equal to the chosen logical output unit number).

    The results of several USRBIN detectors in a same FLUKA run can be written on the same file, but of course only if they are all in the same mode (all formatted, or all unformatted).
    It is also possible in principle to write on the same file the results of different kinds of estimators (USRBDX, USRTRACK, etc.) but this is not recommended, especially in the case of an unformatted file, because it would make very difficult any reading and analysis.
11. In R–Φ–Z binnings, the azimuthal Φ coordinates extend from $-\pi$ to $+\pi$ (-180° to $+180^\circ$). $\Phi = 0$ corresponds to the x-axis.
12. Binning data can be obtained also separately for each "event" ("event" = history of a primary particle and all its descendants). See option EVENTBIN (p. 111) for details.
13. An example on how to read USRBIN unformatted output is given below. An explanation of the meaning of the different variables is given in the comments at the beginning of the program. The program lists for each bin its boundaries and the corresponding scored quantity.
    Two more complex programs, USBSUW and USBREA, are available with the normal FLUKA code distribution in directory $FLUPRO/flutil. USBSUW allows to compute standard deviations over several runs, and returns the standard deviations and the averages in an unformatted file. USBREA reads an unformatted file and returns the equivalent formatted file, including the standard deviations if the input file was produced by USBSUW.

```
      PROGRAM RDBIN
*----------------------------------------------------------------------*
*  Up to MXUSBN user defined binnings are allowed                      *
*     itusbn = type of binning (must be decoded if .ge. 10)            *
*     idusbn = distribution to be scored: usual values allowed         *
*     titusb = binning name                                            *
*     nxbin  = number of x (r for RZ) intervals                        *
*     nybin  = number of y (1 for RZ) intervals                        *
*     nzbin  = number of z intervals                                   *
*  xlow/high = minimum and maximum x (r   for R-Phi-Z)                 *
*  ylow/high = minimum and maximum y (phi for R-Phi-Z)                 *
*  zlow/high = minimum and maximum z                                   *
*     dxusbn = x (r) bin width                                         *
*     dyusbn = y (Phi) bin width                                       *
*     dzusbn = z bin width                                             *
*     tcusbn = time cut-off (seconds) for this binning                 *
*     bkusbn = 1st Birk's law parameter for this binning               *
*              (meaningful only for energy scoring)                    *
```

```
*     b2usbn = 2nd Birk's law parameter for this binning               *
*              (meaningful only for energy scoring)                    *
*     xaxusb = x-axis offset for R-Z binning (not possible for R-Phi-Z)*
*     yaxusb = y-axis offset for R-Z binning (not possible for R-Phi-Z)*
**-------------------------------------------------------------------*
      PARAMETER ( MXUSBN = 100 )    ! max. number of binnings
      PARAMETER ( MXSCOR = 500000 ) ! storage for results

      LOGICAL LUSBIN, LUSEVT, LUSTKB
      CHARACTER RUNTIT*80, RUNTIM*32, TITUSB*10, FILNAM*80, CHSTAT*10

      DIMENSION MB(MXUSBN),XLOW(MXUSBN),  XHIGH(MXUSBN), YLOW(MXUSBN),
     &   YHIGH (MXUSBN), ZLOW  (MXUSBN), ZHIGH (MXUSBN), DXUSBN(MXUSBN),
     &   DYUSBN(MXUSBN), DZUSBN(MXUSBN), TCUSBN(MXUSBN), BKUSBN(MXUSBN),
     &   B2USBN(MXUSBN), NXBIN (MXUSBN), NYBIN (MXUSBN), NZBIN (MXUSBN),
     &   ITUSBN(MXUSBN), IDUSBN(MXUSBN), KBUSBN(MXUSBN), IPUSBN(MXUSBN),
     &   LEVTBN(MXUSBN), LNTZER(MXUSBN), LTRKBN(MXUSBN), TITUSB(MXUSBN),
     &   XAXUSB(MXUSBN), YAXUSB (MXUSBN), SCORED(MXSCOR)


      WRITE(*,*) ' Type the name of the input file:'
      READ (*,'(A)') FILNAM
      LQ = INDEX(FILNAM,' ') - 1
      OPEN (UNIT=1, FILE=FILNAM, STATUS='OLD', FORM='UNFORMATTED')
      OPEN (UNIT=2, FILE=FILNAM(1:LQ)//'.txt', STATUS='NEW')
*----------- read and write 1st record --------------------------------
      READ (1) RUNTIT, RUNTIM, WEIPRI, NCASE
      WRITE(2,100) RUNTIT, RUNTIM, NCASE, WEIPRI
*----------- loop on binning detector data in the present file ---------
      DO 1 IB = 1, MXUSBN
         NB = IB
*        ---------------- read and write 2nd record --------------------
         READ (1,END=1000) MB(NB), TITUSB(NB), ITUSBN(NB), IDUSBN(NB),
     &      XLOW(NB),  XHIGH(NB),  NXBIN(NB),  DXUSBN(NB), YLOW(NB),
     &      YHIGH(NB), NYBIN(NB),  DYUSBN(NB), ZLOW(NB),   ZHIGH(NB),
     &      NZBIN(NB), DZUSBN(NB), LNTZER(NB), BKUSBN(NB), B2USBN(NB),
     &      TCUSBN(NB)
         ITUHLP = MOD (ITUSBN(NB),10)
         IF ( ITUHLP .EQ. 2) THEN
*           Region binning
            NBIN = MAX(NXBIN(NB),NYBIN(NB),NZBIN(NB))
            IR1A = NINT(XLOW(NB))
            IR1B = NINT(XHIGH(NB))
            IDR1 = NINT(DXUSBN(NB))
            IR2A = NINT(YLOW(NB))
            IR2B = NINT(YHIGH(NB))
            IDR2 = NINT(DYUSBN(NB))
            IR3A = NINT(ZLOW(NB))
            IR3B = NINT(ZHIGH(NB))
            IDR3 = NINT(DZUSBN(NB))
            READ(1) (SCORED(J), J = 1, NBIN)
            WRITE(2,101) MB(NB), TITUSB(NB), IDUSBN(NB), NBIN,
     &         IR1A, IR1B, IDR1, IR2A, IR2B, IDR2, IR3A, IR3B, IDR3
            DO 2 I = 1, NBIN
               WRITE(2,1010) IR1A + (I-1)*IDR1, IR2A + (I-1)*IDR2,
     &                       IR3A + (I-1)*IDR3, SCORED(I)
 2          CONTINUE
         ELSE IF ( ITUHLP .EQ. 8 ) THEN
*           Region/Lattice/User binning
            IR1A = NINT(XLOW(NB))
            IR1B = NINT(XHIGH(NB))
            IDR1 = NINT(DXUSBN(NB))
```

```
            IR2A = NINT(YLOW(NB))
            IR2B = NINT(YHIGH(NB))
            IDR2 = NINT(DYUSBN(NB))
            READ(1) (SCORED(J), J = 1, NXBIN(NB)*NYBIN(NB)*NZBIN(NB))
            WRITE(2,102) MB(NB), TITUSB(NB), IDUSBN(NB), NXBIN(NB),
     &         IR1A, IR1B, IDR1, IR2A, IR2B, IDR2, ZLOW(NB), ZHIGH(NB),
     &         NZBIN(NB), DZUSBN(NB)
            J = 0
            IR1 = IR1A
            IR2 = IR2A
            UVAR = ZLOW(NB)
            DO 3 IZ = 1, NZBIN(NB)
               DO 4 IY = 1, NYBIN(NB)
                  DO 5 IX = 1, NXBIN(NB)
                     J = J + 1
                     WRITE(2,1020) IR1, IR1 + IDR1, IR2, IR2 + IDR2,
     &                              UVAR, UVAR + DZUSBN(NB), SCORED(J)
                     IR1 = IR1 + IDR1
 5                CONTINUE
                  IR1 = IR1A
                  IR2 = IR2 + IDR2
                  WRITE(2,*)
 4             CONTINUE
               IR2 = IR2A
               UVAR = UVAR + DZUSBN(NB)
               WRITE(2,*)
 3          CONTINUE
         ELSE IF ((ITUHLP.EQ.1.OR.ITUHLP.EQ.7).AND.NYBIN(NB).LT.2) THEN
*           R-Z binning
*           XAXUSB(NB) = YLOW (NB)
*           YAXUSB(NB) = YHIGH(NB)
            NBIN = NXBIN(NB) * NZBIN(NB)
            READ(1) (SCORED(J), J = 1, NBIN)
            WRITE(2,103)  MB(NB),     TITUSB(NB), IDUSBN(NB), XLOW(NB),
     &         XHIGH(NB), NXBIN(NB),  DXUSBN(NB), ZLOW(NB),  ZHIGH(NB),
     &         NZBIN(NB), DZUSBN(NB)      !, XAXUSB(NB), YAXUSB(NB)
            J = 0
            RR = XLOW(NB)
            ZZ = ZLOW(NB)
            DO 6 IZ = 1, NZBIN(NB)
               DO 7 IX = 1, NXBIN(NB)
                  J = J + 1
                  WRITE(2,1030) RR, RR+DXUSBN(NB), ZZ, ZZ+DZUSBN(NB),
     &                          SCORED(J)
                  RR = RR + DXUSBN(NB)
 7             CONTINUE
               RR = XLOW(NB)
               ZZ = ZZ + DZUSBN(NB)
               WRITE(2,*)
 6          CONTINUE
         ELSE IF ( ITUHLP.EQ.1.OR.ITUHLP.EQ.7 ) THEN
*           R-Phi-Z binning
            NBIN = NXBIN(NB) * NYBIN(NB) * NZBIN(NB)
            READ(1) (SCORED(J), J = 1, NBIN)
             WRITE(2,104)  MB(NB),     TITUSB(NB), IDUSBN(NB), XLOW(NB),
     &         XHIGH(NB), NXBIN(NB),  DXUSBN(NB), YLOW(NB),  YHIGH(NB),
     &         NYBIN(NB), DYUSBN(NB), ZLOW(NB),   ZHIGH(NB), NZBIN(NB),
     &         DZUSBN(NB)
            J = 0
            RR = XLOW(NB)
            PH = YLOW(NB)
            ZZ = ZLOW(NB)
```

```
            DO 8 IZ = 1, NZBIN(NB)
               DO 9 IY = 1, NYBIN(NB)
                  DO 10 IX = 1, NXBIN(NB)
                     J = J + 1
                     WRITE(2,1040) RR, RR + DXUSBN(NB), PH,
     &                   PH + DYUSBN(NB), ZZ, ZZ + DZUSBN(NB), SCORED(J)
                     RR = RR + DXUSBN(NB)
 10               CONTINUE
                  RR = XLOW(NB)
                  PH = PH + DYUSBN(NB)
                  WRITE(2,*)
 9             CONTINUE
               PH = YLOW(NB)
               ZZ = ZZ + DZUSBN(NB)
               WRITE(2,*)
 8          CONTINUE
         ELSE IF ( ITUHLP .EQ. 0 ) THEN
*           Cartesian binning
            NBIN = NXBIN(NB) * NYBIN(NB) * NZBIN(NB)
            READ(1) (SCORED(J), J = 1, NBIN)
            WRITE(2,105)  MB(NB),     TITUSB(NB), IDUSBN(NB), XLOW(NB),
     &         XHIGH(NB), NXBIN(NB),  DXUSBN(NB), YLOW(NB),  YHIGH(NB),
     &         NYBIN(NB), DYUSBN(NB), ZLOW(NB),   ZHIGH(NB), NZBIN(NB),
     &         DZUSBN(NB)
            J = 0
            XX = XLOW(NB)
            YY = YLOW(NB)
            ZZ = ZLOW(NB)
            DO 11 IZ = 1, NZBIN(NB)
               DO 12 IY = 1, NYBIN(NB)
                  DO 13 IX = 1, NXBIN(NB)
                     J = J + 1
                     WRITE(2,1040) XX, XX + DXUSBN(NB), YY,
     &                   YY + DYUSBN(NB), ZZ, ZZ + DZUSBN(NB), SCORED(J)
                     XX = XX + DXUSBN(NB)
 13               CONTINUE
                  XX = XLOW(NB)
                  YY = YY + DYUSBN(NB)
                  WRITE(2,*)
 12            CONTINUE
               YY = YLOW(NB)
               ZZ = ZZ + DZUSBN(NB)
               WRITE(2,*)
 11         CONTINUE
         END IF
 1    CONTINUE               ! end loop on detectors
*     ---------------------------------------------------------------
 1000 CONTINUE

 100  FORMAT(/,1X,'*****',2X,A80,2X,'*****',/,/,10X,A32,/,/,
     &        10X,'Total number of particles followed ',I9,', for a ',
     &        'total weight of ',1P,E15.8,/)
 101  FORMAT (/, 3X, 'Region    binning n. ', I3, '  "', A10,
     &  '" , generalised particle n. ', I4, /, 6X, I5,
     &  ' bins corresponding to the region sets:' /, 6X, 'from region ',
     &  I5, ' to region ', I5, ' in step of ', I5, ' regions, or', /,
     &  6X, 'from region ', I5, ' to region ', I5, ' in step of ', I5,
     &  ' regions, or', /, 6X, 'from region ', I5, ' to region ', I5,
     &  ' in step of ', I5, ' regions',//, 7X, 'Set 1', 5X, 'Set 2',
     &  5X, 'Set 3', 6X, 'Score',/)
 102  FORMAT (/, 3X, 'Reg/Lat/U binning n. ', I3, '  "', A10,
     &  '" , generalised particle n. ',I4, /, 6X, I5,
```

```
     &  ' bins corresponding to the region-related set:' /, 6X,
     &  'from region ', I5, ' to region ', I5, ' in step of ', I5,
     &  ' regions, and', /, 6X, I5,
     &  ' bins corresponding to the lattice-related set:' /, 6X,
     &  'from lattice', I5, ' to lattice', I5, ' in step of ', I5,
     &  ' lattices, and', /, 6X, 'U coordinate: from ', 1P, E11.4,
     &  ' to ', E11.4, ' ux, ', 0P, I5, ' bins (', 1P, E11.4,
     &  ' ux wide)', /)
  103  FORMAT('1', /, 3X, 'R - Z     binning n. ', I3, '  "', A10,
     &  '" , generalised particle n. ', I4,/, 6X, 'R coordinate: from ',
     &  1P, E11.4, ' to ', E11.4, ' cm, ', 0P, I5, ' bins (', 1P, E11.4,
     &  ' cm wide)', /, 6X, 'Z coordinate: from ', 1P, E11.4, ' to ',
     &  E11.4, ' cm,  ', 0P, I5, ' bins (', 1P, E11.4, ' cm wide)', //,
     &  ' between R1',2X, 'and', 5X, 'R2', 13X, 'between Z1', 2X, 'and',
     &  5X, 'Z2', 15X, 'Score', /)
*    &  6X, 'axis coordinates: X =', 1P, E11.4, ', Y = ', E11.4, ' cm'/
  104  FORMAT('1', /, 3X, 'R-Phi-Z   binning n. ', I3, '  "', A10,
     &  '" , generalised particle n. ', I4,/, 6X,
     &  'R coordinate:   from ',1P, E11.4, ' to ', E11.4, ' cm, ', 0P,
     &  I5, ' bins (', 1P, E11.4, ' cm wide)', /, 6X,
     &  'Phi coordinate: from ', 1P, E11.4, ' to ', E11.4, ' rad, ', 0P,
     &  I5, ' bins (', 1P, E11.4, ' rad wide)', /, 6X,
     &  'Z coordinate:   from ', 1P, E11.4, '  to  ', E11.4, ' cm,', 0P,
     &  I5, ' bins (', 1P, E11.4, ' cm  wide)', //, ' between R1',1X,
     &  'and', 4X, 'R2', 11X, 'between Phi1', 1X, 'and', 2X, 'Phi2',
     &  10X, 'between Z1', 1X, 'and', 4X, 'Z2', 13X, 'Score', /)
  105  FORMAT('1', /, 3X, 'Cartesian binning n. ', I3, '  "', A10,
     &  '" , generalised particle n. ', I4,/, 6X, 'X coordinate: from ',
     &  1P, E11.4, ' to ', E11.4, ' cm, ', 0P, I5, ' bins (', 1P, E11.4,
     &  ' cm wide)', /, 6X, 'Y coordinate: from ', 1P, E11.4, ' to ',
     &  E11.4, ' cm, ', 0P, I5, ' bins (', 1P, E11.4, ' cm wide)',/, 6X,
     &  'Z coordinate: from ', 1P, E11.4, ' to ', E11.4, ' cm, ', 0P,I5,
     &  ' bins (', 1P, E11.4, ' cm wide)' //, ' between X1',1X,
     &  'and', 4X, 'X2', 12X, 'between Y1', 1X, 'and', 3X, 'Y2',
     &  12X, 'between Z1', 1X, 'and', 4X, 'Z2', 13X, 'Score', /)
  1010 FORMAT(3I10, 5X, 1P, E11.4)
  1020 FORMAT(I11, 4X, I11, 5X, I11, 4X, I11, 5X, 1P, E11.4, 4X, E11.4,
     &       6X, E11.4)
  1030 FORMAT(1P, E11.4, 5X, E11.4, 6X, E11.4, 5X, E11.4, 8X, E11.4)
  1040 FORMAT(1P, E11.4, 4X, E11.4, 5X, E11.4, 4X, E11.4, 5X, E11.4, 4X,
     &          E11.4, 6X, E11.4)
       END
```

Example:

```
*...+....1....+....2....+....3....+....4....+....5....+....6....+....7...+...8
USRBIN          10.0       3.0     -25.0       7.0       7.0     12.1 verythin
USRBIN          -7.0      -7.0      12.0      35.0      35.0      1.0 &
*  Cartesian binning of electron track-length density, to be written
*  unformatted on unit 25. Mesh is 35 bins between x = -7 and x = 7, 35 bins
*  between y = -7 and y = 7, and 1 bin between z = 12 and z = 12.1
```

7.78 USRCOLL

Defines a detector for a collision fluence estimator

See also USRBDX, USRBIN, USRTRACK

The full definition of the detector may require two successive cards. The second card, identified by the character "&" in any column from 71 to 78 (or in the last field in case of free format input), must be given unless the corresponding defaults are acceptable to the user.

First card:

WHAT(1) = 1.0: linear binning in energy
= -1.0: logarithmic binning in energy
Default = 1.0 (linear binning)

WHAT(2) = (generalised) particle type to be scored
Default = 201.0 (all particles)

WHAT(3) = logical output unit:
> 0.0: formatted data are written on WHAT(3) unit
< 0.0: unformatted data are written on |WHAT(3)| unit
Values of |WHAT(3)| < 21.0 should be avoided (with the exception of +11).
Default = 11.0 (standard output unit)

WHAT(4) = region defining the detector
Default = 1.0

WHAT(5) = volume of the detector in cm^3
Default = 1.0

WHAT(6) = number of energy bins
Default = 10.0

SDUM = any character string (not containing "&") identifying the detector (max. 10 characters)

Continuation card:

WHAT(1) = maximum kinetic energy for scoring (GeV)
Default = beam particle total energy as set by the BEAM (p. 64) option (if no BEAM card is given, the energy corresponding to 200 GeV/c momentum will be used)

WHAT(2) = minimum kinetic energy for scoring (GeV)
Default = 0.0 if linear energy binning, 0.001 GeV otherwise

WHAT(3) – **WHAT(6)**: not used

SDUM = "&" in any position in column 71 to 78 (or in the last field if free format is used)

Default (option USRCOLL not given): no fluence collision estimator

Notes

1. **IMPORTANT!** the results of a USRCOLL collision estimator are always given as *differential* distributions of fluence in energy, in units of cm^{-2} GeV^{-1} per incident primary. Thus, for example, when requesting a fluence energy spectrum, to obtain *integral binned* results (fluence in cm^{-2} *per energy bin* per primary) one must multiply the value of each energy bin by the width of the bin (even for logarithmic binning).
2. If the generalised particle is `208.0` or `211.0`, the quantity scored is differential energy fluence, expressed in GeV per cm^2 per energy unit per primary. That can sometimes lead to confusion since GeV cm^{-2} GeV^{-1} = cm^{-2}, where energy does not appear. Note that integrating over energy one gets GeV/cm^2.
3. The maximum number of collision + track-length detectors (see option USRTRACK, p. 231) that the user can define is 400. This value can be changed by modifying the parameter `MXUSTC` in member `USRTRC` of the `flukaadd` library or directory and then re-compiling and linking FLUKA.
4. The logical output unit for the estimator results (WHAT(3) of the first USRCOLL card) can be any one of the following:
   - the standard output unit `11`: estimator results will be written on the same file as the standard FLUKA output.
   - a pre-connected unit (via a symbolic link on most UNIX systems, `ASSIGN` under VMS, or equivalent commands on other systems)
   - a file opened with the FLUKA command OPEN
   - a file opened with a Fortran `OPEN` statement in a user-written initialisation routine such as `USRINI` or `SOURCE` (see Chap. 12.2.26, 12.2.19)
   - a dynamically opened file, with a default name assigned by the Fortran compiler (typically `fort.xx` or `ftn.xx`, with `xx` equal to the chosen logical output unit number).

   The results of several USRCOLL and USRTRACK detectors in a same FLUKA run can be written on the same file, but of course only if they are all in the same mode (all formatted, or all unformatted).
   It is also possible in principle to write on the same file the results of different kinds of estimators (USRBDX, USRBIN, etc.) but this is not recommended, especially in the case of an unformatted file, because it would make very difficult any reading and analysis.
5. When scoring neutron fluence, and the requested energy bin structure overlaps with that of the low-energy neutron groups, bin boundaries are forced to coincide with group boundaries and no bin can be smaller than the corresponding group.
   Actually, the program uses the requested energy limits and number of bins to estimate the desired bin width. The number of bins above the upper limit of the first low-energy neutron group is recalculated according to such width.
   Note that the energy limits of the thermal neutron group are 10^{-14} GeV (10^{-5} eV) and 4.14×10^{-10} GeV (0.414 eV) for the ENEA data sets. All group energy boundaries are listed in Table 10.1 on p. 294.
6. An example on how to read USRCOLL unformatted output is given in Note 6 to option USRTRACK on p. 232 (the two options produce output with identical format). An explanation of the meaning of the different variables is given in the comments at the beginning of the program.
   A more complex program `USTSUW`, which allows to compute also standard deviations over several runs and cumulative distributions, is available with the normal FLUKA code distribution in directory `$FLUPRO/flutil`.

Example:

```
*...+....1....+....2....+....3....+....4....+....5....+....6....+....7...+...8
USRCOLL         -1.0       8.0      23.0      15.0     540.0     350. NeutFlu
USRCOLL        250.0    1.E-14       0.0       0.0       0.0       0.   &
*   Calculate neutron fluence spectrum in region 15 from thermal energies to
*   250 GeV, in 350 logarithmic energy intervals. Write formatted results on
*   unit 23. The volume of region 15 is 540 cm3.
```

7.79 USRICALL

Calls user-dependent initialisation.

See also USROCALL

The meaning of **WHAT(1)** ... **WHAT(6)**, **SDUM** is defined by the user.
A call to the user-written routine USRINI with 6 WHAT numerical values and one character string SDUM as arguments is issued every time this card is read.

Default (option USRICALL not given): no user initialisation

Notes

1. In subroutine USRINI, WHAT and SDUM must be declared as follows:
   ```
   DOUBLE PRECISION WHAT (6)
   CHARACTER SDUM*8
   ```
2. A description of routine USRINI and instructions about its use are given in 12.2.26.
3. It is suggested that the USRINI routine shall contain at least the three standard INCLUDE files:
   DBLPRC, DIMPAR, IOUNIT
   Other useful files to be INCLUDEd, depending on the problem, can be BEAMCM, CASLIM, SOURCM, SUMCOU, FLKMAT, PAPROP, SCOHLP, USRBDX, USRBIN, USRSNC, USRTRC, USRYLD. See 12.1.1 for more information on INCLUDE files useful in user routines.

Example:

```
*...+....1....+....2....+....3....+....4....+....5....+....6....+....7...+...8
USRICALL        123.      456.       1.0      -2.0      18.0      18. FLAG12
*  Call initialisation routine passing over 6 numerical values and a string
```

7.80 USROCALL

Calls user-dependent output.

See also USRICALL

The meaning of **WHAT(1)** ... **WHAT(6)**, **SDUM** is defined by the user.
A call to the user-written routine `USROUT` with 6 WHAT numerical values and one character string SDUM as arguments is issued every time this card is read.

Default (option USROCALL not given): no user-defined output

Notes

1. In subroutine `USROUT`, WHAT and SDUM must be declared as follows:

```
DOUBLE PRECISION WHAT (6)
CHARACTER SDUM*8
```

2. It is suggested that the `USROUT` routine shall contain at least the three standard `INCLUDE` files:
   `DBLPRC, DIMPAR, IOUNIT`
   Other useful files to be `INCLUDE`d, depending on the problem, can be:
   `BEAMCM` (characteristics of beam particles)
   `CASLIM, SOURCM, SUMCOU` (normalisation factors)
   `FLKMAT` (info on materials)
   `PAPROP` (particle properties)
   `SCOHLP` (scoring help: flags identifying the different estimators and binnings)
   `USRBDX, USRBIN, USRSNC, USRTRC, USRYLD` (info about different estimators and binnings)
   See 12.1.1 for more information on `INCLUDE` files useful in user routines.
3. The USROCALL card must occupy a meaningful position in the input stream. For example, a typical position could be just after the START card and before the STOP card.

Example:

```
*...+....1....+....2....+....3....+....4....+....5....+....6....+....7...+...8
USROCALL        17.0      17.0      -5.5       1.1     654.0     321. OK
*  Call output routine passing over 6 numerical values and a string
```

7.81 USRTRACK

Defines a detector for a track-length fluence estimator.

See also USRBDX, USRBIN, USRCOLL

The full definition of the detector may require two successive cards (the second card, identified by the character "&" in any column from 71 to 78 (or in the last field in case of free format input), must be given unless the corresponding defaults are acceptable to the user)

First card:

WHAT(1) = 1.0: linear binning in energy
= -1.0: logarithmic binning in energy
Default = 1.0 (linear binning)

WHAT(2) = (generalised) particle type to be scored
Default = 201.0 (all particles)

WHAT(3) = logical output unit:
> 0.0: formatted data are written on WHAT(3) unit
< 0.0: unformatted data are written on |WHAT(3)| unit
Values of |WHAT(3)| < 21.0 should be avoided (with the exception of +11).
Default = 11.0 (standard output unit)

WHAT(4) = region defining the detector
Default = 1.0

WHAT(5) = volume of the detector in cm^3
Default = 1.0

WHAT(6) = number of energy bins
Default = 10.0

SDUM = any character string (not containing "&") identifying the detector (max. 10 characters)

Continuation card:

WHAT(1) = maximum kinetic energy for scoring (GeV)
Default = beam particle total energy as set by the BEAM (p. 64) option (if no BEAM card is given, the energy corresponding to 200 GeV/c momentum will be used)

WHAT(2) = minimum kinetic energy for scoring (GeV)
Default = 0.0 if linear energy binning, 0.001 GeV otherwise

WHAT(3) – **WHAT(6)**: not used

SDUM = "&" in any position in column 71 to 78 (or in the last field if free format is used)

Default (option USRTRACK not given): no tracklength fluence estimator

Notes

1. **IMPORTANT!** The results of a USRTRACK track-length estimator are always given as *differential* distributions of fluence in energy, in units of cm^{-2} GeV^{-1} per incident primary. Thus, for example, when requesting a fluence energy spectrum, to obtain *integral binned* results (fluence in cm^{-2} *per energy bin* per primary) one must multiply the value of each energy bin by the width of the bin (even for logarithmic binning).
2. If the generalised particle is `208.0` or `211.0`, the quantity scored is differential energy fluence, expressed in GeV per cm^2 per energy unit per primary. That can sometimes lead to confusion since GeV cm^{-2} GeV^{-1} = cm^{-2}, where energy does not appear. Note that integrating over energy one gets GeV/cm^2.
3. The maximum number of track-length + collision detectors (see option USRCOLL, p. 227) that the user can define is 400. This value can be changed by modifying the parameter `MXUSTC` in member `USRTRC` of the `flukaadd` library or directory and then re-compiling and linking FLUKA.
4. The logical output unit for the estimator results (WHAT(3) of the first USRTRACK card) can be any one of the following:
   - the standard output unit `11`: estimator results will be written on the same file as the standard FLUKA output.
   - a pre-connected unit (via a symbolic link on most UNIX systems, `ASSIGN` under VMS, or equivalent commands on other systems)
   - a file opened with the FLUKA command OPEN
   - a file opened with a Fortran `OPEN` statement in a user-written initialisation routine such as `USRINI` or `SOURCE` (see Chap. 12.2.26, 12.2.19)
   - a dynamically opened file, with a default name assigned by the Fortran compiler (typically `fort.xx` or `ftn.xx`, with `xx` equal to the chosen logical output unit number).

   The results of several USRTRACK and USRCOLL detectors in a same FLUKA run can be written on the same file, but of course only if they are all in the same mode (all formatted, or all unformatted).
   It is also possible in principle to write on the same file the results of different kinds of estimators (USRBDX, USRBIN, etc.) but this is not recommended, especially in the case of an unformatted file, because it would make very difficult any reading and analysis.
5. When scoring neutron fluence, and the requested energy bin structure overlaps with that of the low-energy neutron groups, bin boundaries are forced to coincide with group boundaries and no bin can be smaller than the corresponding group.
   Actually, the program uses the requested energy limits and number of bins to estimate the desired bin width. The number of bins above the upper limit of the first low-energy neutron group is recalculated according to such width.
   Note that the energy limits of the thermal neutron group are 10^{-14} GeV (10^{-5} eV) and 4.14×10^{-10} GeV (0.414 eV) for the ENEA data sets. All group energy boundaries are listed in Table 10.1 on p. 294.
6. An example on how to read USRTRACK unformatted output is given below. An explanation of the meaning of the different variables is given in the comments at the beginning of the program. The program lists the bin energy boundaries (in increasing order) and the corresponding differential and cumulative (integral) fluence.
   A more complex program `USTSUW`, which allows to compute also standard deviations over several runs, is available with the normal FLUKA code distribution in directory `$FLUPRO/flutil`.

```
      PROGRAM RDTRK
*----------------------------------------------------------------------
*  Up to MXUSTC user-defined track or collision detectors are allowed
*
*    DETCBN = energy bin width (GeV) if linear in energy or referring
*             to a low-energy neutron energy group, otherwise
*             ratio between upper and lower edge of energy intervals
*    ETCHGH = maximum energy (GeV) (WHAT(1) of 2nd card)
*    ETCLOW = minimum energy (GeV) (WHAT(2) of 2nd card)
*    ELIMIT = lowest energy of regular intervals
*    ENGMAX = upper energies (GeV) of the neutron groups
*    IDUSTC = (generalised) particle scored (WHAT(2) of first card)
*    IGMUTC = maximum low-energy neutron group to be scored
*    ITUSTC = type of detector
*              1 = linear energy, tracklength
*              2 = linear energy, collision
*             -1 = logarithmic energy, tracklength
```

```
*             -2 = logarithmic energy, collision
*    LLNUTC = no low-energy neutron scoring if false, yes if true
*    NTC    = id-number of the tracklength/collision detector
*    NCASE  = number of beam particles followed
*    NETCBN = number of energy intervals
*    NRUSTC = region of the detector (WHAT(4) of first card)
*    RUNTIT = title of the run (as given by card TITLE)
*    SCORED = result array
*    TITUTC = detector name (SDUM in first USRTRACK/USRCOLL card)
*    VUSRTC = volume (cm**3) of the detector (WHAT(5) of first card)
*    WEIPRI = total weight of primaries
*    NHIGH  = number of energy bins above low-energy neutron limit
*
*----------------------------------------------------------------------
      PARAMETER ( MXUSTC = 1500 )   ! max. number of trk/coll detectors
      PARAMETER ( MXSCOR = 1000 )   ! storage for results
      PARAMETER ( NMXGRP = 100 )    ! # of low-energy neutron groups

      LOGICAL LLNUTC
      CHARACTER RUNTIT*80, RUNTIM*32, TITUTC*10, FILNAM*80
      DOUBLE PRECISION CUMUL, EN1, EN2

      DIMENSION ETCLOW(MXUSTC), ETCHGH(MXUSTC), DETCBN(MXUSTC),
     &        NETCBN(MXUSTC), NRUSTC(MXUSTC), ITUSTC(MXUSTC),
     &        IDUSTC(MXUSTC), IGMUTC(MXUSTC), LLNUTC(MXUSTC),
     &        TITUTC(MXUSTC), VUSRTC(MXUSTC), ENGMAX(NMXGRP+1),
     &        SCORED(MXSCOR), NTC(MXUSTC)

      WRITE(*,*)' Type the name of the input file:'
      READ (*,'(A)') FILNAM
      LQ = INDEX(FILNAM,' ') - 1
      OPEN (UNIT=1, FILE=FILNAM, STATUS='OLD', FORM='UNFORMATTED')
      OPEN (UNIT=2, FILE=FILNAM(1:LQ)//'.txt', STATUS='NEW')
*----------- read and write 1st record ---------------------------------
      READ (1) RUNTIT, RUNTIM, WEIPRI, NCASE
      WRITE(2,100) RUNTIT, RUNTIM, NCASE, WEIPRI
*----------- loop on tracklength detector data in the present file -----
      DO 1 IX = 1, MXUSTC
         NX = IX
*        ---------------- read and write 2nd record --------------------
         READ (1, END=1000) NTC(NX), TITUTC(NX), ITUSTC(NX), IDUSTC(NX),
     &       NRUSTC(NX), VUSRTC(NX), LLNUTC(NX), ETCLOW(NX), ETCHGH(NX),
     &       NETCBN(NX), DETCBN(NX)
*        ------------ tracklength or collision estimator ---------------
         IF(ABS(ITUSTC(NX)) .EQ. 1) THEN
            WRITE(2,101) NTC(NX),
     &                TITUTC(NX), IDUSTC(NX), NRUSTC(NX), VUSRTC(NX)
         ELSE
            WRITE(2,106) NTC(NX),
     &                TITUTC(NX), IDUSTC(NX), NRUSTC(NX), VUSRTC(NX)
         END IF
*        ------------ if low-en. neutrons, read group energies ---------
         IF ( LLNUTC (NX) ) THEN
            READ (1) IGMUTC(NX), (ENGMAX(IG), IG = 1, IGMUTC(NX)+1)
            WRITE (2,102) IGMUTC(NX)
         ELSE
            IGMUTC(NX) = 0
         END IF
*        -------------------- linear or log in energy ------------------
         IF ( ITUSTC (NX) .GT. 0 ) THEN
            WRITE (2,103) ETCLOW(NX), ETCHGH(NX), NETCBN(NX), DETCBN(NX)
         ELSE
```

```
            WRITE (2,104) ETCLOW(NX), ETCHGH(NX), NETCBN(NX), DETCBN(NX)
         END IF
*        interv = total number of energy intervals
*        (intervals above the limit for n-groups + intervals below)
         INTERV = NETCBN(NX) + IGMUTC(NX)
*------------ read the scoring results as a 1-dimensional array --------
         READ (1) (SCORED(J), J = 1, INTERV)
         WRITE (2,'(34X,"Differential",8X,"Integral")')
         WRITE (2,'(36X,"Fluence",12X,"Fluence")')
         WRITE (2,'("Lower energy",4X,"Upper energy",5X,
     &              "cm**-2 GeV**-1",8X,"cm**-2"/)')
         CUMUL = 0.D0
         ELIMIT = ETCLOW(NX)
*        ---------------------- low-energy neutrons -------------------
         IF ( LLNUTC (NX) ) THEN
*           low-energy neutron data, if present, are stored backwards
            IG  = IGMUTC(NX)
            EN1 = ENGMAX(IG+1)
*           ---------------- loop on low-energy groups ----------------
*           (last groups are lowest energies, print them low to high)
            DO 2 JG = NETCBN(NX) + IGMUTC(NX), NETCBN(NX) + 1, -1
              EN2 = ENGMAX(IG)
              CUMUL = CUMUL + (EN2 - EN1) * SCORED(JG)
              WRITE(2,105) EN1, EN2, SCORED(JG), CUMUL
              IG = IG - 1
              EN1 = EN2
 2          CONTINUE             ! end loop on low-energy groups
*           find lower limit of first bin above or straddling the
*           n-group limit. Nhigh: counts the high energy bins
            NHIGH = 0
*           set energy boundary at n-group limit
            ELIMIT = EN1
            EN1 = ETCHGH(NX)
            DO 3 IE = 1, NETCBN(NX)
               IF(ITUSTC(NX) .GT. 0) THEN
                  EN2 = EN1 - DETCBN(NX)
               ELSE
                  EN2 = EN1 / DETCBN(NX)
               END IF
               EN1 = EN2
               NHIGH = NHIGH + 1
               IF(EN1 .LE. ELIMIT) GO TO 4
 3          CONTINUE
 4          CONTINUE
         ELSE
            EN1   = ETCLOW(NX)
            NHIGH = NETCBN(NX)
         END IF
*        ---------------------------------------------------------------
*        first bin above or straddling the n-group limit
         IF(ITUSTC(NX) .GT. 0) THEN
            EN2 = EN1 + DETCBN(NX)
         ELSE
            EN2 = EN1 * DETCBN(NX)
         END IF
         CUMUL = CUMUL + (EN2 - ELIMIT) * SCORED(NETCBN(NX) - NHIGH + 1)
         WRITE(2,105) ELIMIT, EN2,
     &                SCORED(NETCBN(NX) - NHIGH + 1), CUMUL
         EN1 = EN2
*        ----------- loop on energies above the n-group limit ----------
         DO 5 IE = 2, NHIGH
            IF(ITUSTC(NX) .GT. 0) THEN
```

```
                  EN2 = EN1 + DETCBN(NX)
               ELSE
                  EN2 = EN1 * DETCBN(NX)
               END IF
               CUMUL = CUMUL + (EN2 - EN1)* SCORED(NETCBN(NX) - NHIGH + IE)
               WRITE(2,105) EN1, EN2,SCORED(NETCBN(NX) - NHIGH + IE), CUMUL
               EN1 = EN2
 5          CONTINUE            ! end loop on energies above limit
*             Case of generalised particles .ne. 8 but including neutrons
              IF (LLNUTC(NX) .AND. IDUSTC(NX) .NE. 8 .AND.
     &                             NETCBN(NX) .GT. NHIGH) THEN
                WRITE(2,110) ELIMIT
                EN1 = ETCLOW(NX)
                DO 7 IE = 1, NETCBN(NX) - NHIGH
                   IF(ITUSTC(NX) .GT. 0) THEN
                      EN2 = EN1 + DETCBN(NX)
                   ELSE
                      EN2 = EN1 * DETCBN(NX)
                   END IF
                   CUMUL=CUMUL+(EN2-EN1)*SCORED(NETCBN(NX)+IGMUTC(NX)+IE)
                   WRITE(2,105) EN1,EN2, SCORED(NETCBN(NX)+IGMUTC(NX)+IE),
     &                          CUMUL
                   EN1 = EN2
 7              CONTINUE
               END IF
*           ------------------------------------------------------------
 1      CONTINUE               ! end loop on detectors
*       ------------------------------------------------------------------
 1000 CONTINUE

 100  FORMAT(/,1X,'*****',2X,A80,2X,'*****',/,/,10X,A32,/,/,
     &        10X,'Total number of particles followed ',I9,', for a ',
     &        'total weight of ',1P,E15.8,/)
 101  FORMAT(/,3X,'Track n. ',I3,'  "',A10,
     &        '" , generalised particle n. ',I4,', region n. ',I6,
     &        /,6X,'detector volume: ',1P,E15.8,' cm**3')
 102  FORMAT(6X,'low-energy neutrons scored from group 1 to group ', I5)
 103  FORMAT(6X,'linear energy binning from ',1P,E11.4,' to ',
     &        E11.4,' GeV, ',0P,I5,' bins (',1P,E11.4,' GeV wide)'/)
 104  FORMAT (6X,'logar. energy binning from ',1P,E11.4,' to ',
     &        E11.4,' GeV, ',0P,I5,' bins (ratio :',1P,E11.4,')'/)
 105  FORMAT(1P,2(E11.4,5X),2(E14.7,5X))
 106  FORMAT(/,3X,'Coll. n. ',I3,'  "',A10,
     &        '" , generalised particle n. ',I4,', region n. ',I6,
     &        /,6X,'detector volume: ',1P,E15.8,' cm**3')
 110  FORMAT(/,20X,"Particles other than neutrons below E = ",1P,E11.4/)
      END
```

Example:

```
*...+....1....+....2....+....3....+....4....+....5....+....6....+....7...+...8
USRTRACK         1.0       7.0     -24.0      16.0    4500.0      150.PhotFlu
USRTRACK         1.5       0.0       0.0       0.0       0.0       0.   &
*   Calculate photon fluence spectrum in region 16 from 0 to 1.5 GeV, in
*   150 linear energy intervals. Write unformatted results on
*   unit 24. The volume of region 16 is 4500 cm3.
```

7.82 USRYIELD

Defines a detector to score a double-differential particle yield around an extended or a point target

See also USRBDX
The full definition of the detector may require two successive cards (the second card, identified by the character "&" in any column from 71 to 78 (or in the last field in case of free format input), must be given unless the corresponding defaults are acceptable to the user)

First card:

For SDUM = anything but BEAMDEF:

WHAT(1) = $i_e + i_a \times 100$, where i_e and i_a indicate the two physical quantities with respect to which the double-differential yield is calculated.
If $i_e > 0$, the yield will be analysed in linear intervals with respect to the first quantity; if $i_e < 0$, the yield distribution will be binned logarithmically.
(Note that for rapidity, pseudorapidity and Feynman-x logarithmic intervals are not available and will be forced to linear if requested).
For the second quantity, indicated by i_a, only one interval will be considered.

$|i_e|$ or $|i_a|$ = 1 : kinetic energy in GeV
= 2 : total momentum in GeV/c
= 3 : rapidity in the lab frame *(only linear scoring available)*
= 4 : rapidity in the c.m.s. frame *(only linear scoring available)*
= 5 : pseudorapidity in the lab frame *(only linear scoring available)*
= 6 : pseudorapidity in the c.m.s. frame *(only linear scoring available)*
= 7 : Feynman-x in the lab frame (E/E_{beam}) *(only linear scoring available)*
= 8 : Feynman-x in the c.m.s. frame *(only linear scoring available)*
= 9 : transverse momentum in GeV/c
= 10 : transverse mass in GeV
= 11 : longitudinal momentum in the lab frame (in GeV/c)
= 12 : longitudinal momentum in the c.m.s. frame (in GeV/c)
= 13 : total energy in GeV
= 14 : polar angle in the lab frame (see Note 6)
= 15 : polar angle in the c.m.s. frame (see Note 6)
= 16 : square transverse momentum in $(\text{GeV/c})^2$
= 17 : weighted angle in the lab frame (see Note 6)
= 18 : weighted transverse momentum in GeV/c (see Note 6)
= 19 : ratio laboratory momentum/beam momentum
= 20 : transverse kinetic energy
= 21 : excitation energy
= 22 : particle charge
= 23 : particle LET
= 24 : like 14, but with input in degrees rather than in radians (see Note 6)
= 25 : like 15, but with input in degrees rather than in radians (see Note 6)
= 26 : laboratory kinetic energy/nucleon
= 27 : laboratory momentum/nucleon

WHAT(2) > 0.0: (generalised) particle type to be scored.
< -80.0 and WHAT(4) = -1.0 and WHAT(5) = -2: the (generalised) particles of type IJ *entering* an inelastic interaction are scored by setting WHAT(2) = -100 -IJ
Default = 201.0 (all particles)

WHAT(3) = logical output unit:
> 0.0: formatted data are written on WHAT(3) unit

$<$ 0.0: unformatted data are written on |WHAT(3)| unit
Values of |WHAT(3)| $<$ 21.0 should be avoided (with the exception of +11).
Default = 11.0 (standard output unit)

WHAT(4) $>$ 0.0: first region defining the boundary (upstream region)
= -1.0 and WHAT(5) = -2.0: the yield of particles *emerging* from inelastic interactions is scored
Default = 1.0

WHAT(5) $>$ 0.0: second region defining the boundary (downstream region)
= -2.0 and WHAT(4) = -1.0: the yield of particles *emerging* from inelastic interactions is scored
Default = 2.0

WHAT(6) = normalisation factor (the results will be divided by WHAT(6))

SDUM = any character string (not containing "&") identifying the yield detector (max. 10 characters)

Continuation card:

WHAT(1) = Upper limit of the scoring interval for the first quantity
Default = beam energy value

WHAT(2) = Lower limit of the scoring interval for the first quantity
Default = 0.0 if linear binning, 1.0 otherwise. Note that these values might not be meaningful for all available quantities.

WHAT(3) = number of scoring intervals for the first quantity
Default = 50.0

WHAT(4) = Upper scoring limit for the second quantity
Default : no default!

WHAT(5) = Lower scoring limit for the second quantity
Default = 0.0

WHAT(6) = $ix_a + ix_m \times 100$, where ix_a indicates the kind of yield or cross-section desired and ix_m the target material (if needed in order to calculate a cross-section, otherwise $ix_m = 0$). See Note 4 in case of a thick target

ix_a = 1: plain double-differential cross-section $\frac{\mathrm{d}^2\sigma}{\mathrm{d}x_1\mathrm{d}x_2}$, where x_1, x_2 are the first and second quantity
ix_a = 2: invariant cross-section $E\frac{\mathrm{d}^3\sigma}{\mathrm{d}p^3}$
ix_a = 3: plain double differential yield $\frac{\mathrm{d}^2N}{\mathrm{d}x_1\mathrm{d}x_2}$, where x_1, x_2 are the first and second quantity
ix_a = 4: double differential yield $\frac{\mathrm{d}^2(x_2N)}{\mathrm{d}x_1\mathrm{d}x_2}$ where x_1, x_2 are the first and second quantity
ix_a = 5: double differential yield $\frac{\mathrm{d}^2(x_1N)}{\mathrm{d}x_1\mathrm{d}x_2}$, where x_1, x_2 are the first and second quantity
ix_a = 6: double differential fluence yield $\frac{1}{\cos\theta}\frac{\mathrm{d}^2N}{\mathrm{d}x_1\mathrm{d}x_2}$ where x_1, x_2 are the first and second quantity, and θ is the angle between the particle direction and the normal to the surface
ix_m = material number of the target

Default = 0.0 (plain double-differential cross-section — but see Note 4)

SDUM = "&" in any position in column 71 to 78 (or in the last field if free format is used)

For SDUM = BEAMDEF:

WHAT(1) = (projectile particle index)

Default = `IJBEAM` (beam particle)

WHAT(2) = target particle index (used by the code to define the c.m.s. frame)

Default = `1.0` (proton)

WHAT(3) = projectile momentum

Default = `PBEAM` (beam momentum)

WHAT(4,5,6) = projectile direction cosines

Default = `UBEAM, VBEAM, WBEAM` (beam direction cosines)

Default (option USRYIELD not given): no yield estimator is defined

Notes

1. While option USRBDX (p. 211) calculates angular distributions *with respect to the normal* to the boundary at the point of crossing, USRYIELD distributions are calculated *with respect to a fixed direction* (the beam direction, or a different direction specified by the user with SDUM = BEAMDEF).
2. When scoring thick-target yields, the angle considered is that between the direction of the particle at the point where it crosses the target surface and the beam direction (or a different direction specified by the user, see Note 1). The target surface is defined as the boundary between two regions (positive values of WHAT(4) and WHAT(5) of the first USRYIELD card).
3. Point-target yields, i.e., yields of particles emerging from inelastic interactions with single nuclei, are scored by setting WHAT(4) = `-1.0` and WHAT(5) = `-2.0` in the first USRYIELD card). As an alternative, the corresponding cross-sections can be calculated, depending on the value of WHAT(6). In addition, if WHAT(2) in the same card is $<$ `-80.0`, the distributions of particles *entering* the inelastic interactions can be scored.
4. Calculating a cross-section has little meaning in case of a thick target.
5. Differential yields (or cross-sections) are scored over any desired number of intervals for what concerns the first quantity, but over only one interval for the second quantity. However, the results are always expressed as second derivatives (or third derivatives in the case of invariant cross-sections), and *not* as interval-integrated yields. In order to obtain more intervals for the second quantity, the user must define further USRYIELD detectors.
6. In the case of polar angle quantities ($|i_e|$ or $|i_a| = 14, 15, 17, 18, 24, 25$) the differential yield is always referred to solid angle in steradian, although input is specified in radian or degrees.
7. A USRYIELD card with SDUM = BEAMDEF, if given, does not refer to a particular detector, but modifies the reference projectile or target parameters for all USRYIELD detectors of the current run. No continuation card has to be given after one with SDUM = BEAMDEF.
8. The logical output unit for the estimator results (WHAT(3) of the first USRYIELD card) can be any one of the following:
   - the standard output unit `11`: estimator results will be written on the same file as the standard FLUKA output.
   - a pre-connected unit (via a symbolic link on most UNIX systems, `ASSIGN` under VMS, or equivalent commands on other systems)
   - a file opened with the FLUKA command OPEN
   - a file opened with a Fortran `OPEN` statement in a user-written initialisation routine such as `USRINI` or `SOURCE` (see Chap. 12.2.26, 12.2.19)
   - a dynamically opened file, with a default name assigned by the Fortran compiler (typically `fort.xx` or `ftn.xx`, with `xx` equal to the chosen logical output unit number).

   The results of several USRYIELD detectors in a same FLUKA run can be written on the same file, but of course only if they are all in the same mode (all formatted, or all unformatted).
   It is also possible in principle to write on the same file the results of different kinds of estimators (USRBDX, USRBIN, etc.) but this is not recommended, especially in the case of an unformatted file, because it would make very difficult any reading and analysis.

9. Not all 27×27 combinations of quantities are accepted by the code, nor are they all meaningful (for instance one could run successfully by setting in the first USRYIELD card WHAT(1) with $i_a = i_e$, but the result would have no physical meaning).
10. An example on how to read USRYIELD unformatted output is given below. An explanation of the meaning of the different variables is given in the comments at the beginning of the program. The program lists the bin boundaries of the first variable ("energy", in increasing order), and the corresponding double-differential and cumulative yield (integrated over both variables ("energy" and "angle").
    A more complex program `USYSUW`, which allows to compute also standard deviations over several runs, is available with the normal FLUKA code distribution in directory `$FLUPRO/flutil`.

```
      PROGRAM RDYLD
*---------------------------------------------------------------------*
*  Up to MXUSYL user-defined yield detectors are allowed              *
*                                                                     *
*  AYLHGH = maximum angle (steradian) (WHAT(4) of 2nd card)           *
*  AYLLOW = minimum angle (steradian) (WHAT(5) of 2nd card)           *
*  DAYLBN = angle bin width                                           *
*  DEYLBN = energy bin width (GeV) if linear in energy or referring   *
*           to a low-energy neutron energy group, otherwise           *
*           ratio between upper and lower edge of energy intervals    *
*  EYLHGH = maximum energy (GeV) (WHAT(1) of 2nd card)                *
*  EYLLOW = minimum energy (GeV) (WHAT(2) of 2nd card: may be         *
*           re-defined if low-energy neutrons are scored              *
*  ENGMAX = upper energies (GeV) of the neutron groups                *
*  IDUSYL = (generalised) particle scored (WHAT(2) of first card)     *
*  IGMUYL = maximum low-energy neutron group to be scored             *
*  IJUSYL = projectile identity                                       *
*  JTUSYL = target identity                                           *
*  ITUSYL = type of binning = WHAT(1) of first card = ie + ia*100     *
*                   |ie| = 1 : kinetic  energy binning               *
*                   |ie| = 2 : total momentum  binning               *
*                   |ie| = 3 : lab.   rapidity binning               *
*                   |ie| = 4 : cms    rapidity binning               *
*                   |ie| = 5 : lab. pseudorap. binning               *
*                   |ie| = 6 : cms  pseudorap. binning               *
*                   |ie| = 7 : lab. x          binning               *
*                   |ie| = 8 : cms  Feynmann x binning               *
*                   |ie| = 9 : transverse mom. binning               *
*                   |ie| =10 : transverse mass binning               *
*                   |ie| =11 : lab. long. mom. binning               *
*                   |ie| =12 : cms  long. mom. binning               *
*                   |ie| =13 : total energy    binning               *
*                   |ie| =14 : lab. angle      binning               *
*                   |ie| =15 : cms  angle      binning               *
*                   |ie| =16 : p_t squared     binning               *
*                   |ie| =17 : lab. angle      binning               *
*                              with 1/2pi sin(theta) weight           *
*                   |ie| =18 : p_t             binning               *
*                              with 1/(2pi p_t) weight                *
*                   |ie| =19 : frac. lab mom.  binning               *
*                   |ie| =20 : trans. kin. en. binning               *
*                   |ie| =21 : excitation en.  binning               *
*                   ie > 0 --> linear, ie < 0 --> logarithmic         *
*                   ia has the same meaning but for the 2nd variable  *
*  IXUSYL = cross section kind, ixa + ixm * 100                       *
*  LLNUYL = no low-energy neutron scoring if false, yes if true       *
*  MY     = id-number of the yield detector                           *
*  NCASE  = number of beam particles followed                         *
*  NEYLBN = number of energy intervals (re-defined by the program if  *
*           low-energy neutrons are scored)                           *
*  NR1UYL = first region (WHAT(4) of first card)                      *
*  NR2UYL = second region (WHAT(5) of first card)                     *
*  PUSRYL = momentum of projectile to be used to define (possible)    *
```

```
*           Lorentz transformations, Feynman X etc.                    *
*  RUNTIM = date and time of the run (as printed on standard output)   *
*  RUNTIT = title of the run (as given by card TITLE)                  *
*  SCORED = result array                                               *
*  SGUSYL = adopted cross section (if any)                             *
*  SQSUYL = cms energy for Lorentz transformation                      *
*  TITUYL = detector name (SDUM in first USBDRX card)                  *
*  USNRYL = user normalisation factor                                  *
*  UUSRYL,VUSRYL,WUSRYL = laboratory projectile direction              *
*  WEIPRI = total weight of primaries                                  *
*  IAUSYL,IEUSYL = ausiliary arrays where itusyl is decoded            *
*  NHIGH  = number of energy bins above low-energy neutron limit       *
*  CUMUL  = energy-angle cumulative yield                              *
*----------------------------------------------------------------------*
      PARAMETER ( MXUSYL = 100 )    ! max. number of usryield detectors
      PARAMETER ( MXSCOR = 100000 ) ! storage for results
      PARAMETER ( NMXGRP = 100 )    ! max # of low-energy neutron groups
      PARAMETER ( PIPIPI = 3.141592653589793D+00 )
      PARAMETER ( HLFHLF = 0.5D+00 )
      PARAMETER ( ONEONE = 1.D+00 , TWOTWO = 2.D+00  )
      PARAMETER ( TWOPIP = 2.D+00 * PIPIPI )

      LOGICAL LLNUYL
      CHARACTER RUNTIT*80,RUNTIM*32,TITUYL*10,FILNAM*80
      DOUBLE PRECISION CUMUL, EN1, EN2

      DIMENSION EYLLOW(MXUSYL), EYLHGH(MXUSYL), AYLLOW(MXUSYL),
     &          AYLHGH(MXUSYL), DEYLBN(MXUSYL), NEYLBN(MXUSYL),
     &          NR1UYL(MXUSYL), NR2UYL(MXUSYL), USNRYL(MXUSYL),
     &          SGUSYL(MXUSYL), ITUSYL(MXUSYL), IAUSYL(MXUSYL),
     &          IDUSYL(MXUSYL), IEUSYL(MXUSYL), DAYLBN(MXUSYL),
     &          IGMUYL(MXUSYL), IXUSYL(MXUSYL), LLNUYL(MXUSYL),
     &          TITUYL(MXUSYL), ENGMAX (NMXGRP+1), SCORED(MXSCOR),
     &          MY(MXUSYL), NHIGH(MXUSYL)

      WRITE(*,*)' Type the name of the input file:'
      READ (*,'(A)') FILNAM
      LQ = INDEX(FILNAM,' ') - 1
      OPEN (UNIT=1, FILE=FILNAM, STATUS='OLD', FORM='UNFORMATTED')
      OPEN (UNIT=2, FILE=FILNAM(1:LQ)//'.txt', STATUS='NEW')
*----------- read and write 1st record ---------------------------------
      READ (1) RUNTIT,RUNTIM,WEIPRI,NCASE
      WRITE(2,100) RUNTIT, RUNTIM, NCASE, WEIPRI
      READ (1) IJUSYL, JTUSYL, PUSRYL, SQSUYL, UUSRYL, VUSRYL, WUSRYL
*----------- loop on bdryx detector data in the present file -----------
      DO 1 IX = 1, MXUSYL
         NY = IX
*        ---------------- read and write 2nd record --------------------
         READ (1, END=1000) MY(NY), TITUYL(NY), ITUSYL(NY), IXUSYL(NY),
     &      IDUSYL(NY), NR1UYL(NY), NR2UYL(NY), USNRYL(NY), SGUSYL(NY),
     &      LLNUYL(NY), EYLLOW(NY), EYLHGH(NY), NEYLBN(NY), DEYLBN(NY),
     &      AYLLOW(NY), AYLHGH(NY)
         WRITE(2,101) MY(NY), TITUYL(NY), IDUSYL(NY), NR1UYL(NY),
     &      NR2UYL(NY), USNRYL(NY), SGUSYL(NY)
         AAUSYL = 0.01D+00 * DBLE (ITUSYL(NY))
         IAUSYL (NY) = NINT (AAUSYL)
         IEUSYL (NY) = ITUSYL (NY) - IAUSYL (NY) * 100
*        ---------------------- low-energy neutrons --------------------
*        ------------ if low-en. neutrons, read group energies ---------
         IF ( LLNUYL (NY) ) THEN
            READ (1) IGMUYL(NY), (ENGMAX(IG), IG = 1, IGMUYL(NY)+1)
            WRITE (2,102) IGMUYL(NY)
```

```
         ELSE
            IGMUYL(NY) = 0
         END IF
*        ----- if second variable is angle, differential is in sr ------
         IF ( IAUSYL (NY) .EQ. 14 .OR. IAUSYL (NY) .EQ. 15 ) THEN
            IF ( AYLHGH (NY) .LT. 0.3D+00 * PIPIPI ) THEN
*              Small angle (exact!):
               TTLOW  = TAN (HLFHLF*AYLLOW(NY))
               TTHGH  = TAN (HLFHLF*AYLHGH(NY))
               DAYLBN (NY) = TWOPIP * TWOTWO * ( TTHGH + TTLOW )
     &                       * ( TTHGH - TTLOW ) / ( ONEONE + TTLOW**2 )
     &                       / ( ONEONE + TTHGH**2 )
            ELSE
*              Large angle:
               DAYLBN (NY) = TWOPIP * ( COS (DBLE(AYLLOW(NY)))
     &                       - COS (DBLE(AYLHGH(NY))) )
            END IF
         ELSE
            DAYLBN (NY) = AYLHGH (NY) - AYLLOW (NY)
         END IF
*        ------------------- linear or log in "energy" -----------------
         IF ( IEUSYL (NY) .GT. 0 ) THEN
*           Linear 1st variable binning
            WRITE (2,103) EYLLOW(NY), EYLHGH(NY), NEYLBN(NY), DEYLBN(NY)
         ELSE
*           Log 1st variable binning
            WRITE (2,104) EYLLOW(NY), EYLHGH(NY), NEYLBN(NY), DEYLBN(NY)
         END IF
*        -------------- there is 1 angle bin only ----------------------
         WRITE (2,105) AYLLOW(NY), AYLHGH(NY), DAYLBN(NY)
*        interv = total number of energy intervals
*        (intervals above the limit for n-groups + intervals below)
         INTERV = NEYLBN(NY) + IGMUYL(NY)
*------------ read the scoring results as a 1-dimensional array --------
         READ (1) (SCORED(J), J = 1, INTERV)
         WRITE (2,'("                              ",
     &     "     Differential         Integral" )')
         WRITE (2,'("                               ",
     &     "        Yield              Yield " )')
         WRITE (2,'("Lower energy    Upper energy",
     &              "    sr**-1 GeV**-1",3X,"number of particles"/)')
         CUMUL = 0.D0
         IF ( LLNUYL (NY) ) THEN
*           low-energy neutron data, if present, are stored backwards
            IG = IGMUYL(NY)
            EN1 = ENGMAX(IG+1)
*           ---------------- loop on low-energy groups -----------------
            DO 2 JG = NEYLBN(NY) + IGMUYL(NY), NEYLBN(NY) + 1,-1
               EN2 = ENGMAX(IG)
               CUMUL = CUMUL + (EN2 - EN1) * SCORED(JG) * DAYLBN(NY)
               WRITE(2,106) EN1, EN2, SCORED(JG), CUMUL
               IG = IG - 1
               EN1 = EN2
 2          CONTINUE            ! end loop on low-energy groups
*           ------------------------------------------------------------
*           adjust lower limit of high energies to make upper limit
*           coincide with the requested one. Count the high energy bins
            NHIGH(NY) = 0
            EYLLOW(NY) = EN1
            EN1 = EYLHGH(NY)
            DO 3 IE = 1, NEYLBN(NY)
               IF(IEUSYL(NY) .GT. 0) THEN
```

```
                  EN2 = EN1 - DEYLBN(NY)
               ELSE
                  EN2 = EN1 / DEYLBN(NY)
               END IF
               EN1 = EN2
               NHIGH(NY) = NHIGH(NY) + 1
               IF(EN1 .LE. EYLLOW(NY)) GO TO 4
 3          CONTINUE
 4          CONTINUE
         ELSE
            EN1 = EYLLOW(NY)
            NHIGH(NY) = NEYLBN(NY)
         END IF
*        ---------------------------------------------------------------
*        first bin above the n-group limit (may have different width)
         IF(IEUSYL(NY) .GT. 0) THEN
            EN2 = EN1 + DEYLBN(NY)
         ELSE
            EN2 = EN1 * DEYLBN(NY)
         END IF
*        CUMUL = CUMUL + (EN2 - ETCLOW(NX)) *
         CUMUL = CUMUL +
     &      (EN2 - EN1) * SCORED(NEYLBN(NY) - NHIGH(NY) +1) * DAYLBN(NY)
         WRITE(2,106) EYLLOW(NY), EN2,
     &               SCORED(NEYLBN(NY) - NHIGH(NY) + 1), CUMUL
         EN1 = EN2
*        ---------- loop on energies above the n-group limit ----------
         DO 5 IE = 2, NHIGH(NY)
            IF(IEUSYL(NY) .GT. 0) THEN
               EN2 = EN1 + DEYLBN(NY)
            ELSE
               EN2 = EN1 * DEYLBN(NY)
            END IF
            CUMUL = CUMUL +
     &          (EN2 - EN1)*SCORED(NEYLBN(NY)-NHIGH(NY)+IE) * DAYLBN(NY)
            WRITE(2,106) EN1, EN2,SCORED(NEYLBN(NY)-NHIGH(NY)+IE), CUMUL
            EN1 = EN2
 5       CONTINUE            ! end loop on energies above limit
*        ---------------------------------------------------------------
 1    CONTINUE               ! end loop on detectors
*     ---------------------------------------------------------------
 1000 CONTINUE

 100  FORMAT(/,1X,'*****',2X,A80,2X,'*****',/,/,10X,A32,/,/,
     &        10X,'Total number of particles followed ',I9,', for a ',
     &        'total weight of ',1P,E15.8,/)
 101  FORMAT(/,3X,'Yield n. ',I3,'  "',A10,
     &        '" , generalised particle n. ',I4,', from region n. ',I6,
     &        ' to region n. ',I6,
     &        /,6X,'user normalisation: ',1P,E15.8,', adopted cross ',
     &        'section (if any): ',1P,E15.8,' mb')
 102  FORMAT(6X,'low-energy neutrons scored from group 1 to group ', I5)
 103  FORMAT(6X,'linear energy binning from  ',1P,E11.4,' to ',
     &        E11.4,' GeV, ',0P,I5,' bins (',1P,E11.4,' GeV wide)')
 104  FORMAT(6X,'logar. energy binning from ',1P,E11.4,' to ',
     &        E11.4,' GeV, ',0P,I5,' bins (ratio :',1P,E11.4,')')
 105  FORMAT(6X,'One angular bin from ',1P,E11.4,' to ',
     &        E11.4,' rad , (',1P,E11.4,' sr wide )'/)
 106  FORMAT(1P,2(E11.4,5X),2(E14.7,5X))
      END
```

Example:

```
*...+....1....+....2....+....3....+....4....+....5....+....6....+....7...+...8
USRYIELD      1399.0       13.      21.0       3.0       2.0       1.0TotPi+(E)
USRYIELD        50.0     0.001     100.03.14159265       0.0       3.0 &
* Score double differential yield of positive pions going from region 3 to
* region 2 with a polar angle between 0 and pi with respect to the beam
* direction. Energy distribution is in 100 logarithmic intervals between 1 MeV
* and 50 GeV. Normalisation factor = 1. Results are written formatted on
* unit 21.
```

7.83 WW–FACTOr

Defines Weight Windows in selected regions

See also BIASING, WW–THRESh, WW–PROFIle
Attention: Option WW–FACTOr alone is not sufficient to define a weight window. One or more WW–THRESh cards (p. 249) are also necessary in order to activate the window.

WHAT(1) $\geq$ `0.0`: Russian Roulette (RR) parameter (Window "bottom" weight at the lower energy threshold set by WW–THRESh).
$<$ `0.0`: resets to `-1.0` (no RR) a possible positive value set in a previous WW–FACTOr card
This value can be modified by WHAT(4) in option WW–THRESh or by WHAT(2) in WW–PROFIle, and can be overridden in the user routine `UBSSET` (p. 320) by assigning a value to variable `WWLOW`.
Default = `-1.0` (no RR)

WHAT(2) $>$ `1.7*WHAT(1)`: Splitting parameter (Window "top" weight at the lower energy threshold set by WW–THRESh)
= `0.0`: ignored
$\leq$ `1.7*WHAT(1)`: resets to ∞ (no splitting) a possible value set in a previous WW-FACTOr card
This value can be modified by WHAT(4) in option WW–THRESh or by WHAT(2) in WW–PROFIle (p. 247), and can be overridden in the user routine `UBSSET` (p. 320) by assigning a value to variable `WWHIG`.
Default = ∞ (no splitting)

WHAT(3) $>$ `0.0`: Multiplicative factor to be applied to the two energy thresholds for RR/splitting (defined by option WW-THRESh) in the region of interest
= `0.0`: ignored
$<$ `0.0`: resets to `1.0` (thresholds not modified) a possible value set in a previous WW–FACTOr card
This value can be overridden in the user routine `UBSSET` (p. 320) by assigning a value to variable `WWMUL`.
Default = `1.0` (RR/splitting thresholds are not modified)

WHAT(4) = lower bound of the region indices in which the indicated RR and/or splitting parameters apply
("*From region WHAT(4)...*")
Default = `2.0`

WHAT(5) = upper bound of the region indices in which the indicated RR and/or splitting parameters apply
("*... to region WHAT(5)...*")
Default = WHAT(4)

WHAT(6) = step length in assigning indices
("*... in steps of WHAT(6)*")
Default = `1.0`

SDUM = a number from `1.0` to `5.0` in any position, indicating the low-energy neutron weight window profile to be applied in the regions selected (see WW–PROFIle). Exceptionally, here SDUM must be a number, in free format, rather than a character string.
= blank, zero or non numerical: ignored
$<$ `0.0`: resets to `1.0` a possible value previously given.
This value can be overridden in the user routine `UBSSET` (p. 320) by assigning a value to variable `JWSHPP`.

Default (if no WW–PROFIle card is present): profile number 1

Default : (option WW–FACTOr or WW–THRESh not given): no weight window is defined

Notes

1. Option WW–FACTOr, which must be used together with WW–THRESh (p. 249), allows the user to define a very detailed weight window for Russian Roulette and splitting: energy-dependent, per region and per particle. WW–THRESh is used to set two basic energy values for each particle (including electrons and photons but not low-energy neutrons). From each basic couple of energies, a different couple of thresholds is generated for each region by multiplication with the factor provided in WHAT(3). A weight window of minimum width is defined at the lower threshold by its bottom and top edges (WHAT(1) and WHAT(2)); a second wider window is obtained from it at the higher threshold by increasing the "top edge" (splitting level) and decreasing the "bottom edge" (RR level) by the amplification factor given with WW–THRESh. The whole energy range is thus divided in three parts. In the high-energy part (above the higher threshold) the window is of infinite width, i.e., no splitting/RR takes place. In the medium-energy range the window narrows down continuously with decreasing energy, its top and bottom edges varying linearly with energy between the two thresholds. In the low-energy range the window width remains constant and equal to the minimum value it has at the lower threshold.
2. Russian Roulette is played in a given region if the particle weight is lower than the bottom window edge for that energy, particle and region. The particle survives with a probability equal to the ratio between its weight and the RR edge, and is given a new weight equal to the RR edge itself.
   Splitting is performed if the particle weight is higher than the top window edge for that energy, particle and region. The particle is replaced by two identical ones with half its weight. Note that the top edge must always be at least a factor two higher than the bottom one, in order to avoid repeated and useless changes of weight. Actually, it is suggested to never make this factor less than 3 or 4.
3. For low-energy neutrons, a different scheme applies. Instead of dividing the energy range into three parts (constant window, continuously varying window, infinite window), the window is assigned group by group by means of option WW–PROFIle (7.84), creating a so-called "weight-window profile". On the other hand, it is not possible to assign a different profile to each region, but only a maximum of 5 different profiles are allowed.
4. A form of splitting and Russian Roulette is also provided by option BIASING (p. 71). The two options, however, are different in many respects:
   - with WW–FACTOr, splitting and RR are played at the moment a particle is taken from the stack and starts to be transported. With BIASING, splitting/RR happens when a particle crosses a boundary (in the case of hadrons also — on request — before loading in stack the secondaries from an inelastic hadron collision)
   - while the criterion used by BIASING to trigger splitting/RR depends only on the *relative importance* of various regions of phase space, the weight window is based on *absolute weight standards* pre-assigned to different phase space regions
   - BIASING can have two purposes: when used at collisions with RR only, i.e., reducing factor < 1, it aims at increasing the total number of histories simulated in a given time, namely to sample over a more extended part of phase space (e.g., more primary interactions) without leaving any important part not sufficiently represented. (This is also true of leading particle biasing for electrons and photons via option EMF–BIAS, p. 98). At the same time (and this holds also for splitting, especially when the option is used at boundary crossing) it can be applied to sample preferentially from those regions of phase space which contribute more to the result.
     This second purpose is also that of the WW–FACTOr weight window, but in addition this option has the advantage to avoid excessive weight fluctuations. These can be dangerous in two ways. In general, if transport is biased and no control is kept on particle weight, it can happen that too much time is wasted by tracking particles of very low weight which can only contribute little to the score. On the other hand, too large weights can also be a problem. If the part of phase space used for scoring (the "detector") is very small (typically an element of a "binning" mesh), so that only a limited number of particles have a chance to enter it, it is statistically important that they all make contributions of the same order. A rare particle of large weight crossing the detector would give rise to an anomalous score not compensated by opposite fluctuations.

   Why should one then use BIASING and not the weight window? The answer is that "tuning" an absolute weight by region and energy is more powerful but also much more difficult and time-consuming than just quantifying relative spatial importances. In general, it requires a lot of experience which can often be obtained only by performing repeated runs of the same case and by making a careful statistical analysis of history distributions in phase space. Not all problems are worth of it and not all users are able to do it.

5. It can also be said that WW–FACTOr and BIASING (and other non-analogue transport options) are not necessarily mutually exclusive; on the contrary the weight window can be successfully used to damp excessive weight fluctuations originated by other techniques. However, it is the user's responsibility to ensure that the average absolute weights produced independently by the different options be of the same order of magnitude. Otherwise, possible conflicts could give rise to a waste of time due to an excessive rate of weight adjustments, and even to incorrect results.
6. The weight limits defined by WW–FACTOr apply to all particles: however, it is possible to set different values for specific particles (see WHAT(3) of option WW–THRESh). This is especially necessary when secondary particles are generated with a weight much smaller than the parent particles of a different kind (for instance, as the result of the LAM–BIAS option).
7. WW-FACTOr is one of the two FLUKA options where SDUM is used to input numerical data. (Actually, the material number is first read as a string and then an internal reading is performed on the string to get the number).

Example 1:

```
*...+....1....+....2....+....3....+....4....+....5....+....6....+....7...+...8
WW-FACTOR       76.0    1200.0       1.0       5.0       6.0       0.0
*  In regions 5 and 6, set the lower weight limit = 76.0 and set the upper
*  limit = 1200. No modification of the two energy thresholds.
```

Example 2:

```
*...+....1....+....2....+....3....+....4....+....5....+....6....+....7...+...8
WW-FACTOR       13.0     120.0       1.5      27.0      31.0       2.0   3.
*  In regions 27, 29 and 31, set the lower weight limit = 13. and set the
*  upper limit = 120. The two energy thresholds set by WW-THRES are multiplied
*  by a factor 1.5. Apply low-energy neutron profile number 3.
```

7.84 WW–PROFIle

Defines extra factors dependent on energy group ("profiles"), to modify the basic setting of the low-energy neutron weight windows in selected sets of regions, or the low-energy neutron importances in each region

See also BIASING, WW–FACTOr, WW–THRESh

WHAT(1) = weight-window extra factor for the profile defined by WHAT(6), concerning the energy groups defined by WHAT(3), WHAT(4) and WHAT(5) (both the top and bottom window levels will be multiplied by WHAT(1)). See Note 2.
= `0.0`: ignored
< `0.0`: resets to default (extra factor = `1.0`)
Default = `1.0` (windows are not modified)

WHAT(2) = importance extra factor. See Note 3.
= `0.0`: ignored
< `0.0`: resets to default (extra factor = `1.0`)
Default = `1.0` (importances are not modified)

WHAT(3) = lower bound of the group numbers for which the extra factor WHAT(1) or WHAT(2) is requested
("*From group WHAT(3)...*")
Default = `1.0` (the group of highest energy)

WHAT(4) = upper bound of the group numbers for which the extra factor WHAT(1) or WHAT(2) is requested
("*...to group WHAT(4)...*")
Default = WHAT(3)

WHAT(5) = step length in assigning group numbers
("*...in steps of WHAT(5)*")
Default = `1.0`

WHAT(6) = profile number defined by WHAT(1), WHAT(3–5) (up to 5 different profiles are allowed).
Default : profile number 1

SDUM : not used

Default : (option WW–PROFIle not given): no extra factor is applied to low-energy neutron windows and importances

Notes

1. Option WW–PROFIle applies only to low-energy neutrons. It is used to refine the basic bias setting defined by two other options: WW–FACTOr (p. 244) and BIASING (p. 71).
2. WHAT(1) refers to WW–FACTOr: it allows the user to tune the weight window by energy group (WW–FACTOr does the same by region). The profile defined will be applied to raise or lower the weight-window levels (for low-energy neutrons only) in a group of regions selected by means of WHAT(4–6) and SDUM in option WW–FACTOr.
3. WHAT(2) refers to BIASING: its aim is to define a reference weight level in each region, which is used by the program to avoid excessive biasing in some critical cases. If the user has defined a weight-window (options WW–FACTOr and WW–THRESh), the reference weight level is not needed because it is derived directly from the window parameters. If the user has not defined a weight-window but has defined region importances

(option BIASING), the reference weight level for a region is assumed in most cases to be the inverse of the corresponding importance. However, since importance biasing is not based on absolute values of importance but on importance ratios, in some rare cases the user may give importances which are not equal to the inverse of the average particle weight, but only proportional to it. (This is in order to better exploit the full importance range, since for technical reasons in FLUKA allowed importance values range only from `0.0001` to `10000.`). In such cases it is possible to multiply all the importances by a factor WHAT(2) *only for the purpose of calculating the reference weight level*.

Modification of importances by a factor WHAT(2) apply to *all* regions (but only for low-energy neutrons). If neither weight-window nor importances have been given, FLUKA still calculates a weight reference level from the ratio of physical to biased non-absorption probability. If a particle's weight exceeds the reference level in a given region by more than a factor calculated at run time, non-absorption probability biasing is switched off and transport continues according to the physical absorption probabilities (analogue transport).

Example 1:

```
*...+....1....+....2....+....3....+....4....+....5....+....6....+....7...+...8
WW-PROFIle       0.9       0.0        1.       11.       0.0       4.0
WW-PROFIle       0.7       0.0       12.       70.       0.0       4.0
WW-PROFIle       0.5       0.0       71.       72.       0.0       4.0
*  Profile n. 4 is defined a multiplication factor for weight windows, where
*  the upper and the lower weight limits (as defined by WW-FACTOr and
*  WW-THRESh) are multiplied by 0.9 for the first 11 neutron groups, by 0.7
*  for groups 12 to 70, and by 0.5 for groups 71 and 72.
```

Example 2:

```
*...+....1....+....2....+....3....+....4....+....5....+....6....+....7...+...8
WW-PROFIle       0.0       1.8        1.       65.       0.0       2.0
WW-PROFIle       0.0       2.3       66.       72.       0.0       2.0
*  Profile n. 2 is defined a multiplication factor for importances, where
*  the importances (as defined by BIASING) are multiplied by 1.8 for the first
*  65 neutron groups, and by 2.3 for groups 66 to 72.
```

7.85 WW–THRESh

Defines the energy limits for a Russian Roulette/splitting weight window and applies particle-dependent modification factors to the windows defined by WW–FACTOr

See also BIASING, WW–FACTOr, WW–PROFIle

WHAT(1) $>$ 0.0: upper kinetic energy threshold in GeV for Russian Roulette (RR)/Splitting with a weight window. For low-energy neutrons, corresponding (smallest) group number (included)
$=$ 0.0: ignored
$<$ 0.0: any previously selected threshold is cancelled

WHAT(2) $\geq$ 0.0 and $<$ WHAT(1): lower kinetic energy threshold in GeV for RR/Splitting with a weight window. For low-energy neutrons, corresponding (largest) group number (included)
$<$ 0.0 or $>$ WHAT(1): WHAT(2) is set $=$ WHAT(1)

WHAT(3) $>$ 0.0: amplification factor used to define the weight window width at the higher energy threshold represented by WHAT(1). The weight window at the higher energy threshold is obtained by multiplying by WHAT(3) the top edge of the window (splitting level) at the lower threshold, and dividing by the same factor the bottom edge (RR-level)
$<$ 0.0: |WHAT(3)| is used as a multiplication factor for the bottom and top levels of every region for the particles selected by WHAT(4–6). That is, for such particles both bottom and top are multiplied by |WHAT(3)|
Default $=$ 10.0 (amplification factor for the splitting and RR-level at the higher threshold). The particle dependent multiplication factor by default is set $=$ 1.0

WHAT(4) $=$ lower bound of the particle numbers to which the indicated weight window energy limits apply. Note that particle number 40.0 indicates low-energy neutrons (for this purpose only!). Particle number 8.0 indicates neutrons with energy $>$ 19.6 MeV
("*From particle* WHAT(4)*...*")
Default $=$ 1.0

WHAT(5) $=$ upper bound of the particle numbers to which the indicated weight window energy limits apply
("*...to particle WHAT(5)...*")
Default $=$ WHAT(4) if WHAT(4) $>$ 0.0, all particles otherwise

WHAT(6) $=$ step length in assigning numbers
("*...in steps of WHAT(6)*")
Default $=$ 1.0

SDUM $=$ PRIMARY: the weight window applies also to primary particles
$=$ NOPRIMARy: the weight window doesn't apply to primaries
Default $=$ PRIMARY

Default : (option WW–THRESh or WW–FACTOr not given): no weight window is defined

Notes

1. Option WW–THRESh is only meaningful when WW-FACTOr (7.83) is also requested. See Note 1 to that option for more information.
2. For low-energy neutrons, the two energy thresholds are expressed as group numbers, while for all other particles (including high-energy neutrons) they are expressed in GeV. Therefore, thresholds for low-energy neutrons must be assigned in a separate WW–THRESh card from other particles.

Example:

```
*...+....1....+....2....+....3....+....4....+....5....+....6....+....7...+...8
WW-THRESh        2.0      0.05       2.4       3.0       7.0       0.0
*  The weight window weight limits for particles 3 to 7 (practically electrons,
*  positrons and photons, since neutrinos are discarded), which have been set
*  by WW-FACTOr are applied as such below 50 MeV. Above that energy, the width
*  of the window is progressively increased up to 2 GeV: at 2 GeV, the upper
*  weight limit is a factor 2.4 larger and the lower limit a factor 2.4
*  smaller.
```

Chapter 8

Combinatorial Geometry

8.1 Introduction

The Combinatorial Geometry (CG) used by FLUKA is a modification of the package developed at ORNL for the neutron and gamma-ray transport program MORSE [44] which was based on the original combinatorial geometry by MAGI (Mathematical Applications Group, Inc.) [78, 98].

The default input format is fixed, and different from that adopted elsewhere in the FLUKA code. The input sequence must be completely contained between a GEOBEGIN and a GEOEND card (see the corresponding description on p. 121, 123).

Two concepts are fundamental in CG: bodies and regions. Originally, MORSE bodies were defined as convex solid bodies (finite portions of space completely delimited by surfaces of first or second degree, i.e., planes or quadrics). In FLUKA, the definition has been extended to include infinite cylinders (circular and elliptical) and planes (half-spaces). Use of such "infinite bodies" is encouraged since it makes input preparation and modification much easier and less error-prone. They also provide a more accurate and faster tracking.

Regions are defined as combinations of bodies obtained by boolean operations: Union, Subtraction and Intersection. Each region is not necessarily simply connected (it can be made of two or more non contiguous parts), but must be of homogeneous material composition. Because the ray tracing routines cannot track across the outermost boundary, all the regions must be contained within a surrounding "blackhole" (an infinitely absorbing material, in MORSE jargon "external void", designated by the FLUKA material number 1), so that all escaping particles are absorbed. It is suggested to make the external blackhole region rather big, so as not to interfere with possible future modifications of the problem layout. The external blackhole must be completely surrounded by the boundary of a closed body, and therefore cannot be defined by means of half-spaces or infinite cylinders only. Inside such outermost boundary, *each point of space must belong to one and only one region.*

Note that in MORSE the concept of "region" refers to a portion of space of homogeneous statistical importance or weight setting, which may extend over one or several "zones" (homogeneous in material composition). Since the two MORSE concepts of region and zone coincide in FLUKA (there is a one-to-one correspondence), the term "region" will be used here to define "a portion of space of uniform material composition, obtained by boolean operations on one or more subregions", while "zone" will indicate one of such subregions, obtained by boolean operations on one or more geometrical bodies.

Repetition of sets of regions according to symmetry transformations is possible in FLUKA through the card LATTICE (p. 268) and through a user-written routine. This allows, for instance, to model in detail only a single cell of a calorimeter and to replicate it in the entire volume.

8.2 Combinatorial Geometry input

CG input must respect the following sequential order:

GEOBEGIN card	(in FLUKA standard format, or free format if requested by a FREE or GLOBAL command)
Geometry title	(in special format, or in free geometry format if requested by a GLOBAL command)
Body data	(in special or free geometry format)
END card	(in special or free geometry format)
Region data	(in special or free geometry format)
END card	(in special or free geometry format)
LATTICE cards	(optional, in FLUKA standard format, or in free format if requested by a FREE or GLOBAL command)
Region volumes	(optional, see Geometry title card)

GEOEND card (in FLUKA standard format, or free format if requested by a FREE or GLOBAL command)

8.2.1 GEOBEGIN card

This card follows the general FLUKA format (see description on p. 45). It can be in free format if the latter has been requested with option FREE (see p. 120) or GLOBAL (p. 125). The rest of the geometry must be in a special fixed format described below, unless free geometry format has been requested by GLOBAL.

The meanings of the WHAT and SDUM parameters are:

WHAT(1) : flag for printing geometry error messages:
≠ 0.0: only a global summary is given

Default = 0.0 (all geometry error messages are printed)

WHAT(2) : used to set the accuracy parameter — reserved for program development

WHAT(3) > 0.0: logical unit for geometry input. The name of the corresponding file must be input on the next card if WHAT(3) is different from 5.0. Note that values of WHAT(3) ≠ 5.0 and < 21.0 must be avoided because of possible conflicts with FLUKA pre-defined units.

Default = 5.0 (i.e., geometry input follows)

WHAT(4) > 0.0: logical unit for geometry output. If different from 11, the name of the corresponding file must be input on the next card if WHAT(3) = 0.0 or 5.0, otherwise on the card following the next one. Values of WHAT(3) ≠ 11.0 and < 21.0 must be avoided because of possible conflicts with FLUKA pre-defined units.

Default = 11.0 (i.e., geometry output is printed on the standard output)

WHAT(5), WHAT(6): not used

SDUM : must be blank or = COMBINAT

For further information on the use of the GEOBEGIN card, the reader is referred to option GEOBEGIN on page 121.

8.2.2 Geometry Title card

Three variables are input in the CG Title card: IVOPT, IDBG, TITLE. The format is (2I5,10X,A60).
The first integer value (IVOPT = Input Volume OPTion) is a flag to indicate how to normalise the quantities scored in regions by the FLUKA option SCORE (p. 195).

IVOPT = 0 means that no normalisation must be performed (output values are total stars or total energy deposited in each region).

IVOPT = 1 and IVOPT = 2 have no meaning in FLUKA and are not allowed.

IVOPT = 3 means that the scores must be normalised dividing by region volumes input by the user just before the GEOEND card (see 8.2.8)

The second integer value can be used to modify the format with which body and region data are read:

IDBG = 0 or 10 : default fixed format for both bodies and regions
= -10 : high-accuracy fixed format for bodies; default fixed region
= -100 : high-accuracy fixed format for bodies; region fixed format allowing more than 10000 regions
= 100 : default fixed format for bodies; region fixed format allowing more than 10000 regions

Any other value should be avoided. The value of IDBG is irrelevant if free format has been requested (see

the GLOBAL command, p. 125). The remaining 60 characters can be used for any alphanumeric string at the user's choice.

8.2.3 Body data

The geometry must be specified by establishing two tables. The first table describes the type, size and location of the bodies used in the geometry description. The second table defines the physical regions in terms of these bodies.

Each body *type* is referred to by a three-letter code.

There are three kinds of possible input formats, two fixed and one free. Free format, if used, implies necessarily also the use of free format in region input (see 8.2.6).

8.2.3.1 Fixed format body input

Fixed format for both body and region input is the default, unless requested differently by a GLOBAL command at the beginning of the input file. In fixed format, each *input* body is defined by: its code, a sequential number, and a set of floating point numerical parameters defining its size, position and orientation in space (all in cm).

Default fixed format is the original CG one as used in MORSE and in other Monte Carlo programs. It expects up to 6 floating point values per line.
High-accuracy fixed format allows to enter numerical data with full precision (16 significant digits) and accommodates only a maximum of 3 floating point values per line.

The fixed input format for each body depends on the value of the IDBG variable given in the Geometry Title card (see 8.2.2 above).
If `IDBG = 0, 10` or `100`, the body input format is `(2X, A3, I5, 6D10.3)`;
if `IDBG = -10` or `-100`, the format is `(2X, A3, I5, 3D22.15)`;
where the 3-letter code in columns 3-5 is one of the following:

```
ARB BOX ELL PLA RAW RCC REC RPP SPH TRC
WED XCC XEC XYP XZP YCC YEC YZP ZCC ZEC
```

(columns 3–5 must be left blank in continuation cards). The integer in columns 6–10 is the body sequential number (if left blank numbers are assigned automatically, but this is not recommended; it must be left blank in continuation cards).
The floating-point numbers in columns 11–76 are geometrical quantities defining the body (their number depends on the body type as explained below, and can extend over several continuation lines). The presence of the decimal point in the numerical data is compulsory.

8.2.3.2 Free format body input

Free format is used for both body and region input only if requested by a GLOBAL command (see p. 125) at the beginning of the input file.

In free format, each body is defined by: its code, its identifier (an alphanumeric string of up to 8 characters, with the first character alphabetical) and a set of numerical parameters defining its size, position and orientation in space (all in cm).

Free format has been introduced only recently and is expected to supersede soon the other formats, which will be kept however for reasons of back compatibility. Its main advantages, in addition to the freedom from strict alignment rules, are the possibility to modify the input sequence without affecting the region description (for instance, by inserting a new body) and the availability of parentheses to perform complex boolean operations in the description of regions.

The input for each body consists of a 3-letter code indicating the body type

```
ARB BOX ELL PLA RAW RCC REC RPP SPH TRC
WED XCC XEC XYP XZP YCC YEC YZP ZCC ZEC
```

followed by a unique "body name" (alphanumeric identifier) and a set of geometrical quantities defining the body (their number depends on the body type as explained below). The different items, separated by one or more blanks, or by one of the separators `, / ; :` can extend over as many lines as needed. See option FREE (p. 120) for more detailed instructions on the use of separators.

After the last body description, end with a line having the code `END`.

With all input formats, a card having an asterisk (*) in column 1 is treated as a comment card. Such comment cards can be inserted freely at any point of Body and Region input, allowing easier identification.

8.2.4 Body types

8.2.4.1 Rectangular Parallelepiped. Code: RPP

An RPP (Fig. 8.1) has its edges parallel to the coordinate axes.
It is defined by 6 numbers in the following order: X_{min}, X_{max}, Y_{min}, Y_{max}, Z_{min}, Z_{max} (minimum and maximum coordinates which bound the parallelepiped).
An RPP definition extends over one single card in default fixed format, or over two cards in high-accuracy body fixed format (`IDBG = -10` or `-100` in the CG Title card, see 8.2.2).
Example (the comment lines shown are allowed input lines):

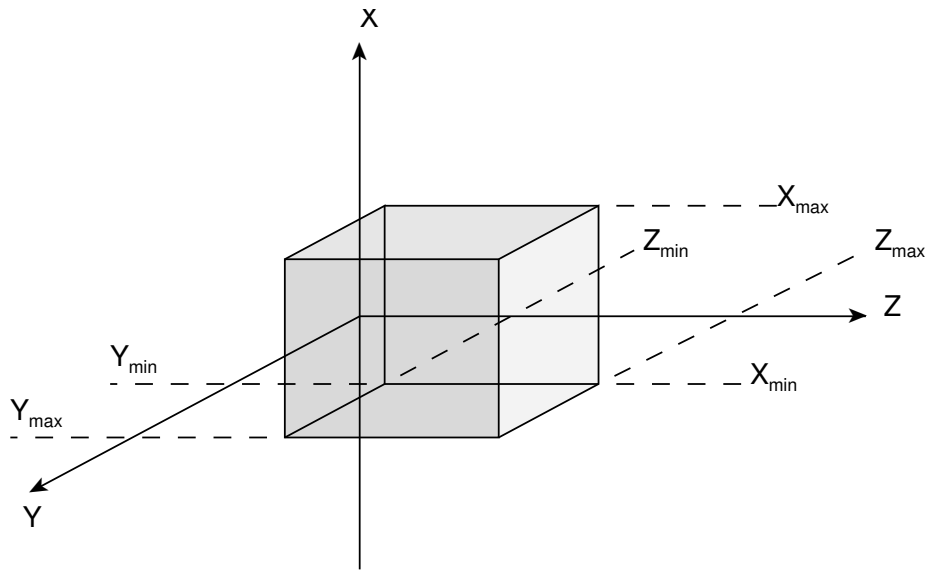

Fig. 8.1: Rectangular Parallelepiped (RPP)

```
*...+....1....+....2....+....3....+....4....+....5....+....6....+....7..
  RPP    4     -20.0     +20.0     -50.0     +50.0     -38.5     +38.5
* (a parallelepiped centred on the origin)
```

The same input, in high-accuracy fixed format, would be:

```
*...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+.
  RPP    4                -20.0               +20.0                -50.0
                          +50.0               -38.5                +38.5
```

The same input could be, in free format:

```
RPP SmlBrick  -20.0  +20.0  -50.0  +50.0  -38.5  +38.5
```

8.2.4.2 General Rectangular Parallelepiped. Code: BOX

A BOX (Fig. 8.2) is also a Rectangular Parallelepiped, but with arbitrary orientation in space. Its use is generally not recommended, since it can be replaced by a suitable combination of infinite planes (PLA, XYP, XZP, YZP). Planes are easier to define, make tracking more accurate and often only a few are needed in a region description.

A BOX is defined by 12 numbers in the following order: V_x, V_y, V_z (coordinates of a vertex), $H_x^{(1)}$, $H_y^{(1)}$, $H_z^{(1)}$, $H_x^{(2)}$, $H_y^{(2)}$, $H_z^{(2)}$, $H_x^{(3)}$, $H_y^{(3)}$, $H_z^{(3)}$ (x-, y- and z- components of three mutually *perpendicular* vectors representing the height, width and length of the box). Note that it is the user's responsibility to ensure perpendicularity. This is best attained if the user has chosen high-accuracy input fixed format (IDBG = -10 or -100 in the CG Title card, see 8.2.2), or free format, and the value of each vector component is expressed with the largest available number of significant digits.

A BOX definition extends over 2 cards in default fixed format, or over 4 cards in high-accuracy body fixed format.

Example in default fixed format:

```
*...+....1....+....2....+....3....+....4....+....5....+....6....+....7..
  BOX   18       0.0       0.0       0.0 7.0710678 7.0710678       0.0
         -14.142136 14.142136       0.0       0.0       0.0      30.0
* (a parallelepiped with a corner on the origin, with edges 10, 20 and
*  30 cm long, rotated counterclockwise by 45 degrees in the x-y plane)
```

The same example in high-accuracy body fixed format:

```
*...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+.
  BOX   18                   0.0                   0.0                  0.0
              7.071067811865475     7.071067811865475                  0.0
             -14.14213562373095     14.14213562373095                  0.0
                            0.0                   0.0                 30.0
```

The same, in free format:

```
*...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+.
BOX tiltslab  0.0  0.0  0.0  7.071067811865475  7.071067811865475  0.0
           -14.14213562373095  14.14213562373095  0.0  0.0  0.0  30.0
```

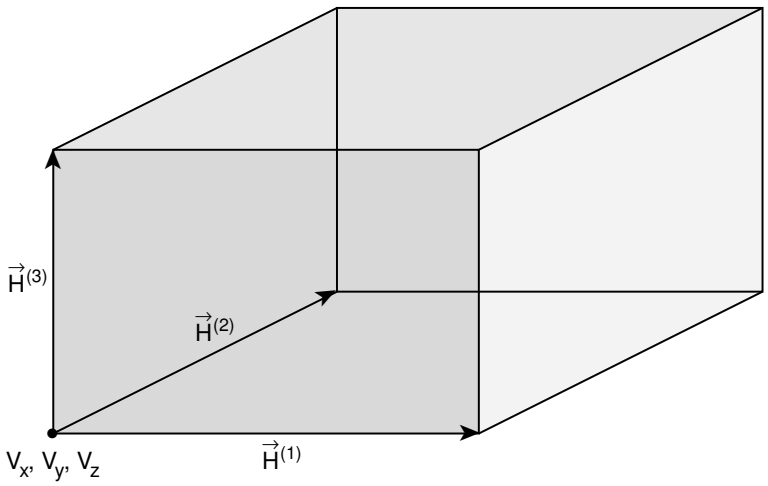

Fig. 8.2: General Rectangular Parallelepiped (BOX)

8.2.4.3 Sphere. Code: SPH

A SPH (Fig. 8.3) is defined by 4 numbers: V_x, V_y, V_z (coordinates of the centre), R (radius).
A SPH definition extends over one single card in default fixed format, or over two cards in high-precision body fixed format.
Example in default fixed format:

```
*...+....1....+....2....+....3....+....4....+....5....+....6....+....7..
  SPH  002       5.0     -30.0      27.0     528.2
* (a sphere centred at point x=5, y=30, z=27, with a radius of 528.2 cm)
```

The same example in high-accuracy body fixed format:

```
*...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+.
  SPH  002                   5.0                 -30.0                  27.0
                           528.2
```

The same, in free format:

```
SPH  TheBall  5.0  -30.0  27.0  528.2
```

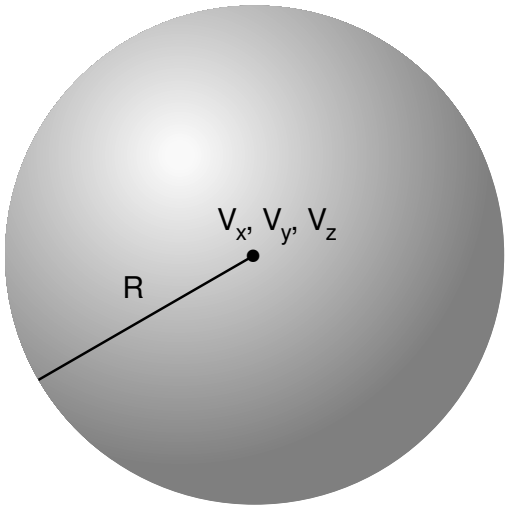

Fig. 8.3: Sphere (SPH)

8.2.4.4 Right Circular Cylinder. Code: RCC

An RCC (Fig. 8.4) can have any orientation in space. It is limited by a cylindrical surface and by two plane faces perpendicular to its axis. (If the cylinder axis is parallel to one of the coordinate axes, it is worth considering instead the use of an infinite cylinder XCC, YCC or ZCC (see below), leading to increased tracking speed).
Each RCC is defined by 7 numbers: V_x, V_y, V_z (coordinates of the centre of one of the circular plane faces), H_x, H_y, H_z (x-, y- and z- components of a vector corresponding to the cylinder height, pointing to the other plane face), R (cylinder radius).
An RCC definition extends over 2 cards in default fixed format, or over 3 cards in high-accuracy body fixed format.
Example in default fixed format:

```
*...+....1....+....2....+....3....+....4....+....5....+....6....+....7..
  RCC   07       5.0       5.0       5.0 57.735027 57.735027 57.735027
                 37.
* (a circular cylinder 100 cm long and of 37 cm radius, with base
```

```
*  centred at point x=5, y=5, z=5, its axis making equal angles to
*  the three coordinate axes).
```

The same example in high-accuracy body fixed format:

```
*...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+.
  RCC   07                   5.0                   5.0                   5.0
               57.73502691896258     57.73502691896258     57.73502691896258
                           37.
```

The same cylinder, in free format:

```
RCC  BYGCYL  5.0  5.0  5.0  57.73502691896258  57.73502691896258
       57.73502691896258  37.
```

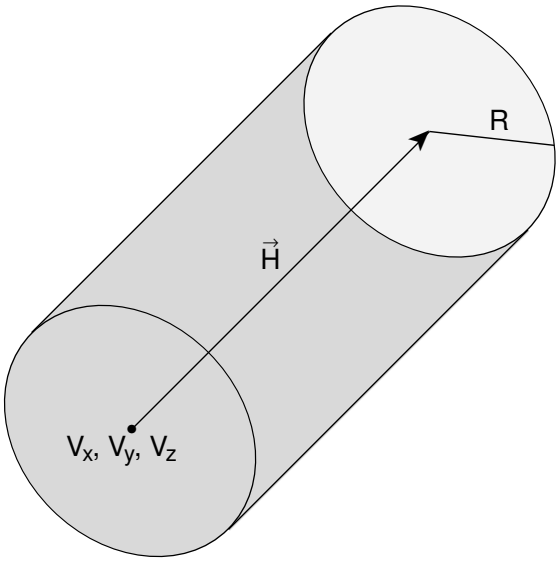

Fig. 8.4: Right Circular Cylinder (RCC)

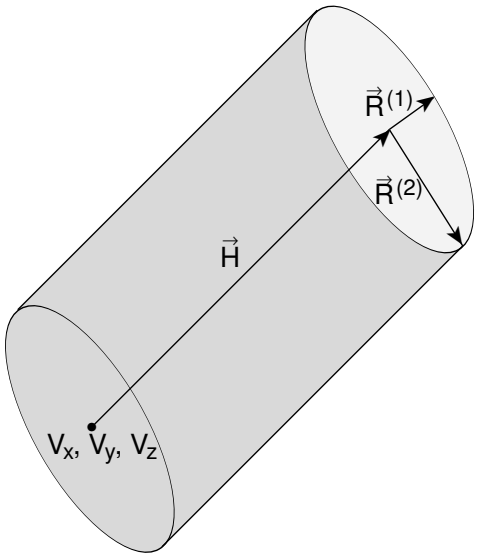

Fig. 8.5: Right Circular Cylinder (REC)

8.2.4.5 Right Elliptical Cylinder. Code: REC

A REC (Fig. 8.5) can have any orientation in space. It is limited by a cylindrical elliptical surface and by two plane faces perpendicular to its axis. (If the cylinder axis is parallel to one of the coordinate axes, it is worth considering instead the use of an infinite cylinder XEC, YEC or ZEC (see below), leading to increased tracking speed).
Each REC is defined by 12 numbers: V_x, V_y, V_z (coordinates of the centre of one of the elliptical plane faces), H_x, H_y, H_z (x-, y- and z- components of a vector corresponding to cylinder height, pointing to other plane face), $R_x^{(1)}$, $R_y^{(1)}$, $R_z^{(1)}$ (components of a vector corresponding to the minor half-axis of the cylinder elliptical base), $R_x^{(2)}$, $R_y^{(2)}$, $R_z^{(2)}$ (ditto for the major half-axis).
A REC definition extends over 2 cards in default fixed format, or over 4 cards in high-accuracy body fixed format.
Example in default fixed format:

```
*...+....1....+....2....+....3....+....4....+....5....+....6....+....7..
  REC    1     -10.0      12.0       7.0       0.0      58.0       0.0
                 9.       0.0       0.0       0.0       0.0      17.0
* (an elliptical cylinder 58 cm long parallel to the y axis, with minor
*  half-axis 9 cm long parallel to the x coordinate axis, major
*  half-axis 17 cm long parallel to the z axis, base centred at point
*  x=-10, y=12, z=7)
```

The same example in high-accuracy body fixed format:

```
*...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+.
  REC    1                -10.0                 12.0                  7.0
                            0.0                 58.0                  0.0
                             9.                  0.0                  0.0
                            0.0                  0.0                 17.0
```

The same, in free format:

```
REC pipesurf -10.0 12.0  7.0  0.0  58.0  0.0  9.0  0.0  0.0  0.0  0.0  17.0
```

8.2.4.6 Truncated Right Angle Cone. Code: TRC

A TRC (Fig. 8.6) can have any orientation in space. It is bounded by a conical surface and by two circular plane faces perpendicular to the axis of the cone.
Each TRC is defined by 8 numbers: V_x, V_y, V_z, (coordinates of the centre of the major circular base), H_x, H_y, H_z (components of a vector corresponding to the TRC height, directed from the major to the minor base), $R^{(1)}$ (radius of the major base), $R^{(2)}$ (radius of the minor base)
A TRC definition extends over 2 cards in default fixed format, or over 3 cards in high-accuracy body fixed format.
Example in default fixed format:

```
*...+....1....+....2....+....3....+....4....+....5....+....6....+....7..
  TRC  102       0.0       0.0    -130.0       0.0       0.0    1000.0
               600.     150.0
* (a truncated cone 1000 cm long parallel to the z axis, with the
* larger circular base 600 cm in radius located at z = -130 and the
* smaller base 150 cm in radius located at z = 870)
```

The same example in high-accuracy body fixed format:

```
*...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+.
  TRC  102                  0.0                   0.0                -130.0
                            0.0                   0.0                1000.0
                           600.                 150.0
```

The same, in free format:

```
TRC  NewCone  0.0  0.0  -130.0  0.0  0.0  1000.0  600.  150.0
```

8.2.4.7 Ellipsoid of Revolution. Code: ELL

An ELL (Fig. 8.7) is a prolate (cigar-shaped) ellipsoid, obtainable by revolution of an ellipse around its major axis, and having any orientation in space.
Each ELL is defined by 7 numbers: $F_x^{(1)}$, $F_y^{(1)}$, $F_z^{(1)}$, $F_x^{(2)}$, $F_y^{(2)}$, $F_z^{(2)}$, (coordinates of the two foci on the major ellipsoid axis), L (length of the major axis).
An ELL definition extends over 2 cards in default fixed format, or over 3 cards in high-accuracy body fixed format.
Example in default fixed format:

```
*...+....1....+....2....+....3....+....4....+....5....+....6....+....7..
  ELL  003    -400.0       0.0       0.0     400.0       0.0       0.0
              1000.
* (an ellipsoid obtained by revolution around the x axis of an ellipse
*  centred at the origin, with major axis parallel to x 1000 cm long and
*  minor axis 600 cm long).
```

The same example in high-accuracy body fixed format:

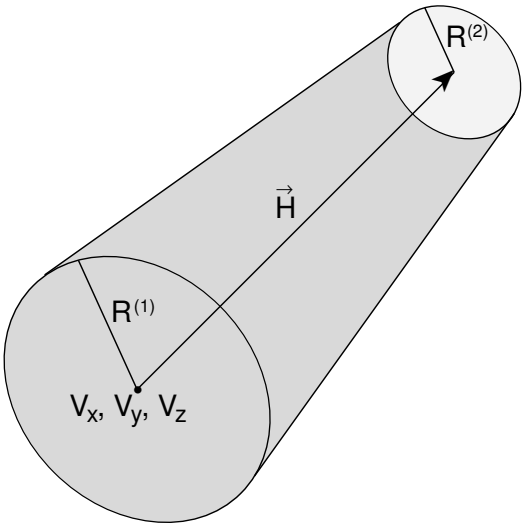

Fig. 8.6: Truncated Right Angle Cone (TRC)

```
*...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+.
  ELL  003               -400.0                 0.0                   0.0
                          400.0                 0.0                   0.0
                          1000.
```

The same example in free format:

```
ELL   TheCigar   -400.0   0.0   0.0   400.0   0.0   0.0  1000.0
```

8.2.4.8 Right Angle Wedge. Code: `WED` *or* `RAW`

A `WED` (Fig. 8.8) is the half of a `BOX` (see), cut by a plane passing through its centre and through four corners. Its use, like that of the `BOX`, is now mostly superseded by the availability of infinite planes (`XYP, XZP, YZP` and `PLA`).
A `WED` is defined by 12 numbers: V_x, V_y, V_z (coordinates of one of rectangular corners), H^1_x, H^1_y, H^1_z, H^2_x, H^2_y, H^2_z, H^3_x, H^3_y, H^3_z (x-, y- and z- components of three mutually *perpendicular* vectors corresponding to the height, width and length of the wedge). Note that it is the user's responsibility to ensure perpendicularity. This is best attained if the user has chosen high-accuracy input format (`IDBG = -10` or `-100` in the CG Title card, see 8.2.2), or free format, and the value of each vector component is expressed with the largest available number of significant digits.
The face defined by vectors 1 and 3 and that defined by vectors 2 and 3 are rectangular; the two faces defined by vectors 1 and 2 are triangular; the fifth face is rectangular with two edges parallel to vector 3 and two edges parallel to the hypotenuse of the triangle made by vectors 1 and 2.
A `WED` definition extends over 2 cards in default fixed format, or over 4 cards in high-accuracy body fixed format.
Example in default fixed format:

```
*...+....1....+....2....+....3....+....4....+....5....+....6....+....7..
  WED   97       0.0       0.0       0.0 7.0710678 7.0710678       0.0
         -14.142136 14.142136       0.0       0.0       0.0      30.0
* (the bottom half of a parallelepiped with a corner on the origin,
*  with edges 10, 20 and 30 cm long, rotated counterclockwise by 45
*  degrees in the x-y plane)
```

The same example in high-accuracy format:

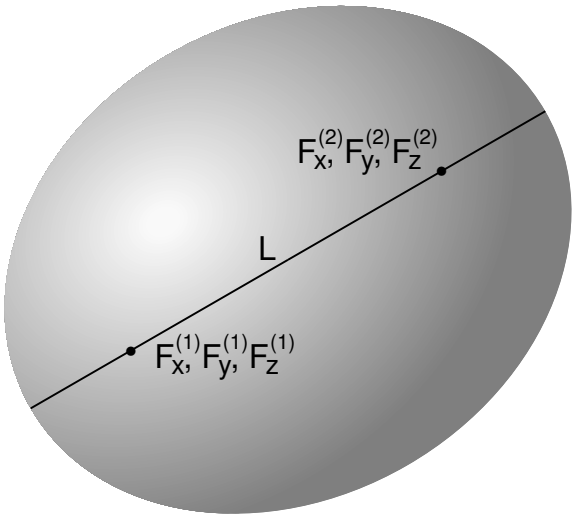

Fig. 8.7: Ellipsoid of Revolution (ELL)

```
*...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+.
  WED   97                  0.0                   0.0                  0.0
              7.071067811865475     7.071067811865475                  0.0
             -14.14213562373095     14.14213562373095                  0.0
                               0.0                   0.0                 30.0
```

The same body, in free format:

```
WED  halfbox1  0.0  0.0  0.0  7.071067811865475  7.071067811865475
     0.0  -14.14213562373095  14.14213562373095  0.0  0.0  0.0  30.0
```

8.2.4.9 Arbitrary Convex Polyhedron. Code: ARB

An ARB (Fig. 8.9) is a portion of space bounded by 4, 5 or 6 plane faces. Its use is rather complicated and is now superseded by the availability of infinite planes (XYP, XZP, YZP and PLA). For completeness, however, a description of input will be reported here.

Assign an index (1 to 8) to each vertex. For each vertex, give the x, y, z coordinates on 4 cards (8 cards in high-accuracy body fixed format), with 6 numbers on each card (3 numbers per card in high-accuracy format):

$$\begin{array}{cccccc} V_x^{(1)}, & V_y^{(1)}, & V_z^{(1)}, & V_x^{(2)}, & V_y^{(2)}, & V_z^{(2)}, \\ V_x^{(3)}, & V_y^{(3)}, & V_z^{(3)}, & V_x^{(4)}, & V_y^{(4)}, & V_z^{(4)}, \\ V_x^{(5)}, & V_y^{(5)}, & V_z^{(5)}, & V_x^{(6)}, & V_y^{(6)}, & V_z^{(6)}, \\ V_x^{(7)}, & V_y^{(7)}, & V_z^{(7)}, & V_x^{(8)}, & V_y^{(8)}, & V_z^{(8)} \end{array}$$

The vertices not appearing in face descriptions are ignored, but eight must always be supplied. The above vertex cards are followed by one card describing the faces (two cards in high-accuracy format). Each face is described by a four-digit number in floating point format, such that each digit gives the indices of the three or four vertex points of that face. These indices must be entered in the same order for each face (i.e., all clockwise or all counterclockwise).

When the number of the faces is less than 6, the remaining face description(s) must be zero, and must appear at the end of the list. If a face has three vertices, the omitted position may be either 0 or a repeat of one of the other vertices.

An ARB definition extends over 5 cards in default fixed format, or over 10 cards in high-accuracy body fixed format.

Example in default fixed format:

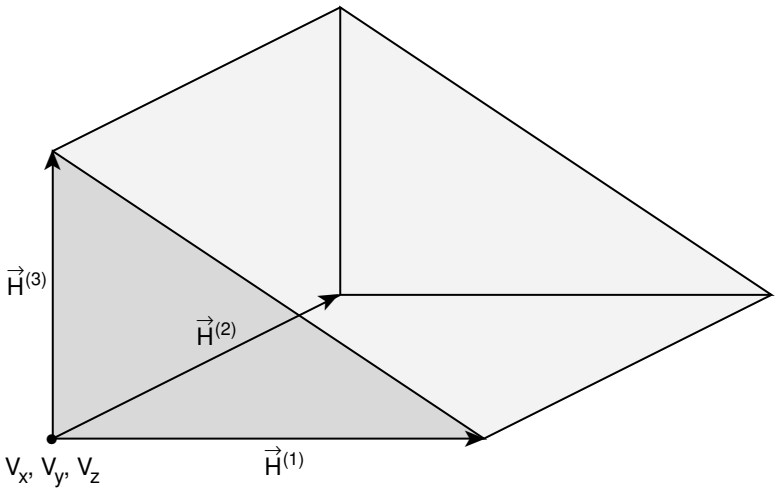

Fig. 8.8: Right Angle Wedge (WED or RAW)

```
*...+....1....+....2....+....3....+....4....+....5....+....6....+....7..
  ARB    5     -44.2     -36.5    3572.0     -44.2     -33.5    3572.0
               -37.1     -31.0    3572.0     -37.1     -39.0    3572.0
               -44.2     -36.5       0.0     -44.2     -33.5       0.0
               -37.1     -31.0       0.0     -37.1     -39.0       0.0
               1234.     1562.     5876.     1485.     4378.     6732.
* (a right prism of trapezoidal base, of height 3572 cm parallel to the
*  z axis)
```

The same example in high-accuracy body fixed format:

```
*...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+.
  ARB    5                -44.2                -36.5               3572.0
                          -44.2                -33.5               3572.0
                          -37.1                -31.0               3572.0
                          -37.1                -39.0               3572.0
                          -44.2                -36.5                  0.0
                          -44.2                -33.5                  0.0
                          -37.1                -31.0                  0.0
                          -37.1                -39.0                  0.0
                           1234.               1562.                5876.
                           1485.               4378.                6732.
```

The same example in free format:

```
ARB oddbody -44.2 -36.5 3572.0 -44.2 -33.5 3572.0 -37.1 -31.0 3572.0 -37.1
        -39.0 3572.0 -44.2 -36.5 0.0 -44.2 -33.5 0.0 -37.1 -31.0 0.0 -37.1
        -39.0 0.0 1234. 1562. 5876. 1485. 4378. 6732.
```

8.2.4.10 Infinite half-space delimited by a coordinate plane. **Code:** `XYP`, `XZP`, `YZP`

There are 4 kinds of infinite half-spaces. Three of them are delimited by planes perpendicular to the coordinate axes:

1. Delimited by a plane perpendicular to the x-axis. **Code:** `YZP`
2. Delimited by a plane perpendicular to the y-axis. **Code:** `XZP`
3. Delimited by a plane perpendicular to the z-axis. **Code:** `XYP`

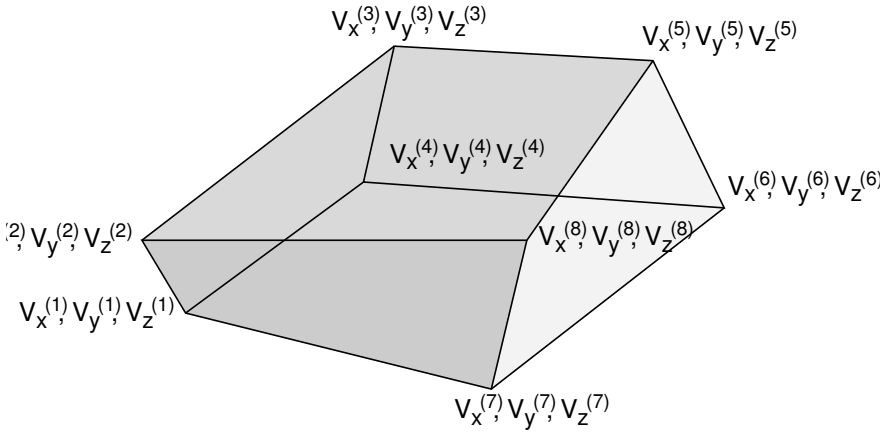

Fig. 8.9: Arbitrary Convex Polyhedron (ARB)

Each of these half-spaces is defined by a single number:
V_x (resp. V_y, or V_z), the coordinate of the plane on the corresponding perpendicular axis.
The half-space "inside the body" is the locus of points for which x $<$ V_x (resp. y $<$ V_y, or z $<$ V_z).
An `YZP, XZP, XYP` (Fig. 8.10) definition, in fixed format, extends always over a single card.
Example in default fixed format:

```
*...+....1....+....2....+....3....+....4....+....5....+....6....+....7..
  YZP   71     -44.2
  XZP   72     108.0
  XYP   73       0.0
* (respectively, all points having x < -44.2, those having y < 108,
* and those having z < 0)
```

The same example in high-accuracy body fixed format:

```
*...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+.
  YZP   71                 -44.2
  XZP   72                 108.0
  XYP   73                   0.0
```

The same example in free format:

```
YZP  horizPla  -44.2
XZP  vertPla1   108.0
XYP  vertPla2     0.0
```

8.2.4.11 Generic infinite half-space. Code: `PLA`

Each `PLA` (Fig. 8.11) is defined by 6 numbers: H_x, H_y, H_z (x-, y- and z- components of a vector of arbitrary length perpendicular to the plane), V_x, V_y, V_z (coordinates of any point lying on the plane).
The half-space "inside the body" is that from which the vector is pointing (i.e., the vector points "outside").
A `PLA` definition extends over a single card in default fixed format, and over two cards in high-accuracy body fixed format.
Example in default fixed format:

```
*...+....1....+....2....+....3....+....4....+....5....+....6....+....7..
  PLA   74       0.0       1.0       1.0     200.0    -300.0     240.0
*  (all points "below" a plane at 45 degrees in the y-z projection which
*  passes through the point x=200, y=-300, z=240 - note that for such a
*  plane the x value of the point is completely irrelevant - )
```

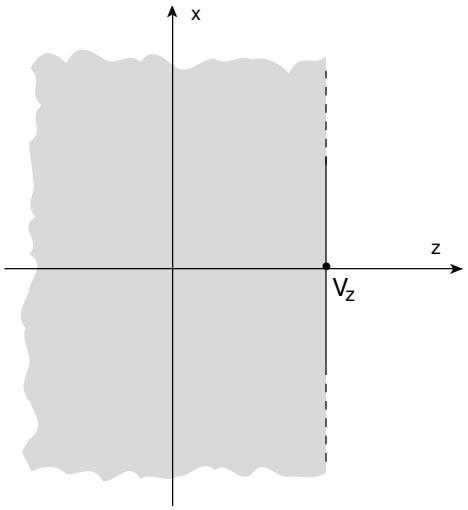

Fig. 8.10: Infinite half-space delimited by a plane perpendicular to the z axis (XYP)

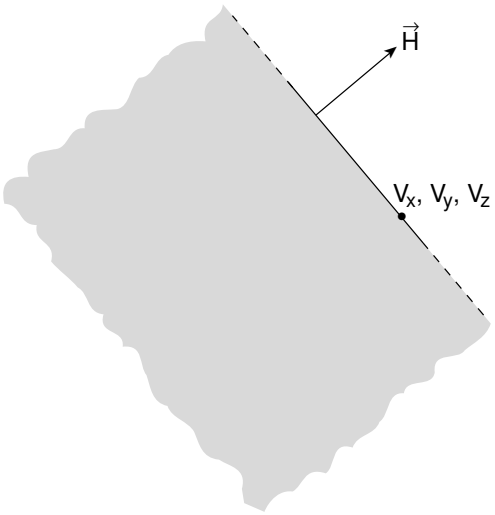

Fig. 8.11: Infinite half-space delimited by a generic plane (PLA)

The same example in high-accuracy body fixed format:

```
*...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+.
  PLA   74                   0.0                   1.0                   1.0
                           200.0                -300.0                 240.0
```

The same plane in free format:

```
PLA  tiltPla   0.0  1.0  1.0  200.0  -300.0  240.0
```

8.2.4.12 Infinite Circular Cylinder parallel to a coordinate axis. Code: XCC, YCC, ZCC

A XCC (YCC, ZCC) is an infinite circular cylinder parallel to the x (y, z) axis.
Each XCC (YCC, ZCC) (Fig. 8.12) is defined by 3 numbers: A_y, A_z (A_z, A_x for YCC, A_x, A_y for ZCC) (coordinates of the cylinder axis), R (cylinder radius)
An XCC (YCC, ZCC) definition, in fixed format, extends always over one single card.
Example in default fixed format:

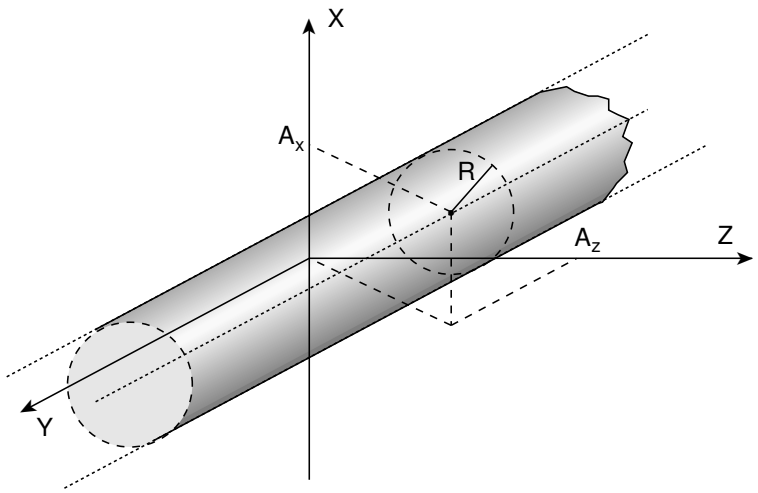

Fig. 8.12: Infinite Circular Cylinder parallel to the y-axis (YCC)

```
*...+....1....+....2....+....3....+....4....+....5....+....6....+....7..
  XCC  013    -480.0      25.0     300.0
* (an infinite cylinder of radius 300 cm, with axis defined by y=-480,
*  z=25)
  ZCC  014       0.0       0.0       2.5
* (an infinite cylinder of radius 2.5 cm, with axis equal to the z axis)
```

The same example in high-accuracy body fixed format:

```
*...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+.
  XCC  013                -480.0                  25.0                 300.0
  ZCC  014                   0.0                   0.0                   2.5
```

The same, in free format:

```
 XCC  Column  -480.0  25.0  300.0
 ZCC  Tunnel   0.0   0.0    2.5
```

8.2.4.13 Infinite Elliptical Cylinder parallel to a coordinate axis. Code: XEC, YEC, ZEC

A XEC (YEC, ZEC) is an infinite elliptical cylinder parallel to the x (y, z) axis, and with the axes of the ellipse parallel to the other two coordinate axes.
Each XEC (YEC, ZEC) is defined by 4 numbers: A_y, A_z (A_z, A_x for YEC, A_x, A_y for ZEC) (coordinates of the cylinder axis), L_y, L_z (L_z, L_x for YEC, L_x, L_y for ZEC) (semiaxes of the ellipse).
A XEC (YEC, ZEC) definition extends over one single card in default fixed format, and over two cards in high-accuracy body fixed format.
Example in default fixed format:

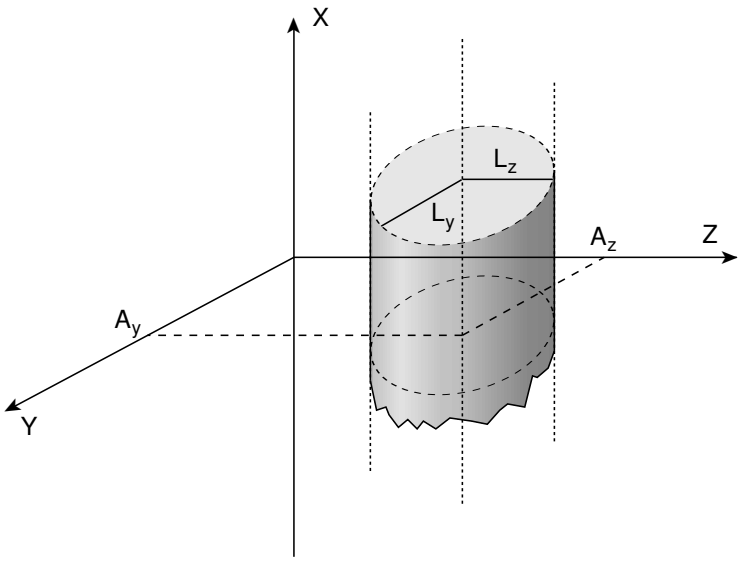

Fig. 8.13: Infinite Elliptical Cylinder parallel to the x-axis (XEC)

```
*...+....1....+....2....+....3....+....4....+....5....+....6....+....7..
  ZEC  101      15.0     319.0      33.0      80.0
* (an infinite elliptical cylinder, centred on x=15, y=319, with the
* ellipse minor semi-axis parallel to x, 33 cm long, and the major
* semi-axis parallel to y, 80 cm long)
```

The same example in high-accuracy body fixed format:

```
*...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+.
  ZEC  101                 15.0                319.0                 33.0
                           80.0
```

The same body, described in free format:

```
ZEC  ChimneyA  15.0  319.0  33.0  80.0
```

NOTE

> *Whenever it is possible, the following bodies should be preferred:*
> PLA, RPP, SPH, XCC, XEC, XYP, XZP, YCC, YEC, YZP, ZCC, ZEC
> *These should indeed make tracking faster, since for them extra coding ensures that unnecessary boundary intersection calculations are avoided when the length of the next step is smaller than the distance to any boundary of the current region.*

8.2.5 Body END card

Body definitions must be terminated by a card with the string END (in column 3–5 if fixed format is used).

8.2.6 Region data

The various regions are described in terms of differences, intersections and unions of bodies. As in the case of body description, the user has the choice between free format and two kinds of fixed format. One of the latter is the traditional format used by the original Combinatorial Geometry as implemented for example in MORSE. The other fixed format is similar to it, but has been extended to allow body numbers larger than 10000. Both fixed formats are now superseded by the more convenient free region input format, recently introduced. Free format is based on body mnemonic "names" instead of sequential numerical identifiers and allows the use of parentheses to perform more complex boolean operations. However, the two fixed formats are still available for back compatibility reasons. With any input format, a line having an asterisk (*) in column 1 is treated as a comment card.

8.2.6.1 Fixed format region input

Each region is described as a combination of one or more bodies, by means of the three operator symbols: - + and OR
referring respectively to the boolean operations of subtraction (or complement), intersection and union.

Each body is referred to by its sequential number in the body description table (see 8.2.3).
Input for each region extends on one or more cards, in a format which depends on the value of IDBG on the Geometry Title card (see 8.2.2).
If IDBG = 0, 10 or -10, region input format is (2X,A3,I5,9(A2,I5));
if IDBG =100 or -100, region input format is (2X,A3,I5,7(A2,I7));
where the 3 characters in columns 3–5 are:

- on the first input card of a given region, an arbitrary non-blank string chosen by the user (it can be used, together with optional comment cards, to help identifying the region or the region material). Note that regions are identified in the code by an integer number corresponding to the order of their input definition: therefore it can be useful (but not mandatory) to have that number appearing in the string. For instance, if the 5th region is air, it could be labelled AI5.
- on all continuation cards, columns 3-5 must be blank.

the integer in columns 6–10 (variable NAZ):

- is the number of regions which can be entered by a particle leaving any of the bodies defined for the region being described (leave blank in continuation cards). The NAZ number is used to allocate memory

for this so-called "contiguity list", and it is not essential that it be exact (if left blank, it is set to 5). Any number is accepted, but if the final sum of these integers is close to the actual sum, tracking speed can be slightly improved.

in columns 11-73:

– alternate as many 2-character fields ('OR' or blank) and integer fields (body numbers preceded by + or − sign), as are needed to complete the description of that region (see below for an explanation of symbol meaning). If one card is not sufficient, any number of continuation cards can be added (identified by a blank field in column 3–5).

8.2.6.2 Meaning of the + − OR operators

If a body number appears in a zone description preceded by a + operator, it means that the zone being described is wholly contained *inside* the body.
If a body number appears in a zone description preceded by a − operator, it means that the zone being described is wholly *outside* the body.
Obviously, in the description of each region the symbol + must appear at least once.
Examples:

```
*...+....1....+....2....+....3....+....4....+....5....+....6....+....7..
  BA1    4     +7     +3
* (the above region is the part of space common to body 7 and 3)
  MU2    7     +3     -4     -7    +20
* (the above region is the part of space common to body 3 and 20,
* excluding however that which is inside body 4 and that which is
* inside 7)
  AIR    5    +19
* (the latter region coincides entirely with body 19)
```

In some instances a region may be described in terms of subregions, lumped together. The OR operator is used as a boolean union operator in order to combine subregions (partially overlapping or not). Subregions (also called "zones" in this manual) are formed as intersections or differences as explained above, and then the region is formed by a union of these subregions. When OR operators are used there are always two or more of them, and they refer to all body numbers following them until the next OR or the end of the region description, each body being preceded by its + or − sign.
Examples:

```
*...+....1....+....2....+....3....+....4....+....5....+....6....+....7...
  SA7   11OR   +4     +6     -7     -8OR   +6     +3    -21
*         <---- first subregion ----->< second subregion ->
  G18    2OR   +9OR  +15OR   +1OR   +8     -2OR   +8     -3OR   +8    +18
*         < 1st >< 2nd >< 3rd ><--- 4th ----><--- 5th ----><--- 6th ---->
          OR  +12    -10    -11    -13    -14
*< blank ><---- 7th and last subregion -----> (continuation line)
```

8.2.6.3 Free format region input

Each region is described as a combination of one or more bodies, by means of the three operator symbols:

−		+	and	|

referring respectively to the boolean operations of subtraction (or complement), intersection and union.
Each body is referred to by its "name" (an alphanumeric string of up to 8 characters, the first character being alphabetical) in the body description table (see the description of free format body input in 8.2.3.2).
Input for each region starts on a new line and extends on as many continuation lines as are needed. It is of the form:

	REGNAME NAZ *boolean-zone-expression*
or	REGNAME NAZ | *boolean-zone-expression* | *boolean zone expression* | ...

where REGNAME, NAZ and the remaining part are separated by one or more blanks.

REGNAME is the region "name" (an arbitrary unique alphanumeric character string chosen by the user). The

region name must begin by an alphabetical character and must not be longer than 8 characters.

`NAZ` is an integer indicating (approximately) the number of regions which can be entered by a particle leaving any of the bodies appearing in the region description that follows. The `NAZ` number is used to allocate memory for this so-called "contiguity list", and it is not essential that it be exact. Any number is accepted, but if the final sum of these integers is close to the actual sum, tracking speed can be slightly improved. If the input value of `NAZ` is equal to 0, it is set equal to 5.

"boolean-zone-expression" is a sequence of one or more body names preceded by the operators `+` (intersection) or `-` (complement or subtraction). A zone expression can be contained inside one or more sets of left and right parentheses. Several zone expressions can be combined by the union operator `|` (corresponding to `OR` in fixed format input).
When `|` operators are used there are always two or more of them, and they refer to all bodies following them until the next `|` or the end of the region description, each body being preceded by its `+` or `-` sign.
In evaluating the expressions, the highest operator precedence is given to parentheses (the most inner ones first), followed by the `|` operator. In each zone expression, at least one body name preceded by `+` must be present. If one line is not sufficient, any number of continuation lines can be added. Blanks are ignored.
Region description ends with a line containing the single string `END`.

8.2.6.4 Meaning of the + - | operators

If a body name is preceded by a `+` operator in an expression describing a zone (or a zone component surrounded by parentheses) it means that the zone or zone component being described is wholly contained *inside* the body (boolean intersection).

If a body name is preceded by a `-` operator in an expression describing a zone (or a zone component surrounded by parentheses) it means that the zone or zone component being described is wholly *outside* the body (boolean complement).

Obviously, in the description of each region the symbol `+` must appear at least once. The same is true for each zone (subregion delimited by `|` operators) and for each zone component (subzone delimited by parentheses)

Examples of regions consisting of a single zone:

```
lastlayr   4  +bigball  +slab
* Region "lastlayr" is the part of space common to body "bigball" and body
* "slab"

MidVacuu  7 +cylind2   -slit  -sqrhole +Plane1
* Region "MidVacuu" is the part of space common to bodies "cylind2" and
* "Plane1", excluding however that which is inside body "slit" and that which
* is inside body "sqrhole"

H2Osphere   5   +marble
* Region "H2Osphere" coincides entirely with body "marble"
```

Examples of regions consisting of the union of several zones, possibly (but not necessarily) partially overlapping:

```
Corners  6 | +dice +topNorth | +dice +topEast | +dice +topSouth |
+dice +topWest | +dice +botNorth | +dice +botEast | +dice +botSouth | +dice
+botWest
* Region "Corners" is made of the 8 corners of a cube, each of which is
* obtained by the intersection of a cubic body "dice" and a tilted plane
* described by vector pointing to the centre of the cube

twoparts 0 | +leftpart -outerbox | +rghtpart +topplane
* Region "twoparts" is the sum of two parts of space: the space points which
* are inside body "leftpart" but not inside body "outerbox", plus those which
* are common to bodies "rghtpart" and "topplane"
```

Examples of a region defined as a single zone by means of parentheses:

```
AirAroun 5  + tunnel + Column - (+outrpipe-innrpipe)
* Region "AirAroun" contains the space points located inside the intersection
* of body "tunnel" and body "Column", with the exception of those which are
* inside body "outrpipe" but not inside body "innrpipe"

CmplexRg | +longcyl + (+shortcyl +vertPla1 -vertPla2) | (+Brick | + ceeling
- floor)
* Region CmplexRg is the union of two zones. The first zone is the
* intersection of body "longcyl" with a portion of space contained inside
* both bodies "shortcyl" and "vertPla1" but not inside body "vertPla2".
* The second zone is the union of the space points inside body "Brick"
* and the space points contained by body  "ceeling" but located outside
* body "floor".
```

8.2.7 Region END card

Region data must be terminated by a card with the string END (in column 3–5 if format is fixed).

8.2.8 Region Volumes

This is an optional set of cards which must be input if (and only if) flag IVOPT in the CG Title card has been given a value 3 (see 8.2.2). As many volume definition cards must be given as is needed to input a volume for every region. The input variable is an array VNOR(I) = volume of the Ith region. The format is (7E10.5). Volume data are used by FLUKA only to normalise energy or star densities per region, requested by the SCORE command (p. 195).

8.2.9 LATTICE card

This is an optional card for modular geometries. Its use needs some more effort and preparation.

The basic unit of the geometry, composed by an arbitrary number of regions, must be described in full detail in the body and region data.

Additional body and region data must also be provided to describe "container" regions representing the "boxes", or lattice cells, wherein the basic unit has to be replicated. No material assignment is needed for these lattice-cell regions.

A user routine (LATTIC, see 12.2.10) must be written to provide the transformation bringing from any point and direction inside each lattice cell to the corresponding point and direction in the basic unit.

The LATTICE card itself identifies the lattice cells and establishes the correspondence between region number and lattice cell number, where the region number is the sequential number in the region table, and the lattice cell number is that used in the tracking to address the transformation routine, and is chosen by the user. Contiguous numbering is recommended for memory management reasons, but is not mandatory. Non-contiguous numbering can be done using several LATTICE cards.

In the LATTICE card, the meanings of the WHAT parameters are:

WHAT(1) = "Container-region" number of the first lattice cell
("*From region WHAT(1)...*")
Default : No default

WHAT(2) = "Container-region" number of the last lattice cell
("*... to region WHAT(2)...*")
Default = WHAT(1)

WHAT(3) = step length in assigning "Container-region" numbers
("*... in steps of WHAT(3)*")

Default = 1.0

WHAT(4) = lattice number of the first lattice cell, assigned to region WHAT(1)
Default : No default

WHAT(5) = lattice number of the last lattice cell, assigned to region WHAT(2)
Default : No default

WHAT(6) = step length in assigning cell numbers
("*... in steps of WHAT(6)*")
Default = 1.0

A single geometry can be a mixture of modular areas, described by lattices, and "normal" areas, described by standard regions. Many different LATTICE cards may be issued in the same geometry, when different symmetries are present in different areas. In principle, any analytical symmetry transformation can be implemented (rotation, translation, reflection, or combination of these).
Care must be taken to ensure that any region in the basic unit is fully contained (after coordinate transformation) in any lattice cell belonging to its symmetry transformation. Regions falling across two different lattice cells would lead to unpredictable behaviour.
The basic unit does not need necessarily to describe a "real" geometry region, but can as well be used only as a prototype to reproduce in any number of "copies".

NOTE: The lattice cell regions do not need to be included in the other input option cards. Materials, thresholds, etc., must be assigned *only* to the regions contained in the basic unit. Of course, this implies that all copies of a same basic unit share the same material, setting and biasing properties.

IMPORTANT: If the geometry is being described in free format, using alphanumeric names as body and region identifiers, names **must** be used also in the LATTICE card(s) both for regions and lattices.

8.2.10 GEOEND card

A card with the string GEOEND in column 1–6 must terminate the combinatorial geometry input (see p. 123).

The GEOEND card can be used also to activate the *geometry debugger*, using the WHAT and SDUM parameters. In this case, a second GEOEND card (continuation) may be necessary. It is recommended that a STOP card should follow immediately, avoiding to start transport when debugging is completed.

First GEOEND card:

WHAT(1) = $\mathbf{X}_{\max}$ of the geometry region to be debugged (no default)

WHAT(2) = $\mathbf{Y}_{\max}$ of the geometry region to be debugged (no default)

WHAT(3) = $\mathbf{Z}_{\max}$ of the geometry region to be debugged (no default)

WHAT(4) = $\mathbf{X}_{\min}$ of the geometry region to be debugged (no default)

WHAT(5) = $\mathbf{Y}_{\min}$ of the geometry region to be debugged (no default)

WHAT(6) = $\mathbf{Z}_{\min}$ of the geometry region to be debugged (no default)

SDUM = DEBUG to activate the debugger, otherwise must be left blank

Second (optional) GEOEND card:

WHAT(1) = Number of mesh intervals in the **X**-direction between $\mathbf{X}_{\min}$ and $\mathbf{X}_{\max}$
Default = 20.0

WHAT(2) = Number of mesh intervals in the **Y**-direction between $\mathbf{Y}_{\min}$ and $\mathbf{Y}_{\max}$
Default = 20.0

WHAT(3) = Number of mesh intervals in the **Z**-direction between $\mathbf{Z}_{\min}$ and $\mathbf{Z}_{\max}$
Default = 20.0

WHAT(4) – **WHAT(6)**: not used

SDUM = "&" in any position in column 71 to 78 (or in the last field if free format is used)

Default (option GEOEND not given): **not allowed!** GEOBEGIN and GEOEND must always be present.

See the Notes to GEOEND option (7.32) for more details and instructions.

8.2.11 Voxel geometry

It is possible to describe a complex geometry in terms of "voxels" (tiny *identical* parallelepipeds forming a 3-dimensional grid). In principle this can be done with any geometry but it is especially useful when translating a CT scan of a human body into a dosimetry phantom [175]. Therefore, we will use loosely the word "organ" to indicate a contiguous group of voxels (or even more than one group) made of the same material. The code handles each organ as a Combinatorial Geometry region, possibly in addition to other conventional "non-voxel" regions defined by the user, and assigns automatically to each organ a new region number.

To describe a voxel geometry, the user must:

1. Assign an organ to each voxel. Each organ is identified by a unique integer ≤ 32767. The numbering does not need to be contiguous, i.e., gaps in the numbering sequence are allowed. One of the organs must have number 0 and plays the role of the medium surrounding the voxels (usually vacuum or air). The assignment is done via a special file where the organ corresponding to each voxel is listed sequentially in Fortran list-oriented format, with the x coordinate running faster than y, and y running faster than z. In practice the file is always written by a program similar to the one reported below. The user will need to modify the values of the parameters `DX, DY DZ, NX, NY, NZ` (respectively voxel size and number of voxels for each coordinate), and possibly some other more trivial things (file names, title, reading from the original CT scan file).
   The following program takes also care of recompacting the original organ numbers by eliminating all gaps in the sequence, and writes a translation table to the screen:

```
WRITE(*,'(A,2I10)')' New number, old number: ', NO, IC
```

   After having modified the program (assumed to be in a file `writegolem.f`), compile it:

```
$FLUPRO/flutil/fff writegolem.f
```

   link it with the FLUKA library:

```
$FLUPRO/flutil/lfluka -o writegolem writegolem.o
```

   execute it:

```
./writegolem
```

   The result will be a file `golem.vxl` (or equivalent name chosen by the user) which will be referred to by a special command line in the geometry input (see 2 below).

```
      PROGRAM WRITEGOLEM

      INCLUDE '(DBLPRC)'
      INCLUDE '(DIMPAR)'
      INCLUDE '(IOUNIT)'
*  COLUMNS: FROM LEFT TO RIGHT
*  ROWS: FROM BACK TO FRONT
*  SLICES: FROM TOP TO BOTTOM
      PARAMETER ( DX = 0.208D+00 )
      PARAMETER ( DY = 0.208D+00 )
      PARAMETER ( DZ =-0.8D+00   )
```

```
      PARAMETER ( NX = 256 )
      PARAMETER ( NY = 256 )
      PARAMETER ( NZ = 220 )
      DIMENSION GOLEM(NX,NY,NZ)
      INTEGER*2 GOLEM
      CHARACTER TITLE*80
      DIMENSION IREG(1000), KREG(1000)
      INTEGER*2 IREG, KREG
*
      CALL CMSPPR
      DO IC = 1, 1000
         KREG(IC) = 0
      END DO
      OPEN(UNIT=30,FILE='ascii_segm_golem',STATUS='OLD')
      READ(30,*) GOLEM
      NO=0
      MO=0
      DO IZ=1,NZ
         DO IY=1,NY
            DO IX=1,NX
               IF (GOLEM(IX,IY,IZ) .GT. 0) THEN
                  IC = GOLEM(IX,IY,IZ)
                  MO = MAX (MO,IC)
                  DO IR=1,NO
                     IF (IREG(IR) .EQ. IC) GO TO 1000
                  END DO
                  NO=NO+1
                  IREG(NO)=IC
                  KREG(IC)=NO
                  WRITE(*,'(2I10)')' New number, old number: ', NO, IC
 1000             CONTINUE
               END IF
            END DO
         END DO
      END DO
*     NO = number of different organs
*     MO = max. organ number before compacting
      WRITE(*,*) ' NO,MO',NO,MO
      OPEN(UNIT=31,FILE='golem.vxl',STATUS='UNKNOWN',FORM='UNFORMATTED')
      TITLE = 'Golem'
      WRITE(31) TITLE
      WRITE(31) NX,NY,NZ,NO,MO
      WRITE(31) DX,DY,DZ
      WRITE(31) GOLEM
      WRITE(31) (KREG(IC),IC=1,MO)
      STOP
      END
```

2. Prepare the usual FLUKA input file. The geometry must be written like a normal Combinatorial Geometry input (in any of the allowed formats, as part of the normal input stream or in a separate file), but in addition must include:
   - A VOXELS card as a first line, before the Geometry title card (8.2.2), with the following information:
     WHAT(1), WHAT(2), WHAT(3) = x, y, z coordinates chosen as the origin of the "voxel volume", i.e., of a region made of a single RPP body (8.2.4.1) which contains all the voxels
     WHAT(4), WHAT(5), WHAT(6): not used
     SDUM = name of the voxel file (extension will be assumed to be .vxl)
   - The usual list of NB bodies, not including the RPP corresponding to the "voxel volume" (see VOXELS card above). This RPP will be generated and added automatically by the code as the (NB+1)th body, with one corner in the point indicated in the VOXELS card, and dimensions NX*DX, NY*DY and NZ*DZ as read from the voxel file.

- The usual region list of NR regions, with the space occupied by body NB+1 (the "voxel volume") subtracted. In other words, the NR regions listed must cover the whole available space, except the space corresponding to the "voxel volume". This is easily obtained by subtracting body NB+1 in the relevant region definitions, even though this body is not explicitly input at the end of the body list. The code will automatically generate and add several regions:
  - a region NR+1: this is a sort of "cage" for all the voxels. Nothing (energy etc.) should ever be deposited in it: the user shall assign vacuum to it.
  - a region NR+2 containing all voxels belonging to organ number 0. There must be at least 2 of such voxels, but in general they should be many more. Typical material assignment to region NR+2 is air.
  - NO additional regions, where NO = number of non-zero organs:

region NR+3	corresponding to organ 1
region NR+4	corresponding to organ 2
.....	
region NR+2+NO	corresponding to organ NO

The assignment of materials shall be made by command ASSIGNMAt (p. 62) (and in a similar way other region-dependent options) referring to the first NR regions in the usual way, and to the additional regions using the correspondence to organs as explained above.

Chapter 9

Output

The output of FLUKA consists of:

- a main (standard) output, written on logical output unit LUNOUT (predefined as 11 by default)
- a scratch file, of little interest to the user, written on output unit LUNGEO (16 by default). However, if the rfluka script is used to run FLUKA, this file is automatically deleted by the script at the end of the run
- a file with the last random number seeds, unit LUNRAN (2 by default)
- a file of error messages (if any), unit LUNERR (15 by default)
- any number (including zero) of estimator output files. Their corresponding logical unit number is defined by the user: in case the number chosen coincides with one of the above, in particular LUNOUT, estimator formatted output will appear as part of the corresponding output stream. However, this is not recommended, and it is not allowed anyway in the case of unformatted output. Generally, the user can choose between formatted and unformatted output. Only formatted output is presented here, while unformatted output is described at the end of each option description
- possible additional output generated by the user in any user routine, in particular USROUT (see 12.2.28, p. 324)

9.1 Main output

The standard, or main, output is made of several different parts:

- A **banner page**
- The **FLUKA license**
- A **header** with the FLUKA version and the time when the output was printed
- A straight **echo of the input cards**.
  Each input line is echoed in output, but not character by character. The input WHATs and SDUMs are read, and then written with a different format. Any alignment error shows up as a number or a character string different from the one intended: therefore in case of problems checking this part of the output is more effective than checking the input itself.
  Comments are reproduced in output, with the exception of in-line comments preceded by an exclamation mark (!)
- **Geometry output** (if not redirected to a separate file, see GEOBEGIN, Note 1). The geometry output (which is part of the standard output by default, but can be re-directed to a separate file) begins with an echo of the geometry title and the value of the two input variables IVOPT (Input Volume OPTion) and IDBG (in the original CG a debugging flag, but now used to select various format lengths). Then there in an echo of the body and region input, including comment lines, and some lines left over from the original MORSE CG, but which are of little or no meaning in the context of FLUKA: for instance the arrays IR1 and IR2 (originally material and biasing assignment to regions, which in FLUKA however are not part of the geometry data). Other information concerns the memory allocation: FPD (Floating Point Data), INTEGER ARRAY, zone locations ("zone" and "region" in FLUKA have a different meaning than in MORSE). "Code zones" indicates the subregions defined by the input OR operator.
  The next sections, "Interpreted body echo" and "Interpreted region echo", show the numbers assigned by the program to bodies and regions defined by alphanumerical identifiers (if the traditional fixed format has been used, these output sections are of little interest).
  The interpreted echos are followed by the volumes used to normalise a possible output from option SCORE (p. 195). Then, for each region in whose description the OR operator is used, one line similar to the following is printed at the end of the geometry output:

```
*** Region # 2 Dnear according to    no    overlapping ORs ***
*** Region # 3 Dnear according to possible overlapping ORs ***
```

This information concerns the possibility that random seed sequences might not be reproducible, a technical issue which does not affect the quality of the results but can be important for debugging or other purposes (see a more detailed explanation in Note 4 to option GLOBAL) (p. 126).

– **Basic nuclear data used in the program**
The data reported are nuclear masses and model parameters used by the program. This part of the output is constant and does not depend on the problem input (it is printed even if the calculation is purely electromagnetic and does not depend on nuclear models).

– **Information on physical models used in the run**
The nuclear models used by FLUKA to describe intermediate nuclear effects and interactions have been continuously improved since 1989. Each improvement has been first implemented and tested as an option that the user could request via the EVENTYPE command (p. 116). These improvements are still optional in principle, although they are automatically activated if the user chooses the appropriate defaults. Depending on the latter and on the input options used, an informative message is issued concerning the presence of the following:
  – Evaporation from residual nucleus
  – Production of deexcitation gammas
  – Transport of heavy evaporation products
  – High-energy fission
  – Fermi Break-Up

– **Material quantities related to multiple scattering**
The values of various quantities used by the FLUKA multiple Coulomb scattering algorithm are printed for each material and for each type of charged particle.

– **Memory allocation information**
Starting and ending location in memory of various arrays dynamically allocated are printed at different points on main output, depending on the order of input cards.

– **Table of correspondence between materials used in the run and materials in the low-energy neutron cross-section library**
Example:

```
*** Fluka to low en. xsec  material correspondence: printed atomic densities are
    meaningless when used in a compound ***

Fluka medium     Name     Xsec medium  atomic density   Id. 1  Id. 2   Id. 3
  number                   number     ( at/(cm barn) )
      1        BLCKHOLE        0         0.0000E+00        0      0       0
      2        VACUUM       1000         0.0000E+00        0      0       0
      6        CARBON          1         0.0000E+00        6     -2     293
      7        NITROGEN        2         0.0000E+00        7     -2     293
      8        OXYGEN          3         5.3787E-05        8     16     293
     17        LEAD            5         3.2988E-02       82     -2     293
     21        ARGON           4         0.0000E+00       18     -2     293
```

Compounds are not listed in this table, since for the time being the FLUKA neutron library contains only single elements or nuclides. "FLUKA medium number" refers to the material number given by the user via WHAT(4) in option MATERIAL; "Xsec medium number" is the material index used internally by the FLUKA low-energy neutron package. Such index is assigned only to library materials actually used in the current problem, unlike "FLUKA media" which can be predefined or defined in input, without being actually assigned to any region.
Blackhole and vacuum are always assigned Xsec index 0 and 1000.
Atomic densities refer to the material in its elemental form and are printed as 0.0000E+00 if the corresponding element is used only as part of a compound.
The last 3 columns in the table are the material identifiers unique to each library material (see 10.5, p. 295 and option LOW–MAT, p. 138).

– **Information on the low-energy neutron cross-sections**
If low-energy neutrons are transported, some problem-specific information may be printed, e.g., materials for which recoil or (n,p) protons are produced explicitly and not accounted for by kerma factors (usually hydrogen and nitrogen), or materials for which pointwise cross-sections are used (see Note 4

to option LOW–NEUT, p. 141). This is followed by generic information on the neutron cross-section library used (number of energy groups and angles, number of materials, etc.).
If the user requests a more detailed printout (option LOW–NEUT, p. 140) the following information is printed, depending on the value of WHAT(4):

If **WHAT(4)** = 1.0:

For each neutron energy group:

- group energy limits
- average energies
- velocities and momenta corresponding to the group energy limits of each gamma group
- thermal neutron velocities

For each material used: availability of residual nuclei information and, for each neutron energy group:
SIGT = **total cross-section** in barn

SIGST = **"scattering" cross-section** in barn. Actually equal to $\sigma(n,n) + 2\sigma(n,2n) + 3\sigma(n,3n)$ etc.

PNUP = **upscatter probability** (can be different from zero only if the number of thermal groups is more than one)

PNABS = **Probability of Non-ABSorption** (= scattering). It is = SIGST/SIGT, and can sometimes be > 1 because of (n,xn) reactions

GAMGEN = **GAMma GENeration probability** = gamma production cross-section divided by SIGT and multiplied by the average number of gammas per (n,γ) reaction

NU*FIS = **fission neutron production** = fission cross-section divided by SIGT and multiplied by ν (nu), the average number of neutrons per fission

EDEP = **kerma contribution** in GeV per collision

PNEL, PXN, PFISS, PNGAM = **partial cross-sections** , expressed as probabilities (i.e., ratios to SIGT). In the order: non-elastic, (n,xn), fission, (n,γ)
The line: (RESIDUAL NUCLEI INFORMATIONS AVAILABLE), if present, indicates the possibility to use option RESNUCLEi with WHAT(1) = 2.0 (p. 187).

If **WHAT(4)** = 2.0:

the same as above plus:
For each material used and for each neutron energy group:
the downscattering matrix (group-to-group transfer probabilities), as in the following example:

```
                CROSS SECTIONS FOR MEDIA     4
   ..........................................................
   (RESIDUAL NUCLEI INFORMATIONS AVAILABLE)
GROUP....DOWNSCATTER MATRIX
   ..........................................................
   6....0.4927  0.0148  0.0006  0.0012  0.0017  0.0023  0.0028  0.0033
        0.0038  0.0045  0.0056  0.0070  0.0087  0.0104  0.0120  0.0134
        0.0149  0.0163  0.0175  0.0184  0.0190  0.0193  0.0193  0.0190
        0.0185  0.0178  0.0164  0.0329  0.0311  0.0278  0.0247  0.0219
        0.0198  0.0158  0.0126  0.0101  0.0112  0.0070  0.0026  0.0008
        0.0004  0.0002  0.0001  0.0000  0.0000  0.0000  0.0000  0.0000
        0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000
        0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000  0.0000
        0.0000  0.0000  0.0000
    ..........................................................
```

The above table means: after scattering in material 4 of a neutron in energy group 6, the probability of getting a neutron in the same group is 49.27 %; that to get a neutron in the following group (group 7) is 1.48 %, in group 8 is 0.06 % etc. This matrix, normalised to 1, gives the relative probability of each neutron group: but the actual probability *per collision* must be obtained by multiplying by PNABS, the scattering cross-section divided by the total cross-section and multiplied by the average number of neutrons per non absorption reaction.

neutron-to-gamma group transfer probabilities, for instance:

```
                NEUTRON TO GAMMA TRANSFERS FOR MEDIA     6
 NEUT GROUP      GAMGEN        TRANSFER PROBABILITIES
         1     1.7749E+00      0.0000  0.0000  0.0000  0.0003  0.0002  0.0004
                               0.0008  0.0015  0.0027  0.0048  0.0084  0.0144
                               0.0239  0.0378  0.0686  0.0942  0.0967  0.1125
                               0.2830  0.1249  0.0625  0.0625
              ..........................................................
```

The meaning is similar to that explained above, except that each number refers to the probability of getting a gamma in the corresponding gamma group. Again, this matrix, normalised to 1, gives the relative probability of each gamma group: but the actual probability *per collision* must be obtained by multiplying by GAMGEN, the gamma production cross-section divided by the total cross-section and multiplied by the average number of gammas per (n,γ) reaction

If **WHAT(4)** = 3.0:

the same as above plus:

For each material used and for each neutron energy group:

Cumulative scattering probabilities and scattering polar angle cosines as in the following example:

```
 1 SCATTERING PROBABILITIES AND ANGLES FOR MEDIA NUMBER     4
   GP TO GP   PROB   ANGLE    PROB   ANGLE    PROB   ANGLE
 ..........................................................
   6     6  0.8736  0.9557  0.9823  0.3741  1.0000 -0.6421
   6     7  0.4105  0.8383  0.8199  0.1057  1.0000 -0.7588
   6     8  0.4444  0.0001  0.7223  0.7747  1.0000 -0.7746
   6     9  0.4444 -0.0001  0.7223 -0.7746  1.0000  0.7746
   6    10  0.4444  0.0000  0.7223 -0.7746  1.0000  0.7746
   6    11 -1.0000  0.0000  0.0000  0.0000  0.0000  0.0000
   6    12 -1.0000  0.0000  0.0000  0.0000  0.0000  0.0000
   6    13 -1.0000  0.0000  0.0000  0.0000  0.0000  0.0000
 ..........................................................
```

The above table reports 3 discrete angle cosines (corresponding to a Legendre P5 expansion) for each group-to-group scattering combination, with the respective cumulative probabilities. For instance, the line:

```
   6     7  0.4105  0.8383  0.8199  0.1057  1.0000 -0.7588
```

means that neutron scattering from energy group 6 to group 7 has a 0.4105 probability to be at a polar angle of 33 degrees ($0.8383 = \cos 33°$); a probability $0.8199 - 0.4105 = 0.4094$ to be at $84° = \arccos(0.1057)$; and a probability $1.000 - 0.8199 = 0.1801$ to be at $139° = \arccos(-0.7588)$.

A -1.0000 probability indicates an isotropic distribution.

– **Table of available particle types**
This table presents constant properties of all particles transported by FLUKA (name, id-number, rest mass and charge), plus two columns indicating:
  – the particles which are discarded, by default (neutrinos) or on user's request (option DISCARD, p. 94)
  – the particle decay flag (see option PHYSICS with SDUM = DECAYS, p. 176)

  The available generalised particles and their id-numbers are also listed.

– **An expanded summary of the input**:
This part of output summarises the most important input data, decoded and presented in a more colloquial way.

(a) Beam properties
Information given by BEAM and BEAMPOS options (p. 64, 69): type of particle, energy/momentum, direction, etc. If a user SOURCE is used (p. 197, 314), it is indicated here

(b) Energy thresholds
Particle cut-off energies of hadrons and muons as set by default or by option PART–THR (p. 172). Neutron cut-off means the threshold between high and low-energy (multi-group) neutron treat-

ment. Low-energy neutron group cut-offs are reported by region in a separate table: see (f) below. Electron and photon cut-offs are also reported in a separate table.

(c) Termination conditions The maximum number of histories and other ending options set in card START (p. 199) are summarised here

(d) Multiple scattering (hadrons and muons) The logical flags printed are related to option MULSOPT (p. 153) with SDUM = GLOBAL or GLOBHAD. The number of single scatterings to be performed at boundary crossing is also printed.

(e) Treatment of electrons and photons, including multiple scattering. For historical reasons dating from the time when FLUKA was handling only high-energy particles, this title of this part is "Electromagnetic Showers". The logical flags which follow are related to option MULSOPT with SDUM = GLOBAL or GLOBEMF. The number of single scatterings to be performed at boundary crossing is also printed.

(f) Biasing parameters
This table reports several region-dependent biasing and cut-off parameters:
- Particle importances (set by WHAT(1) and WHAT(3) of option BIASING, p. 71)
- Russian Roulette factor (multiplicity biasing set by WHAT(2) of option BIASING)
- Cut-off group (WHAT(1) of option LOW–BIAS, p. 135)
- Group limit for non-analogue absorption (WHAT(1) of option LOW–BIAS)
- Non-analogue survival probability (WHAT(3) of option LOW–BIAS)
- Group limit for biased downscattering (WHAT(1) of option LOW–DOWN)
- Biasing downscattering factor (WHAT(2) of option LOW–DOWN)

(g) Estimators requested
For each requested estimator (USRBIN, USRBDX, USRCOLL, USRTRACK, USRYIELD, RESNUCLEi, DETECT), a complete description is printed (estimator number, particle type, defining region(s) or binning limits, number of intervals/bins, area/volume, linear/logarithmic, type of quantity etc.). If the estimator output file is formatted, the same information is printed also there in an identical format, otherwise it is available on the corresponding binary file.
Note that the estimator detectors are numbered separately according to their type (`Bdrx n. 1, Bdrx n. 2` etc.; `Binning n. 1, Binning n. 2` etc. — independent from the type of binning — `Track n. 1, Track n. 2` etc.), in the order they appear in input. The detector number can be passed (as a variable `JSCRNG` in `COMMON SCOHLP`) to the user routines `COMSCW` and `FLUSCW` (p. 304, 306), to allow different kinds of weighting on the scored quantities, depending on the detector number (see 12.2.2, 12.2.6)

(h) Materials defined and pre-defined
This table includes all materials pre-defined and not overridden by the user, plus those defined in the user input via options MATERIAL and COMPOUND (p. 142, 78), independent from the fact that they have been assigned or not to any region.
The different columns report the material number, name, atomic number Z and atomic mass A (effective Z, A for compounds), density, inelastic and elastic scattering lengths for the primary particles at the energy defined by option BEAM (p. 64) (meaningful only for primary hadrons), radiation length (value not used by the program) and inelastic scattering length for neutrons at the threshold momentum (by default 19.6 MeV unless overridden by PART–THR, p. 172).
For compounds, an insert is printed with the element composition, plus the atom fraction and partial density of each component.

(i) dE/dx tabulations (if requested, see DELTARAY, p. 89)
For each assigned material and for each charged heavy particle (hadrons, muons, recoil ions) a table is printed with the following data:
energy, unrestricted stopping power, η ($= \beta\gamma$), shell correction, restricted stopping power (according to the threshold specified by the user with DELTARAY, WHAT(1)).

(j) Other stopping power information
The following is printed for each material used:
gas pressure (if applicable), average excitation energy, effective Z/A, Sternheimer density effect parameters, delta ray production, threshold, description level for stopping power fluctuations (set with IONFLUCT, WHAT(1) and WHAT(3), p. 128), and threshold for pair production and

bremsstrahlung by heavy particles (set with PAIRBREM, p. 170).

(k) Photonuclear reaction requests
A line of information is printed for each material in which muon photonuclear interactions have been activated (see MUPHOTON, p. 156). A similar information is printed for gamma photonuclear interactions, with the PHOTONUC flag for the relevant energy ranges (see p. 174).

(l) Table of correspondence between materials and regions "Correspondence of regions and EMF-FLUKA material numbers and names"
This table corresponds to the ASSIGNMAt command (p. 62) and may be useful to check the material assignment to the various regions, as well as the regions where a magnetic field is present. The minimum step size set with option STEPSIZE (p. 200) is also reported. The last column refers to the maximum step size for charged particles.

(m) Rayleigh scattering requests
A line of information is printed for each material in which Rayleigh scattering has been activated (option EMFRAY, p. 110).

(n) Fluorescence requests
For each material for which fluorescence X-ray production has been requested, information about the relevant photoelectric cross-section edges is reported (option EMFFLUO, p. 109).

– **Random number generator calls and CPU time for some histories** During the calculation, a couple of lines is printed time and again after a certain number of histories (or after each history, depending on WHAT(5) of option START, p. 199)[1].
One of the two lines contains the number of random number generator calls, expressed in hexadecimal format.
The second line reports, in the following order: the number of primary particles handled, the number of particles still to be handled (with respect to the maximum requested by WHAT(1) of START), the number of particles which can still be handled judging from the average time spent so far, the average time per primary based on the histories already completed, and the estimated time still available with respect to WHAT(3) of START.
The sequence of random number call lines is terminated by a message (`FEEDER` is the FLUKA routine which starts every history):

`All cases handled by Feeder`

`Run termination forced from outside`
if the run has been shortened by the FLUKA "stop file" (see note 4 on option COMMENT, p. 76)

`Feeder ended due to timeout`
if the time limit has been reached — see WHAT(6) of START or system-imposed time limit

– **Results of SCORE option for all regions**
Up to 4 different quantities (energy or star density) are printed at one line per region (see SCORE, p. 195). The volume used for normalisation is also printed (equal to `1.0` unless a different value has been input by the user at the end of the geometry description, see `IVOPT = 3` in 8.2.2, p. 252). Even if SCORE has not been requested a line of zeros is printed for each region.

– **Statistics of Coulomb scattering**
The number of scatterings which were not performed or were performed without LDA (Lateral Displacement Algorithm) because they failed to satisfy Molière's conditions is reported here. This number is usually very small compared to the total number of scatterings, also reported.
If single scatterings have been activated, their number is also printed.

– **Russian Roulette/Splitting counters** (if requested, see BIASING, p. 71)
If the BIASING option has been used with SDUM = PRINT, the following statistics is printed for each region:
`N. of RR` ⟹ Number of Russian Roulette (RR) operations made on particles *entering* that region
`<Wt>  in` ⟹ Average weight of particles submitted to RR when entering the region
`<Wt> kil` ⟹ Average weight of particles killed by RR when entering the region
`N. of Sp` ⟹ Number of splitting operations made on particles *entering* the region
`<wt>  in` ⟹ Average weight of particles splitted when entering the region

[1] Occasional warning messages printed during particle transport are found between the lines, especially if photonuclear reactions have been activated: they have mainly a temporary debugging purpose and should be ignored

`<Wt> out` $\Longrightarrow$ Average weight of particles after being splitted when entering the region
Separate counters are printed for hadrons/muons, electrons/photons and low-energy neutrons (referring to importance biasing requested by BIASING respectively with WHAT(1) = `1.0`, `2.0` and `3.0`, or = `0.0` for all).
The number of RR actually refers to "all particles which have not been splitted" (a particle crossing a boundary between two regions of equal importance is submitted neither to RR nor to splitting, but is counted as if it was a RR).
Therefore, the counters can be used to calculate the following quantities, useful as a guide to set importances and weight windows:
`A = "N. of RR" + "N. of Sp"` = total number of particles entering the region
`B = ("<Wt> in"`$_{RR}$ `* "N. of RR") + ("<Wt> in"`$_{Sp}$ `* "N. of Sp")` = total weight of the particles entering the region
`B/A` = average weight of the particles entering the region
Note that RR and splitting arising from Weight-Window biasing (options WW–FACTOr, WW–THRESh, WW–PROFIle, pp. 244, 249, 247) or from multiplicity biasing (WHAT(2) in option BIASING) are not accounted for in the counters.

– **Final global statistics**
At the end of a successful run, after a title

```
Summary of FLUKA-xxxx run
```

the following are printed:
  – Total number of primary particles
  – Total weight of the primary particles (can be different from the previous one, especially if a user source has been used)
  – Total number of stars (hadron inelastic collisions) generated
  – Total weight of the stars generated
    Note: this statistics includes *all* hadron inelastic collisions, independently from any threshold set by option THRESHOLd. Therefore, this number of stars can be different from that obtained with SCORE or USRBIN.
  – Total number of low-energy neutron interactions
  – Total weight of the low-energy neutron interactions
  – Total CPU time used to describe all the histories
  – Average CPU time per history
  – Maximum CPU time used by a history
  – Time left before time limit (if applicable)

A more detailed statistics of different quantities follows, including all regions and separately for prompt and radioactive decay radiation. The contribution of each type of particle is given, normalised per unit weight of beam/source particles and also in percent of the total.
  – Stars (inelastic hadron interactions)
  – Secondaries created in inelastic hadron interactions
  – High-energy fissions
  – Decay products
  – Decayed particles
    For each particle the decay length at the decay position is also reported (decay length = $c\tau$, with τ = mean life)
  – Particles reaching cut-off energy
  – Secondaries created in low-energy neutron interactions

Energy balance
Each main output ends with a global energy budget. The user should always check it to make sure that the calculation is not affected by any unphysical effect or mistake. The budget entries are:
  – **Total** energy "deposited" per unit weight of beam/source particle.
    "Deposited" means actually "accounted for". This value should normally be equal to the average energy of the primary particles, except in the case of abnormal termination before the end of the first history

- Energy deposited by **hadron and muon dE/dx** (ionisation)
  This is energy deposited by continuous losses of heavy charged particles (hadron, muon and ion stopping power). Note that it depends on several user choices:
  * the thresholds set for delta ray production (energy transferred to electrons is accounted for in the next entry)
  * the transport thresholds (energy of stopping particles is counted in a separate entry below)
  * transport of light ion recoils (if not transported, their energy is deposited at the point of production and is recorded under a separate entry)
- Energy deposited by **electrons and positrons**
  For historical reasons this is labelled as "by electro-magnetic showers". It may depend on delta ray thresholds (see above)
- Energy deposited by **nuclear recoils and heavy fragments**
  This includes only ions which *have not* been transported
- Energy of particles **below threshold**
- **Residual excitation** energy
  This is excitation energy which is left after evaporation of nuclei and not emitted as prompt gamma de-excitation (e.g., radioactive decay energy)
- Energy deposited by **low-energy neutrons**
  This energy is deposited as kerma at the point of collision. It does not include the energy of hydrogen proton recoils and that of protons from the ^{14}N(n,p)^{14}C reaction. These protons are transported explicitly and their energy losses are accounted for as ionisation losses
- **Escaping** the system the system
  More generally, energy entering the blackhole, whether in the external region or in other blackhole regions defined by the user
- Energy of **discarded** particles
  (Remember that neutrinos are always discarded by default)
- Energy of particles **out of time limit**
  Can be different from zero only if the TIME–CUT option has been chosen for at least one type of particles (p. 206)
- **Missing** energy
  This entry is calculated as the difference between the total energy and the sum of all the other entries. While it is usually extremely small in pure electromagnetic problems, it can take substantial values (positive or negative) in problems involving nuclear reactions. It is usually positive, since most nuclear reactions are endothermic (very roughly, about 8 MeV are spent to produce each secondary nucleon). However, most low-energy neutron capture reactions are exothermic, and several MeV per capture are emitted as gamma rays. In problems with fissile materials such as ^{235}U, the "missing" energy can reach very large negative values.

Note that a similar, but more detailed energy balance can be obtained event by event with option EVENTDAT (p. 283). See description below, in 9.5.3.

9.2 Scratch file

The scratch file used by FLUKA is of little importance for most users, and actually is deleted automatically by the `rfluka` script at the end of a successful run. It is mentioned here only for the sake of completeness.

The file, written on output unit `LUNGEO` (`16` by default) is a simple echo of the Combinatorial Geometry input, in a different format. At input time, FLUKA stores temporarily the geometry data on this file and calculates the length of the various geometry arrays, which must include also additional information (e.g., the `DNEAR` value for each body, see Note 4 to option GLOBAL, p. 126). Then the data are retrieved and the final memory allocation takes place.

9.3 Random number seeds

A file of 97 seeds for the random number generator is written on output unit `LUNRAN` (`2` by default) at the end of each run. The file, which can be read in the next run on an input unit defined by the user with option

RANDOMIZe (p. 186, is written in hexadecimal format.

9.4 Error messages

Most error messages are written on output unit LUNERR (15 by default), interspersed with lines giving the number of random number generator calls identical to those printed on standard output (see 9.1 above). This file has generally extension .err. Each error message begins with the name of the routine in which it is originated. Some messages, however (especially if fatal) are printed on standard output. Many error messages (often somewhat cryptic) are printed only for debugging or information purposes and should be of concern to the user unless there is a large number of them: for instance when one of the hadronic event generators fails to conserve some quantity within the strict limits imposed. Example:

```
%  Eventq: charge/baryon conservation failure with Nucrin 5 4 11 10
   Eventv: ekin+am < pla,ij,igreyt  4.93747684  4.94905223 14 1
```

The following type of message is also not important, and is especially frequent in runs with photonuclear reactions activated:

```
   *** Umfnst: eexany,eexdel,eexmin,amepar,enenew,np,ikpmx,eexnew,eexmax  0.
       0.002  0.004319  1.11498839  1.09082268 2 0  0.  0.0171591096
```

Another type of informative message, indicating that a step counter has been reset because it was approaching the upper limit for an integer, is the following:

```
   *** Emfgeo: Ncoun 2000000000
```

Generally, messages issued by the geometry routines are more important. However, fatal ones are written to standard output, for instance:

```
   EXIT BEING CALLED FROM G1, NEXT REGION NOT FOUND
```

In such cases, it is recommended to run the geometry debugger (see command GEOEND on page 123) to find and correct the error.

The following one indicates a real problem if repeated more than a few times:

```
 GEOFAR, TXYZ:  1. -0.0721463599 -0.409276348 -0.909553612
    Nfrom, Nreg, X, Y, Z 1001 3 -0.108787724 -1.26878781  8.78769155

Geofar: Particle in region   3 (cell #    0) in position  1.000000000E+00
0.000000000E+00  1.000000000E+00
 is now causing trouble, requesting a step of  6.258867675E-07 cm
 to direction -2.285059979E-01 -9.412338141E-01  2.487245789E-01, error count: 0
```

[. . . skipped . . .]

```
 Particle index    3 total energy   5.189748600E-04 GeV  Nsurf    0
We succeeded in saving the particle:  current region is n.     2 (cell #    0)
```

As it can be seen, the program has some difficulty to track a particle in a certain direction, and it tries to fix the problem by "nudging" the particle by a small amount, in case the problem is due to a rounding error near a boundary. If the message appears often, it is recommended to run the geometry debugger centring around the position reported in order to find if there is an error in the geometry description.

Other geometry errors concern particles with direction cosines not properly normalised. This happens often with user routines where the user has forgotten to check that the sum of the squares be = 1.0D0 *in double precision*. For instance, the following message is generally caused by an inaccurate MAGFLD user routine:

```
MAGNEW, TXYZ: ...[sum of the squares]... U,V,V: ...[3 cosines]...
```

A similar message may be issued by the tracking routine GEOFAR:

```
GEOFAR, TXYZ: ...[sum of the squares]... U,V,V: ...[3 cosines]...
Nfrom, Nreg, X, Y, Z' ...[calling code, region number, particle position]...
```

Another geometry error message is the following:

```
**************************************
       GEOMETRY SEARCH ARRAY FULL
**************************************
```

This message indicates that insufficient memory has been allocated for the "contiguity list" (list of zones contiguous to each zone, see 8.2.6.1, 8.2.6.3). This is not an actual error, but it is suggested that the user could optimise computer time by increasing the values of the NAZ variable in the geometry region specifications.

9.5 Estimator output

Most estimator results can be printed either as unformatted files or as formatted (ASCII) text, on logical output units chosen by the user. The only exception is DETECT (p. 91), for which only the unformatted option is available, and for which the output unit number cannot be chosen (it is always 17).

If the formatted option is chosen, it is possible to write the estimator output as part of the main output (logical output unit 11). It is also possible to write the results of more than one detector on the same file. However, the task of post-processing analysis programs is easier if estimators of a different kind (e.g., USRBIN and USRBDX), or even detectors with a different structure (e.g., two USRBINs wit a different number of bins), have their outputs directed to separate files.

All the formatted estimator outputs follow the same pattern:

- The title of the run (as given in input with option TITLE, p. 207).
- Date and time
- Total number of particles followed, and their total weight. (Note that the number of particles is written in format I7, that may be insufficient for very large runs. In this case the value will be replaced by a line of asterisks)

9.5.1 DETECT output

Option DETECT (p. 91) produces only unformatted output (see DETECT description on p. 91 for instructions on how to read it). As for all other estimators (p. 277), a complete description in clear of the requested scoring is printed on the standard output. For instance:

```
Detector n.   1  "COINC     " , Ecutoff = 3.142E-07 GeV
 1024 energy bins 2.717E-03 GeV wide, from 3.700E-03 to 2.786E+00 GeV
   energy deposition in   1 regions, (n.:   3)
       in coincidence with
   energy deposition in   1 regions, (n.:   4)

Detector n.   2  "ANTICOINC " , Ecutoff = 6.614E-06 GeV
 1024 energy bins 6.704E-03 GeV wide, from 7.300E-03 to 6.872E+00 GeV
   energy deposition in   1 regions, (n.:   4)
       in anti-coincidence with
   energy deposition in   1 regions, (n.:   3)
```

9.5.2 EVENTBIN output

Option EVENTBIN (p. 111) produces either unformatted or formatted output. The formatted output is seldom used because of its size (the binning results, similar to those produced by option USRBIN (see 9.5.6

below), are printed after each primary event). As for most other estimators, a complete description in clear of the requested scoring is printed also on the standard output. For instance:

```
   Cartesian binning n.   1  "Eventscore" , generalised particle n.  208
      X coordinate: from -1.5000E+02 to  1.5000E+02 cm,    75 bins ( 4.0000E+00 cm wide)
      Y coordinate: from  1.0000E+02 to  2.0000E+02 cm,    50 bins ( 2.0000E+00 cm wide)
      Z coordinate: from -2.0000E+01 to  1.8000E+02 cm,    20 bins ( 1.0000E+01 cm wide)
      data will be printed on unit   21 (unformatted if < 0)
      accurate deposition along the tracks requested
      unnormalised data will be printed event by event
```

The header of the formatted output is practically identical to that of USRBIN, except for the words "event by event" printed after the total number of particles:

```
 *****  Title (as provided by input command TITLE)                                    *****

           DATE:  1/ 5/ 5,  TIME:  8:39:44

          Total number of particles to be followed    8000, event by event

1
   Cartesian binning n.   1  "Eventscore" , generalised particle n.  208
      X coordinate: from -1.5000E+02 to  1.5000E+02 cm,    75 bins ( 4.0000E+00 cm wide)
      Y coordinate: from  1.0000E+02 to  2.0000E+02 cm,    50 bins ( 2.0000E+00 cm wide)
      Z coordinate: from -2.0000E+01 to  1.8000E+02 cm,    20 bins ( 1.0000E+01 cm wide)
      Data follow in a matrix A(ix,iy,iz), format (1(5x,1p,10(1x,e11.4)))

      accurate deposition along the tracks requested
```

The binning matrix is then printed once for each event (8000 times in the above example), every time preceded by a line:

```
  Binning n:    1, "Eventscore",  Event #:     1, Primary(s) weight  1.0000E+00
.....................................................................................
  Binning n:    1, "Eventscore",  Event #:  8000, Primary(s) weight  1.0000E+00
```

As for most other estimators, the matrix is easily read and manipulated by a simple program, using the format reported in the header.

9.5.3 EVENTDAT output

Option EVENTDAT (p. 113) produces either unformatted or formatted output (see EVENTDAT description for instructions on how to read an unformatted output). Unlike other estimators, no information is printed on standard output.

The formatted output begins with run title and run time, followed by a short information about:

- Number of regions
- Number of generalised particle distributions requested

Then, for each primary history:

- History number
- Primary weight
- Primary energy
- Total energy balance for the current history, made of 12 contributions. Some of them correspond to those found in the final balance printed at the end of the standard output, but in this case no

normalisation to the primary weight is made. Note that some of the contributions are meaningful only in specific contexts (e.g., if low-energy neutron transport has been requested). No explanation is given about the meaning of each contribution, which must be found here below in the order they are printed:
1 = energy deposited by ionisation
2 = energy deposited by π^0, electrons, positrons and photons
3 = energy deposited by nuclear recoils and heavy fragments
4 = energy deposited by particles below threshold
5 = energy leaving the system
6 = energy carried by discarded particles
7 = residual excitation energy after evaporation
8 = energy deposited locally by low-energy neutrons (kerma)
9 = energy of particles outside the time limit
10 = energy lost in endothermic nuclear reactions above 50 MeV
11 = energy lost in endothermic low-energy neutron reactions
12 = missing energy

– Energy or stars (depending on the generalised particle scoring distribution) deposited or produced in each region during the current history
– Random number generator information to be read in order to reproduce the current sequence (skipping calls)

Example:

```
  **** Event-Data ****
 Energy deposition by protons in PbWO4
 DATE:  1/ 5/ 5,  TIME: 17:42:33
  No. of regions. 3  No. of distr. scored 1

  Event # 1
  Primary Weight  1.  Primary Energy  2. GeV
      Contributions to the energy deposition
     (GeV not normalised to the weight):
 0.519268453  0.963951886  0.183623865  0.  0.  0.0999941751  0.  0.0797502846
 0.  0.  0.  0.153411314
     Generalised scoring distribution # 208
     from first to last region:
 0.  0.109102778  1.6374917
  Seeds after event # 1
***  FADB81       0       0       0       0       0 33B49B1       0       0       0***

  Event # 2
  Primary Weight  1.  Primary Energy  2. GeV
      Contributions to the energy deposition
     (GeV not normalised to the weight):
 1.04533529  0.827161014  0.00902671926  0.  0.  0.  0.  0.0179061908  0.  0.
 0.  0.100570783
     Generalised scoring distribution # 208
     from first to last region:
 0.  0.00186400011  1.89756525
  Seeds after event # 2
*** 1034722       0       0       0       0       0 33B49B1       0       0       0***
 ................................................................................
```

9.5.4 RESNUCLEi output

Option RESNUCLEi (p. 187) produces either formatted or unformatted output. For the latter, see RESNUCLEi description for instructions on how to read it.
The formatted output begins with the same heading as the standard output (run title and run time), followed by a short information about:

1. Total number of primary particles followed
2. Total weight of the primaries
3. Number and name of the residual nuclei detector, type of reactions considered (high energy or low energy only, or all), region number
4. Detector volume in cm^3
5. Range of Z and N-Z printed
6. Tabulation format

The information reported in 3, 4 and 5 is printed also in the expanded input summary on main output (see (g), p. 277). For instance:

```
Res. nuclei n.   1  "Al-Region " , "high" energy products, region n.     3
   detector volume:  1.0000E+00 cm**3
   Max. Z:  86, Max. N-Z:  49 Min. N-Z: -4
   data will be printed on unit   21 (unformatted if < 0)

Res. nuclei n.   2  "Cu-Region " ,  all         products, region n.     4
   detector volume:  1.0000E+00 cm**3
   Max. Z:  33, Max. N-Z:  12 Min. N-Z: -4
   data will be printed on unit  -22 (unformatted if < 0)
```

On the formatted RESNUCLEi output, the above text is followed by one additional line explaining how to read the result matrix which follows:

```
 Data follow in a matrix A(z,n-z-k), k: -5 format (1(5x,1p,10(1x,e11.4)))
```

Here is an example of a simple program which can be used to display the same results in a more plain way:

```
      PROGRAM READRN
      CHARACTER*125 LINE, FILINP, FILOUT
      PARAMETER (MAXZ = 86, MINNMZ = -4, MAXNMZ = 49, K = -5)
      DIMENSION RESULT(MAXZ, MINNMZ-K:MAXNMZ-K)

      WRITE(*,*) "Filename?"
      READ(*,'(A)') FILINP
      OPEN(UNIT=1, FILE=FILINP, STATUS='OLD')
      LQ = INDEX(FILINP,' ') - 1
      FILOUT = FILINP(1:LQ)//'.rn'
      OPEN(UNIT=2, FILE=FILOUT, STATUS='UNKNOWN')

      DO 1 I = 1, 14
         READ(1,'(A)') LINE    !  skip header lines
 1    CONTINUE

      READ(1,100,END=4) RESULT
 4    CONTINUE
      WRITE(2,'(A)')    '   Z   A    Residual nuclei'
      WRITE(2,'(A,/)') '         per cm**3 per primary'
      DO 2 I = 1, MAXZ
         DO 3 J = MINNMZ-K, MAXNMZ-K
            IF(RESULT(I,J) .GT. 0.D0)
     &       WRITE(2,'(2I4,1P, G15.6)') I, J+K+2*I, RESULT(I,J)
 3       CONTINUE
 2    CONTINUE
100   FORMAT(1(5X,1P,10(1X,E11.4)))
      END
```

9.5.5 USRBDX output

Option USRBDX (p. 211) produces either formatted or unformatted output (for the latter, see USRBDX description for instructions on how to read it). As for most other estimators, a complete description in clear of the requested scoring is printed also on the standard output. For instance:

```
 Bdrx n.   1  "bxlogchb  " , generalised particle n.  202, from region n.     5 to region n.     6
    detector area:  6.3664E+01 cm**2
    this is a two ways estimator
    this is a fluence like estimator
    logar. energy binning from  1.0000E-11 to  3.0000E+00 GeV,   200 bins (ratio : 1.1413E+00)
    linear angular binning from  0.0000E+00 to  1.2566E+01 sr ,     1 bins ( 1.2566E+01 sr wide )
    data will be printed on unit  -25 (unformatted if < 0)
```

After the title and date, and one line reporting the total number of particles and their weight, the header of the formatted output is very similar to the above text:

```
*****  Test boundary crossing estimator                                                *****

          DATE: 10/25/ 4,  TIME: 10:32:59

         Total number of particles followed   10000, for a total weight of  1.0000E+04

1

   Bdrx n.   5  "bxlogchf  " , generalised particle n.  202, from region n.     5 to region n.     6
      detector area:  6.3664E+01 cm**2
      this is a two ways estimator
      this is a fluence like estimator
      logar. energy binning from  1.0000E-11 to  3.0000E+00 GeV,   200 bins (ratio : 1.1413E+00)
      linear angular binning from  0.0000E+00 to  1.2566E+01 sr ,     1 bins ( 1.2566E+01 sr wide )
      Data follow in a matrix A(ie,ia), format (1(5x,1p,10(1x,e11.4)))
```

As for most other estimators, the matrix is easily read and manipulated by a simple program, using the format reported in the header. It can also be cut and pasted into a spreadsheet.

9.5.6 USRBIN output

Option USRBIN (p. 218) produces either formatted or unformatted output (for the latter, see USRBIN description for instructions on how to read it). As for most other estimators, a complete description in clear of the requested scoring is printed also on the standard output. For instance:

```
   Cartesian binning n.   1  "Cufront   " , generalised particle n.  208
      X coordinate: from -2.1100E-01 to  5.5910E+00 cm,    58 bins ( 1.0003E-01 cm wide)
      Y coordinate: from  0.0000E+00 to  5.4010E+00 cm,    53 bins ( 1.0191E-01 cm wide)
      Z coordinate: from  0.0000E+00 to -1.0000E-03 cm,     1 bins (-1.0000E-03 cm wide)
      data will be printed on unit   21 (unformatted if < 0)
      +/- Y symmetry requested and implemented
      accurate deposition along the tracks requested
      normalised (per unit volume) data will be printed at the end of the run
```

After the title and date, and one line reporting the total number of particles and their weight, the header of the formatted output is very similar to the above text:

```
*****  Roman Pot: box with windows                                                     *****

          DATE: 12/ 8/ 3,  TIME: 15:57:27

         Total number of particles followed   30000, for a total weight of  3.0000E+04

1
```

```
Cartesian binning n.   1  "Cufront   " , generalised particle n.  208
   X coordinate: from -2.1100E-01 to  5.5910E+00 cm,    58 bins ( 1.0003E-01 cm wide)
   Y coordinate: from  0.0000E+00 to  5.4010E+00 cm,    53 bins ( 1.0191E-01 cm wide)
   Z coordinate: from  0.0000E+00 to -1.0000E-03 cm,     1 bins (-1.0000E-03 cm wide)
   Data follow in a matrix A(ix,iy,iz), format (1(5x,1p,10(1x,e11.4)))

   +/- Y symmetry requested and implemented
   accurate deposition along the tracks requested
```

As for most other estimators, the matrix is easily read and manipulated by a simple program, using the format reported in the header. It can also be cut and pasted into a spreadsheet.

9.5.7 USRCOLL output

Option USRCOLL (p. 227) produces either formatted or unformatted output (for the latter, see USRTRACK description for instructions on how to read it - the two options produce output with identical format).

As for most other estimators, a complete description in clear of the requested scoring is printed also on the standard output. For instance:

```
Coll  n.   1  "collogchf " , generalised particle n.  202, region n.     6
   detector volume:  4.0000E+01 cm**3
   Warning! Collision estimators not implemented for electrons/positrons and photons
   logar. energy binning from  1.0000E-11 to  1.0000E+00 GeV,  1000 bins (ratio : 1.0257E+00)
   data will be printed on unit   24 (unformatted if < 0)
```

After the title and date, and one line reporting the total number of particles and their weight, the header of the formatted output is very similar to the above text:

```
*****  Test collision estimator                                                  *****

          DATE:  1/ 5/ 5,  TIME: 18:32:28

         Total number of particles followed  100000, for a total weight of  1.0000E+05

1

Coll  n.   1  "collogchf  " , generalised particle n.  202, region n.     6
   detector volume:  4.0000E+01 cm**3
   logar. energy binning from  1.0000E-11 to  1.0000E+00 GeV,  1000 bins (ratio : 1.0257E+00)
   Data follow in a vector A(ie), format (1(5x,1p,10(1x,e11.4)))
```

As for most other estimators, the matrix is easily read and manipulated by a simple program, using the format reported in the header. It can also be cut and pasted into a spreadsheet.

9.5.8 USRTRACK output

Option USRTRACK (p. 231) produces either formatted or unformatted output (for the latter, see USRTRACK description for instructions on how to read it).

As for most other estimators, a complete description in clear of the requested scoring is printed also on the standard output. For instance:

```
Track n.   1  "tklogchb  " , generalised particle n.  202, region n.     6
   detector volume:  4.0000E+01 cm**3
   logar. energy binning from  1.0000E-11 to  1.0000E+00 GeV,  1000 bins (ratio : 1.0257E+00)
   data will be printed on unit  -23 (unformatted if < 0)
```

After the title and date, and one line reporting the total number of particles and their weight, the header of the formatted output is very similar to the above text:

```
 *****  Test track-length/coll. reading program for the manual                          *****

          DATE: 10/25/ 4,  TIME: 10:32:59

         Total number of particles followed   10000, for a total weight of  1.0000E+04

1

   Track n.   5  "tklogchf  " , generalised particle n.  202, region n.     6
      detector volume:  4.0000E+01 cm**3
      logar. energy binning from  1.0000E-11 to  1.0000E+00 GeV,  1000 bins (ratio : 1.0257E+00)
      Data follow in a vector A(ie), format (1(5x,1p,10(1x,e11.4)))
```

As for most other estimators, the matrix is easily read and manipulated by a simple program, using the format reported in the header. It can also be cut and pasted into a spreadsheet.

9.5.9 USRYIELD output

Option USRYIELD (p. 236) produces either formatted or unformatted output (for the latter, see USRYIELD description for instructions on how to read it).

As for most other estimators, a complete description in clear of the requested scoring is printed also on the standard output. For instance:

```
    Yield n.   1  "TotPi+(E) " , generalised particle n.   13, from region n.     3 to region n.     2
       user normalisation:  1.0000E+00, adopted cross section (if any):  1.0000E+00 mb
       logar. 1st variable binning from  1.0000E-03 to  5.0000E+01   100 bins (ratio : 1.1143E+00)
       2nd variable ranges from   0.0000E+00 to  3.1416E+00
       1st variable is: Laboratory Kinetic Energy
       2nd variable is: Laboratory Angle (radians)
       data will be printed on unit   21 (unformatted if < 0)
```

After the title and date, and one line reporting the total number of particles and their weight, the header of the formatted output is very similar to the above text:

```
 *****  Yield calculation                                                               *****

          DATE:  1/ 5/ 5,  TIME: 18:54:19

         Total number of particles followed      10, for a total weight of  1.0000E+01

1

   Yield n.   1  "TotPi+(E) " , generalised particle n.   13, from region n.     3 to region n.     2
      user normalisation:  1.0000E+00, adopted cross section (if any):  1.0000E+00 mb
      logar. 1st variable binning from  1.0000E-03 to  5.0000E+01   100 bins (ratio : 1.1143E+00)
      2nd variable ranges from   0.0000E+00 to  3.1416E+00
      1st variable is: Laboratory Kinetic Energy
      2nd variable is: Laboratory Angle (radians)
      Data follow in a vector A(ie), format (1(5x,1p,10(1x,e11.4)))
```

As for most other estimators, the matrix is easily read and manipulated by a simple program, using the format reported in the header. It can also be cut and pasted into a spreadsheet.

9.6 USERDUMP output

As a default, no formatted output is available for the USERDUMP option (p. 208). A description of the unformatted collision file is given in Chap. 11 (p. 299). However, the user can modify the MGDRAW user routine as described in 12.2.13 (p. 310), to obtain any desired output of selected events in the preferred format.

9.7 RAY output

Tracking RAY pseudoparticles (Chap. 14, p. 336) produces only an unformatted file. No formatted output is available.

9.8 User-generated output

Users can generate their own output in any user routine. However, one special routine, USROUT (p. 324), has been designed for this purpose. It is called on request by command USROCALL (p. 230), usually at the end of the run, after command START. The desired output can be printed on the standard output file (logical unit LUNOUT), or on one or more separate files. These can be opened explicitly with a normal Fortran OPEN statement or with a FLUKA OPEN command (p. 158). Otherwise, any WRITE(xx,...) statement will cause a file to be opened by default with a name fort.xx (ftn.xx on some systems), where xx is a logical unit number. In any case, it is important that the logical unit numbers be ≥ 21 (unit numbers up to 20 are internally reserved for FLUKA).

Chapter 10

Low-energy neutrons in FLUKA

10.1 Multigroup neutron transport

Transport of neutrons with energies lower than a certain energy is performed in FLUKA by a multigroup algorithm. The energy boundary below which multigroup transport takes over depends in principle on the cross-section library used: in the library which is presently distributed with the code this energy is 19.6 MeV[1]

The multi-group technique, widely used in low-energy neutron transport programs, consists in dividing the energy range of interest in a given number of intervals ("energy groups"). Elastic and inelastic reactions are simulated not as exclusive processes, but by group-to-group transfer probabilities forming the so-called *downscattering matrix*.

The scattering transfer probability between different groups is represented by a Legendre polynomial expansion truncated at the $(N+1)^{th}$ term, as shown in the equation:

$$\sigma_s(g \rightarrow g', \mu) \; = \; \sum_{i=0}^{N} \frac{2i+1}{4\pi} \mathrm{P}_i(\mu) \sigma_s^i(g \rightarrow g')$$

where $\mu \; = \; \overline{\Omega} \cdot \overline{\Omega}'$ is the scattering angle and N is the chosen Legendre order of anisotropy.

The particular implementation used in FLUKA has been derived from that of the MORSE program [44] (although the relevant part of the code has been completely rewritten). In the standard cross-section library, the energy range up to 19.6 MeV is divided into 72 energy groups of approximately equal logarithmic width (one of which is thermal). The angular probabilities for inelastic scattering are obtained by a discretisation of a P5 Legendre polynomial expansion of the actual scattering distribution which preserves its first 6 moments. The generalised Gaussian quadrature scheme to generate the discrete distribution is rather complicated: details can be found in the MORSE manual [44]. The result, in the case of a P5 expansion, is a set of 6 equations giving 3 discrete polar angles (actually angle cosines) and 3 corresponding cumulative probabilities.

In the library, the first cross-section table for an isotope (isotropic term P_0) contains the transfer probabilities from each group g to any group g': $\sum_{g \rightarrow g'} / \sum_g$, where $\sum_g$ is the sum over all the g' (including the "in-scattering" term $g' = g$). The next cross-section table provides the P_1 term for the same isotope, the next the P_2 multigroup cross-sections, etc.

10.1.1 Possible artefacts

The multigroup scheme adopted in FLUKA is reliable and much faster than any possible approach using continuous cross-sections. However, it is important to remember that there are two rare situations where the group approximation could give bad results.

One of such situations may occur when each neutron is likely to scatter only once (e.g., in a very thin foil) before being scored: an artefact then is possible, due to the discrete angular distribution. In practice the problem vanishes entirely, however, as soon as there is the possibility of two or more scatterings: it must be kept in mind, in fact, that after a collision only the *polar* angle is sampled from a discrete distribution, while the azimuthal angle is chosen randomly from a uniform distribution. In addition, the 3 discrete angles are different for each $g \rightarrow g'$ combination and for each element or isotope. Thus, any memory of the initial direction is very quickly lost after just a few collisions.

The second possible artefact is not connected with the angular but with the energy structure of the cross-sections used. The group structure is necessarily coarse with respect to the resonance structure in

[1] In FLUKA, there are two neutron energy thresholds: one for high-energy neutrons (set by option PART–THR) and one for low-energy neutrons (set by option LOW–BIAS). The high-energy neutron threshold represents in fact the energy boundary between continuous and discontinuous neutron transport

many materials. A resonance in a material present in a dilute mixture or as a small piece cannot affect much a smooth neutron flux (case of so-called "infinite dilution") but if an isotope is very pure and is present in large amounts, it can act as a "neutron sink", causing sharp dips in the neutron spectrum corresponding to each resonance. This effect, which results in a lower reaction rate $\sigma\phi$, is called *self-shielding* and is necessarily lost in the process of cross-section averaging over the width of each energy group, unless a special correction is made. Such corrected cross-section sets with different degrees of self-shielding have been included in the FLUKA library for a few important elements (Fe, Cu, Pb): but it is the responsibility of the user to select the set with the degree of self-shielding most suitable in each different case. It is worth stressing that non-self-shielded materials are perfectly adequate in most practical cases, because the presence of even small amounts of impurities is generally sufficient to smooth out the effect. On the other hand, in regions of non-resolved resonances the multigroup approach is known to give very good results anyway.

10.2 Pointwise transport

For a few isotopes only, neutron transport can be done also using continuous (pointwise) cross-sections. For ^{1}H, ^{6}Li and ^{10}B, it is applied as a user option (above 10 keV in ^{1}H, for all reactions in ^{6}Li, and only for the reaction ^{10}B(n,tγ)^{4}He in ^{10}B). For the reaction ^{14}N(n,p)^{14}C, pointwise neutron transport is always applied.

10.3 Secondary particle production

10.3.1 Gamma generation

In general, gamma generation by low-energy neutrons (*but not gamma transport*) is treated in the frame of a multigroup scheme too. A downscattering matrix provides the probability, for a neutron in a given energy group, to generate a photon in each of 22 gamma energy groups, covering the range 10 keV to 20 MeV. With the exception of a few important gamma lines, such as the 2.2 MeV transition of Deuterium and the 478 keV photon from ^{10}B(n,γ) reaction, the actual energy of the generated photon is sampled randomly in the energy interval corresponding to its gamma group. Note that the gamma generation matrix does not include only capture gammas, but also gammas produced in other inelastic reactions such as (n,n$'$).

For a few elements (Cd, Xe, Ar), for which evaluated gamma production cross-sections could not be found, a different algorithm, based on published energy level data, has been provided to generate explicitly the full cascade of monoenergetic gammas [61].

In all cases, the generated gammas are transported in the same way as all other photons in FLUKA, using continuous cross-sections and an explicit and detailed description of all their interactions with matter, allowing for the generation of electrons, positrons, and even secondary particles from photonuclear reactions.

10.3.2 Secondary neutrons

In the multigroup transport scheme, the production of secondary neutrons via (n,xn) reactions is taken into account implicitly by the so-called *non-absorption probability*, a group-dependent factor by which the weight of a neutron is multiplied after exiting a collision. If the only possible reactions are capture and scattering, the non-absorption probability is < 1, but at energies above the threshold for (n,2n) reaction it can take values larger than 1.

Fission neutrons, however, are treated separately and created explicitly using a group-dependent fission probability. They are assumed to be emitted isotropically and their energy is sampled from the fission spectrum appropriate for the relevant isotope and neutron energy. The fission neutron multiplicity is obtained separately from data extracted from European, American and Japanese databases.

10.3.3 Generation of charged particles

Recoil protons and protons from N(n,p) reaction are produced and transported explicitly, taking into account the detailed kinematics of elastic scattering, continuous energy loss with energy straggling, delta ray production, multiple and single scattering.

The same applies to light fragments (α,^{3}H) from neutron capture in ^{6}Li and ^{10}B, if pointwise transport

has been requested by the user. All other charged secondaries, including fission fragments (see 10.3.4), are not transported but their energy is deposited at the point of interaction (kerma approximation).

10.3.4 Residual nuclei

For many materials, but not for all, group-dependent information on the residual nuclei produced by low-energy neutron interactions is available in the FLUKA library. This information can be used to score residual nuclei, but it is important that the user check its availability before requesting scoring.

Fission fragments are sampled separately, using evaluated data extracted from European, American and Japanese databases.

10.4 The FLUKA neutron cross-section library

As explained in Chap. 3, an unformatted cross-section data set, or library, is needed for low-energy neutron transport. For a description of the algorithms used for tracking low-energy neutrons, see 10.1. Other useful information can be found in the Notes to options LOW–NEUT (p. 140), LOW–MAT (p. 138) and LOW–BIAS (p. 135).

The FLUKA cross-section library for low-energy neutrons, prepared originally by experts of ENEA [42] using a specialised code [117] and several ad-hoc programs written to adjust the output to the particular structure of this library, is continuously enriched and updated on the basis of the most recent evaluations (ENDF/B, JEF, JENDL etc.) The format of the library is similar to that known as ANISN [46] (or FIDO) format, but it has been modified to include kerma factor data, residual nuclei and partial exclusive cross-sections when available. The latter are not used directly by FLUKA, but can be folded over calculated spectra to get reaction rates and induced activities.

The library contains presently more than 140 different materials (natural elements or single nuclides), selected for their interest in physics, dosimetry and accelerator engineering. More materials can be made available on request, provided corresponding evaluated data files are available.

The cross-sections of some of the materials have been made available at two or three different temperatures (0, 87 and 293 K) mainly in view of simulations of calorimeters containing cryogenic scintillators. Doppler broadening at the corresponding temperatures has been taken into account.

For temperatures different from those for which a material has been prepared, an approximate approach is possible, provided the cross-sections of that material follow closely a 1/v dependence (see option MAT–PROP, p. 144). In this approximation, the thermal group velocities, absorption probabilities, gamma generation probabilities, fission probabilities and kermas are rescaled according to 1/v. No modification is made to the elastic cross-section and hence to the downscattering matrix: this can be a very bad approximation, mostly for light materials. No modification is applied for possible Doppler broadening effects on resonances for thermal and epithermal neutrons: again this can be a bad approximation. The total cross-section is rescaled according to the modified absorption and fission ones.

The averaging inside each energy group has been done according to the weighting function used by the VITAMIN-J cross-section library [154], using — from low to high energies — a Maxwellian at the relevant temperature, a 1/E spectrum in the intermediate energy range, a fission spectrum and again a 1/E spectrum. Hydrogen cross-sections, which have a particular importance in neutron slowing-down, are available also for different types of molecular binding (free, H_2O, CH_2).

At present, the FLUKA library contains only single isotopes or elements of natural isotopic composition, although the possibility exists to include in future also pre-mixed materials.

Neutron energy deposition in most materials is calculated by means of kerma factors (including contributions from low-energy fission). However, recoil protons and protons from N(n,p) reaction are produced and transported explicitly (see 10.3.3 above).

The maximum energy is 0.0196 GeV. Note that the energy groups are numbered in order of *decreasing* energy (group 1 corresponds to the highest energy).

The standard cross-section set has 72 neutron energy groups and 22 gamma groups, a structure which has been chosen for practical considerations. Gamma energy groups are used only for (n,γ) production, since transport of photons in FLUKA is continuous in energy and angle and is performed through the EMF module.

Each material is identified by an alphanumeric name (a string not longer than 8 characters, all in upper case), and by three integer identifiers. Correspondence with FLUKA materials (standard or user-defined) is based on any combination of name and zero or more identifiers. In case of ambiguity, the first material in the list fulfilling the combination is selected. (See command LOW–MAT, p. 138, for more details).

The convention generally used (but there may be exceptions) for the three identifiers is:

1. Atomic number
2. Mass number, or natural isotopic composition if negative (exceptions are possible in order to distinguish between data from different sources referring to the same nuclide)
3. Neutron temperature in degrees Kelvin

The neutron group structure of the currently available data sets is reported in Table 10.1.
The gamma group structure is reported in Table 10.2

Low-energy neutron transport is activated by option LOW–NEUT (p. 140).

Table 10.1: Neutron energy group structure of the 72-group ENEA library

Neutron group n.	Lower limit (GeV)	Upper limit (GeV)	Neutron group n.	Lower limit (GeV)	Upper limit (GeV)
1	$1.7500 \cdot 10^{-2}$	$1.9600 \cdot 10^{-2}$	**37**	$2.7324 \cdot 10^{-4}$	$3.3373 \cdot 10^{-4}$
2	$1.4918 \cdot 10^{-2}$	$1.7500 \cdot 10^{-2}$	**38**	$2.2371 \cdot 10^{-4}$	$2.7324 \cdot 10^{-4}$
3	$1.3499 \cdot 10^{-2}$	$1.4918 \cdot 10^{-2}$	**39**	$1.8316 \cdot 10^{-4}$	$2.2371 \cdot 10^{-4}$
4	$1.2214 \cdot 10^{-2}$	$1.3499 \cdot 10^{-2}$	**40**	$1.4996 \cdot 10^{-4}$	$1.8316 \cdot 10^{-4}$
5	$1.1052 \cdot 10^{-2}$	$1.2214 \cdot 10^{-2}$	**41**	$1.2277 \cdot 10^{-4}$	$1.4996 \cdot 10^{-4}$
6	$1.0000 \cdot 10^{-2}$	$1.1052 \cdot 10^{-2}$	**42**	$8.6517 \cdot 10^{-5}$	$1.2277 \cdot 10^{-4}$
7	$9.0484 \cdot 10^{-3}$	$1.0000 \cdot 10^{-2}$	**43**	$5.2475 \cdot 10^{-5}$	$8.6517 \cdot 10^{-5}$
8	$8.1873 \cdot 10^{-3}$	$9.0484 \cdot 10^{-3}$	**44**	$3.1828 \cdot 10^{-5}$	$5.2475 \cdot 10^{-5}$
9	$7.4082 \cdot 10^{-3}$	$8.1873 \cdot 10^{-3}$	**45**	$2.1852 \cdot 10^{-5}$	$3.1828 \cdot 10^{-5}$
10	$6.7032 \cdot 10^{-3}$	$7.4082 \cdot 10^{-3}$	**46**	$1.5034 \cdot 10^{-5}$	$2.1852 \cdot 10^{-5}$
11	$6.0653 \cdot 10^{-3}$	$6.7032 \cdot 10^{-3}$	**47**	$1.0332 \cdot 10^{-5}$	$1.5034 \cdot 10^{-5}$
12	$5.4881 \cdot 10^{-3}$	$6.0653 \cdot 10^{-3}$	**48**	$7.1018 \cdot 10^{-6}$	$1.0332 \cdot 10^{-5}$
13	$4.9659 \cdot 10^{-3}$	$5.4881 \cdot 10^{-3}$	**49**	$4.8809 \cdot 10^{-6}$	$7.1018 \cdot 10^{-6}$
14	$4.4933 \cdot 10^{-3}$	$4.9659 \cdot 10^{-3}$	**50**	$3.3546 \cdot 10^{-6}$	$4.8809 \cdot 10^{-6}$
15	$4.0657 \cdot 10^{-3}$	$4.4933 \cdot 10^{-3}$	**51**	$2.3054 \cdot 10^{-6}$	$3.3546 \cdot 10^{-6}$
16	$3.6788 \cdot 10^{-3}$	$4.0657 \cdot 10^{-3}$	**52**	$1.5846 \cdot 10^{-6}$	$2.3054 \cdot 10^{-6}$
17	$3.3287 \cdot 10^{-3}$	$3.6788 \cdot 10^{-3}$	**53**	$1.0446 \cdot 10^{-6}$	$1.5846 \cdot 10^{-6}$
18	$3.0119 \cdot 10^{-3}$	$3.3287 \cdot 10^{-3}$	**54**	$6.8871 \cdot 10^{-7}$	$1.0446 \cdot 10^{-6}$
19	$2.7253 \cdot 10^{-3}$	$3.0119 \cdot 10^{-3}$	**55**	$4.5400 \cdot 10^{-7}$	$6.8871 \cdot 10^{-7}$
20	$2.4660 \cdot 10^{-3}$	$2.7253 \cdot 10^{-3}$	**56**	$2.7537 \cdot 10^{-7}$	$4.5400 \cdot 10^{-7}$
21	$2.2313 \cdot 10^{-3}$	$2.4660 \cdot 10^{-3}$	**57**	$1.6702 \cdot 10^{-7}$	$2.7537 \cdot 10^{-7}$
22	$2.0190 \cdot 10^{-3}$	$2.2313 \cdot 10^{-3}$	**58**	$1.0130 \cdot 10^{-7}$	$1.6702 \cdot 10^{-7}$
23	$1.8268 \cdot 10^{-3}$	$2.0190 \cdot 10^{-3}$	**59**	$6.1442 \cdot 10^{-8}$	$1.0130 \cdot 10^{-7}$
24	$1.6530 \cdot 10^{-3}$	$1.8268 \cdot 10^{-3}$	**60**	$3.7267 \cdot 10^{-8}$	$6.1442 \cdot 10^{-8}$
25	$1.4957 \cdot 10^{-3}$	$1.6530 \cdot 10^{-3}$	**61**	$2.2603 \cdot 10^{-8}$	$3.7267 \cdot 10^{-8}$
26	$1.3534 \cdot 10^{-3}$	$1.4957 \cdot 10^{-3}$	**62**	$1.5535 \cdot 10^{-8}$	$2.2603 \cdot 10^{-8}$
27	$1.2246 \cdot 10^{-3}$	$1.3534 \cdot 10^{-3}$	**63**	$1.0677 \cdot 10^{-8}$	$1.5535 \cdot 10^{-8}$
28	$1.1080 \cdot 10^{-3}$	$1.2246 \cdot 10^{-3}$	**64**	$7.3375 \cdot 10^{-9}$	$1.0677 \cdot 10^{-8}$
29	$1.0026 \cdot 10^{-3}$	$1.1080 \cdot 10^{-3}$	**65**	$5.0435 \cdot 10^{-9}$	$7.3375 \cdot 10^{-9}$
30	$9.0718 \cdot 10^{-4}$	$1.0026 \cdot 10^{-3}$	**66**	$3.4662 \cdot 10^{-9}$	$5.0435 \cdot 10^{-9}$
31	$8.2085 \cdot 10^{-4}$	$9.0718 \cdot 10^{-4}$	**67**	$2.3824 \cdot 10^{-9}$	$3.4662 \cdot 10^{-9}$
32	$7.4274 \cdot 10^{-4}$	$8.2085 \cdot 10^{-4}$	**68**	$1.6374 \cdot 10^{-9}$	$2.3824 \cdot 10^{-9}$
33	$6.0810 \cdot 10^{-4}$	$7.4274 \cdot 10^{-4}$	**69**	$1.1254 \cdot 10^{-9}$	$1.6374 \cdot 10^{-9}$
34	$4.9787 \cdot 10^{-4}$	$6.0810 \cdot 10^{-4}$	**70**	$6.8257 \cdot 10^{-10}$	$1.1254 \cdot 10^{-9}$
35	$4.0762 \cdot 10^{-4}$	$4.9787 \cdot 10^{-4}$	**71**	$4.1400 \cdot 10^{-10}$	$6.8257 \cdot 10^{-10}$
36	$3.3373 \cdot 10^{-4}$	$4.0762 \cdot 10^{-4}$	**72**	$1.0000 \cdot 10^{-14}$	$4.1400 \cdot 10^{-10}$

Table 10.2: Gamma energy group structure of the ENEA library

Gamma group n.	Lower limit (GeV)	Upper limit (GeV)	Gamma group n.	Lower limit (GeV)	Upper limit (GeV)
1	$1.4 \cdot 10^{-2}$	$2.0 \cdot 10^{-2}$	**12**	$4.0 \cdot 10^{-3}$	$4.5 \cdot 10^{-3}$
2	$1.2 \cdot 10^{-2}$	$1.4 \cdot 10^{-2}$	**13**	$3.5 \cdot 10^{-3}$	$4.0 \cdot 10^{-3}$
3	$1.0 \cdot 10^{-2}$	$1.2 \cdot 10^{-2}$	**14**	$3.0 \cdot 10^{-3}$	$3.5 \cdot 10^{-3}$
4	$8.0 \cdot 10^{-3}$	$1.0 \cdot 10^{-2}$	**15**	$2.5 \cdot 10^{-3}$	$3.0 \cdot 10^{-3}$
5	$7.5 \cdot 10^{-3}$	$8.0 \cdot 10^{-3}$	**16**	$2.0 \cdot 10^{-3}$	$2.5 \cdot 10^{-3}$
6	$7.0 \cdot 10^{-3}$	$7.5 \cdot 10^{-3}$	**17**	$1.5 \cdot 10^{-3}$	$2.0 \cdot 10^{-3}$
7	$6.5 \cdot 10^{-3}$	$7.0 \cdot 10^{-3}$	**18**	$1.0 \cdot 10^{-3}$	$1.5 \cdot 10^{-3}$
8	$6.0 \cdot 10^{-3}$	$6.5 \cdot 10^{-3}$	**19**	$4.0 \cdot 10^{-4}$	$1.0 \cdot 10^{-3}$
9	$5.5 \cdot 10^{-3}$	$6.0 \cdot 10^{-3}$	**20**	$2.0 \cdot 10^{-4}$	$4.0 \cdot 10^{-4}$
10	$5.0 \cdot 10^{-3}$	$5.5 \cdot 10^{-3}$	**21**	$1.0 \cdot 10^{-4}$	$2.0 \cdot 10^{-4}$
11	$4.5 \cdot 10^{-3}$	$5.0 \cdot 10^{-3}$	**22**	$1.0 \cdot 10^{-5}$	$1.0 \cdot 10^{-4}$

10.5 Available materials

A list of the materials for which ENEA cross-sections are available is reported in Table 10.3.
The different columns of the Table contain, in the order:

[1]: the symbol of the nuclide (if the atomic mass number is not present, the cross-sections refer to the natural element composition)
[2]: a short description of the material
[3]: the temperature in degrees Kelvin at which the cross-sections have been processed
[4]: the evaluated data file (origin) from which the data are derived
[5]: the availability of information on production of residual nuclei
[6]: the name with which FLUKA refers to that material
[7]: the first numerical identifier of the material
[8]: the second numerical identifier of the material
[9]: the third numerical identifier of the material
[10]: the availability of information on gamma production

Table 10.3: Materials for which cross-sections are available in the FLUKA neutron library

[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]	[10]
	Material	K	Origin	Residual nuclei	Name	Identifiers			n,γ
H	H_2O bound nat. Hydrogen (1)	293	JEF–2.2	No	HYDROGEN	1	-2	293	Yes
H	CH_2 bound nat. Hydrogen (1)	293	JEF–2.2	No	HYDROGEN	1	-3	293	Yes
H	Bound natural Hydrogen (1)	293	JEF–2.2	No	HYDROGEN	1	-4	293	Yes
H	Free natural Hydrogen (1)	293	JEF–2.2	No	HYDROGEN	1	-5	293	Yes
^{1}H	H_2O bound Hydrogen 1	293	JEF–2.2	No	HYDROG-1	1	+1	293	Yes
^{1}H	CH_2 bound Hydrogen 1	293	JEF–2.2	No	HYDROG-1	1	+11	293	Yes
^{1}H	Bound Hydrogen 1	293	JEF–2.2	No	HYDROG-1	1	+21	293	Yes
^{1}H	Free Hydrogen 1	293	JEF–2.2	No	HYDROG-1	1	+31	293	Yes
H	H_2O bound nat. Hydrogen (1)	87	JEF–2.2	No	HYDROGEN	1	-2	87	Yes
H	Natural Hydrogen (1)	87	JEF–2.2	No	HYDROGEN	1	-5	87	Yes
^{2}H	Deuterium	293	JEF–1	No	DEUTERIU	1	+2	293	Yes
^{3}He	Helium 3	293	JEF–1	Yes	HELIUM-3	2	+3	293	No
He	Natural Helium (1)	293	JEF–1	Yes	HELIUM	2	-2	293	No
Li	Natural Lithium (1)	293	ENDF/B–VI	Yes	LITHIUM	3	-2	293	Yes

. . . Continues. . .

. . . Continues. . .

[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]	[10]
	Material	K	Origin	Residual nuclei	Name		Identifiers		n,γ
Li	Natural Lithium (1)	87	ENDF/B–VI	Yes	LITHIUM	3	-2	87	Yes
^{6}Li	Lithium 6	293	ENDF/B–VI	Yes	LITHIU-6	3	+6	293	Yes
^{6}Li	Lithium 6	87	ENDF/B–VI	Yes	LITHIU-6	3	+6	87	Yes
^{7}Li	Lithium 7	293	ENDF/B–VI	Yes	LITHIU-7	3	+7	293	Yes
^{7}Li	Lithium 7	87	ENDF/B–VI	Yes	LITHIU-7	3	+7	87	Yes
^{9}Be	Beryllium 9	293	ENDF/B–VI	Yes	BERYLLIU	4	+9	293	Yes
B	Natural Boron (1)	293	JEF–2.2	Yes	BORON	5	-2	293	Yes
^{10}B	Boron 10	293	JEF–2.2	Yes	BORON-10	5	+10	293	Yes
^{11}B	Boron 11	293	ENDF/B–VI	Yes	BORON-11	5	+11	293	Yes
C	Natural Carbon	293	ENDF/B6R8	Yes	CARBON	6	-2	293	Yes
C	Natural Carbon	87	ENDF/B6R8	Yes	CARBON	6	-2	87	Yes
N	Natural Nitrogen (1),(16)	293	ENDF/B6R8	Yes	NITROGEN	7	-2	293	Yes
N	Natural Nitrogen (1),(16)	87	ENDF/B6R8	Yes	NITROGEN	7	-2	87	Yes
^{16}O	Oxygen 16	293	ENDF/B6R8	Yes	OXYGEN	8	+16	293	Yes
^{16}O	Oxygen 16	87	ENDF/B6R8	Yes	OXYGEN	8	+16	87	Yes
^{19}F	Fluorine 19	293	ENDF/B–VI	Yes	FLUORINE	9	-2	293	Yes
^{23}Na	Sodium 23	293	JEF–2.2	Yes	SODIUM	11	-2	293	Yes
Mg	Natural Magnesium (2)	293	JENDL–3.2	Yes	MAGNESIU	12	-2	293	Yes
Mg	Natural Magnesium (2)	87	JENDL–3.2	Yes	available on request				Yes
^{27}Al	Aluminium 27	293	ENDF/B6R8	Yes	ALUMINUM	13	-2	293	Yes
^{27}Al	Aluminium 27	87	ENDF/B6R8	Yes	ALUMINUM	13	-2	87	Yes
^{27}Al	Aluminium 27	4	ENDF/B6R8	Yes	ALUMINUM	13	-2	4	Yes
Si	Natural Silicon (1)	293	JENDL–3.3	Yes	SILICON	14	-2	293	Yes
Si	Natural Silicon (1)	87	JENDL–3.3	Yes	available on request				Yes
^{31}P	Phosphorus 31	293	ENDF/B–VI	Yes	PHOSPHO	15	-2	293	Yes
S	Natural Sulphur (3)	293	JENDL–3.2	Yes	SULFUR	16	-2	293	Yes
Cl	Natural Chlorine (4)	293	ENDF/B–VI	No	CHLORINE	17	-2	293	Yes
Ar	Natural Argon	293	ENDL	Yes	ARGON	18	-2	293	Yes
Ar	Natural Argon	87	ENDL	Yes	ARGON	18	-2	87	Yes
K	Natural Potassium (5)	293	ENDF/B–VI	Yes	POTASSIU	19	-2	293	Yes
K	Natural Potassium (5)	87	ENDF/B–VI	Yes	available on request				Yes
Ca	Natural Calcium (6)	293	JENDL–3.3	Yes	CALCIUM	20	-2	293	Yes
Ti	Natural Titanium (7)	293	JEF–2.2	Yes	TITANIUM	22	-2	293	Yes
V	Vanadium	293	ENDF/B–VI	Yes	VANADIUM	23	-2	293	Yes
V	Vanadium	87	ENDF/B–VI	Yes	available on request				Yes
Cr	Natural Chromium (1)	293	ENDF/B–VI	Yes	CHROMIUM	24	-2	293	Yes
Cr	Natural Chromium (1)	87	ENDF/B–VI	Yes	CHROMIUM	24	-2	87	Yes
^{55}Mn	Manganese 55	293	ENDF/B–VI	Yes	MANGANES	25	55	293	Yes
^{55}Mn	Manganese 55	87	ENDF/B–VI	Yes	available on request				Yes
Fe	Natural Iron (1)	293	ENDF/B–VI	Yes	IRON	26	-2	293	Yes
Fe	Nat. Iron S.S. (1)(8)	293	ENDF/B–VI	Yes	IRON	26	-4	293	Yes
Fe	Nat. Iron Shiel. (1)(9)	293	ENDF/B–VI	Yes	IRON	26	-8	293	Yes
Fe	Natural Iron (1)	87	ENDF/B–VI	Yes	IRON	26	-2	87	Yes
Fe	Nat. Iron S.S. (1)(8)	87	ENDF/B–VI	Yes	IRON	26	-4	87	Yes
Fe	Nat. Iron Shiel. (1)(9)	87	ENDF/B–VI	Yes	IRON	26	-9	87	Yes
^{59}Co	Cobalt 59	293	ENDF/B–VI	Yes	COBALT	27	59	293	Yes
^{59}Co	Cobalt 59	87	ENDF/B–VI	Yes	available on request				Yes
Ni	Natural Nickel (1)	293	ENDF/B–VI	Yes	NICKEL	28	-2	293	Yes
Ni	Natural Nickel (1)	87	ENDF/B–VI	Yes	NICKEL	28	-2	87	Yes
Cu	Natural Copper (1)	293	ENDF/B–VI	Yes	COPPER	29	-2	293	Yes
Cu	Nat. Copper S.S. (1)(8)	293	ENDF/B–VI	Yes	COPPER	29	-4	293	Yes
Cu	Nat. Copper S.N. (1)(10)	293	ENDF/B–VI	Yes	COPPER	29	-5	293	Yes
Cu	Natural Copper (1)	87	ENDF/B–VI	Yes	COPPER	29	-2	87	Yes
Cu	Nat. Copper S.S. (1)(8)	87	ENDF/B–VI	Yes	COPPER	29	-4	87	Yes
Cu	Nat. Copper S.N. (1)(10)	87	ENDF/B–VI	Yes	COPPER	29	-5	87	Yes

. . . Continues. . .

. . . Continues. . .

[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]	[10]
	Material	K	Origin	Residual nuclei	Name	Identifiers			n,γ
Zn	Natural Zinc	293	ENEA	No	ZINC	30	-2	293	Yes
Zn	Natural Zinc	87	ENEA	No	ZINC	30	-2	87	Yes
Ga	Natural Gallium	293	ENDF/B–VI	No	GALLIUM	31	-2	293	Yes
Ga	Natural Gallium	87	ENDF/B–VI	No	available on request				Yes
Ge	Natural Germanium (1)	293	JEF–1	No	GERMANIU	32	-2	293	No
^{75}As	Arsenic 75	293	JEF–1	Yes	ARSENIC	33	75	293	No
Br	Natural Bromine (1)	293	JENDL–3.2	Yes	BROMINE	35	-2	293	No
Kr	Natural Krypton (1)	293	JEF–2.2	Yes	KRYPTON	36	-2	293	No
Kr	Natural Krypton (1)	120	JEF–2.2	Yes	KRYPTON	36	-2	120	No
^{90}Sr	Strontium 90	293	JEF–2.2	Yes	90-SR	38	90	293	No
Zr	Natural Zirconium (1)	293	BROND	Yes	ZIRCONIU	40	-2	293	Yes
^{93}Nb	Niobium 93 (11)	293	ENDF/B6R8	Yes	NIOBIUM	41	93	293	Yes
^{93}Nb	Niobium 93 (11)	87	ENDF/B6R8	Yes	available on request				Yes
Mo	Natural Molybdenum (1)	293	EFF–2.4	Yes	MOLYBDEN	42	-2	293	Yes
Mo	Natural Molybdenum (1)	87	EFF–2.4	Yes	available on request				Yes
^{99}Tc	Technetium 99	293	JEF–2.2	Yes	99-TC	43	99	293	No
Ag	Natural Silver (1)(11)	293	JENDL–3.2	Yes	SILVER	47	-2	293	Yes
Cd	Natural Cadmium	293	JENDL–3	No	CADMIUM	48	-2	293	(Yes) (12)
In	Natural Indium (1)	293	JEF–1	Yes	INDIUM	49	-2	293	No
Sn	Natural Tin (1)	293	JENDL–3	Yes	TIN	50	-2	293	No
Sn	Natural Tin (1)	87	JENDL–3	Yes	TIN	50	-2	87	No
Sb	Natural Antimony (1)	293	ENDF/B–VI	Yes	ANTIMONY	51	-2	293	No
Sb	Natural Antimony (1)	87	ENDF/B–VI	Yes	available on request				No
^{127}I	Iodine 127	293	JEF–2.2	Yes	IODINE	53	127	293	No
^{129}I	Iodine 129	293	JEF–2.2	Yes	available on request				No
Xe	Natural Xenon (1)	293	JEF–2.2	Yes	XENON	54	-2	293	(No) (13)
^{124}Xe	Xenon 124	293	JEF–2.2	Yes	124-XE	54	124	293	(No) (13)
^{126}Xe	Xenon 126	293	JEF–2.2	Yes	126-XE	54	126	293	(No) (13)
^{128}Xe	Xenon 128	293	JEF–2.2	Yes	128-XE	54	128	293	(No) (13)
^{129}Xe	Xenon 129	293	JEF–2.2	Yes	129-XE	54	129	293	(No) (13)
^{130}Xe	Xenon 130	293	JEF–2.2	Yes	130-XE	54	130	293	(No) (13)
^{131}Xe	Xenon 131	293	JEF–2.2	Yes	131-XE	54	131	293	(No) (13)
^{132}Xe	Xenon 132	293	JEF–2.2	Yes	132-XE	54	132	293	(No) (13)
^{134}Xe	Xenon 134	293	JEF–2.2	Yes	134-XE	54	134	293	(No) (13)
^{135}Xe	Xenon 135	293	JEF–2.2	Yes	135-XE	54	135	293	(No) (13)
^{136}Xe	Xenon 136	293	JEF–2.2	Yes	136-XE	54	136	293	(No) (13)
^{133}Cs	Cesium 133	293	JEF–2.2	Yes	CESIUM	55	133	293	No
^{135}Cs	Cesium 135	293	JEF–2.2	Yes	available on request				No
^{137}Cs	Cesium 137	293	JEF–2.2	Yes	available on request				No
Ba	Natural Barium (1)	293	JEF–1	Yes	BARIUM	56	-2	293	No
Ce	Natural Cerium (1)	293	JEF–2.2	Yes	CERIUM	58	-2	293	No
Sm	Natural Samarium (1)	293	ENDF/B6R8	Yes	SAMARIUM	62	-2	293	No
Gd	Natural Gadolinium (1)	293	ENDF/B–VI	Yes	GADOLINI	64	-2	293	No
Gd	Natural Gadolinium (1)	87	ENDF/B–VI	Yes	available on request				No
^{181}Ta	Tantalum 181 (11)(14)	293	ENDF/B6R8	Yes	TANTALUM	73	181	293	Yes
^{181}Ta	Tantalum 181 (11)(14)	87	ENDF/B6R8	Yes	available on request				Yes
W	Natural Tungsten (1)(11)	293	ENDF/B6R8	Yes	TUNGSTEN	74	-2	293	Yes
W	Natural Tungsten (1)(11)	87	ENDF/B6R8	Yes	TUNGSTEN	74	-2	87	Yes
^{186}W	Tungsten 186	293	ENDF/B6R8	Yes	186-W	74	186	293	Yes
Re	Natural Rhenium (1)	293	ENDF/B–VI	Yes	RHENIUM	75	-2	293	No
Re	Natural Rhenium (1)	87	ENDF/B–VI	Yes	available on request				No
^{197}Au	Gold 197 (11)	293	ENDF/B–VI	Yes	GOLD	79	197	293	Yes
Hg	Natural Mercury (1)(15)	293	JENDL–3.3	Yes	MERCURY	80	-2	293	No
Pb	Natural Lead (1)	293	ENDF/B6R6	Yes	LEAD	82	-2	293	Yes

. . . Continues. . .

. . . Continues. . .

[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]	[10]
	Material	K	Origin	Residual nuclei	Name	Identifiers			n,γ
Pb	Natural Lead S.S. (1)(8)	293	ENDF/B6R6	Yes	LEAD	82	-4	293	Yes
^{208}Pb	Lead 208	293	ENDF/B6R6	Yes	208-PB	82	208	293	Yes
^{208}Pb	Lead 208 S.S. (8)	293	ENDF/B6R6	Yes	208-PB	82	1208	293	Yes
Pb	Natural Lead (1)	87	ENDF/B6R6	Yes	LEAD	82	-2	87	Yes
Pb	Natural Lead S.S. (1)(8)	87	ENDF/B6R6	Yes	LEAD	82	-4	87	Yes
^{209}Bi	Bismuth 209 (14)	293	JEF–2.2	Yes	BISMUTH	83	209	293	Yes
^{209}Bi	Bismuth 209 (14)	87	JEF–2.2	Yes	available on request				Yes
^{230}Th	Thorium 230	293	ENDF/B–VI	Yes	230-TH	90	230	293	No
^{232}Th	Thorium 232	293	ENDF/B–VI	Yes	232-TH	90	232	293	Yes
^{233}U	Uranium 233	293	ENDF/B–VI	Yes	233-U	92	233	293	Yes
^{234}U	Uranium 234	293	ENDF/B–VI	Yes	234-U	92	234	293	Yes
^{234}U	Uranium 234	87	ENDF/B–VI	Yes	available on request				Yes
^{235}U	Uranium 235	293	ENDF/B–VI	Yes	235-U	92	235	293	Yes
^{235}U	Uranium 235	87	ENDF/B–VI	Yes	235-U	92	235	87	Yes
^{238}U	Uranium 238	293	ENDF/B–VI	Yes	238-U	92	238	293	Yes
^{238}U	Uranium 238	87	ENDF/B–VI	Yes	238-U	92	238	87	Yes
^{237}Np	Neptunium 237	293	ENDF/B–VI	Yes	237-NP	93	237	293	Yes
^{239}Pu	Plutonium 239	293	ENDF/B–VI	Yes	239-PU	94	239	293	Yes
^{239}Pu	Plutonium 239	87	ENDF/B–VI	Yes	available on request				Yes
^{241}Am	Americium 241	293	ENDF/B–VI	Yes	241-AM	95	241	293	Yes
^{243}Am	Americium 243	293	ENDF/B–VI	Yes	243-AM	95	243	293	Yes

(1) Material of natural isotopic composition obtained by collapsing together single isotope data obtained from the quoted "origin".

(2) The information on residual nuclei for natural Magnesium has been obtained from the individual Mg isotope data from JENDL–3.2. However these data do not contain gamma production, so the residual nuclei data from JENDL–3.2 individual isotopes have been used together with the JENDL–3.2 natural Magnesium cross-sections.

(3) The information on residual nuclei for natural Sulphur has been obtained from the individual S isotope data from JENDL–3.2. However these data do not contain gamma production, so the residual nuclei data from JENDL–3.2 individual isotopes have been used together with the JENDL–3.2 natural Sulphur cross-sections.

(4) The information on residual nuclei for natural Chlorine has been obtained from the individual Cl isotope data from JENDL–3.3. However these data do not contain gamma production, so the residual nuclei data from JENDL–3.3 individual isotopes have been used together with the ENDF/B–VI natural Chlorine cross-sections.

(5) The information on residual nuclei for natural Potassium has been obtained from the individual K isotope data from JENDL–3.2. However these data do not contain gamma production, so the residual nuclei data from JENDL–3.2 individual isotopes have been used together with the ENDF/B–VI natural Potassium cross-sections.

(6) The Calcium data have been processed for each isotope out of JENDL–3.3 since it is the only compilation containing data for individual calcium isotopes with gamma production. They have problems in the kerma factors, for ^{42}Ca, ^{46}Ca, ^{48}Ca, which could result from overall inconsistencies in the data. These isotopes are however a tiny fraction of the total.

(7) The information on residual nuclei for natural Titanium has been obtained from the individual Ti isotope data from JENDL–3.2. However these data do not contain gamma production, so the residual nuclei data from JENDL–3.2 individual isotopes have been used together with the JEF–2.2 natural Titanium cross-sections.

(8) Self-shielded.

(9) Partially self-shielded, appropriate for typical cast iron used for shielding.

(10) Partially self-shielded, appropriate for a natural composition without a dominant isotope.

(11) A couple of kerma values are incorrect.

(12) The cross-sections for gamma generation in Cd in the JENDL–3.2 evaluated file are considered unreliable. For this element, it is recommended to activate the explicit gamma generation routine of FLUKA (option LOW–NEUT, with WHAT(6) = 1.0 or 11.0).

(13) cross-sections for gamma generation in Xe are missing from available evaluated data files. However, for the single isotopes it is possible to activate an explicit gamma generation routine of FLUKA (capture gamma only!!). Use option LOW–NEUT, with WHAT(6) = 1.0 or 11.0). Gamma generation in Xe of natural composition can then be obtained by defining a COMPOUND of the individual isotopes, each with its own natural abundance.

(14) The (capture) gamma multiplicity is strikingly different among different evaluations

(15) The Mercury data have been processed for each isotope out of JENDL–3.3 since it is the only compilation containing data for individual Mercury isotopes up to now. There are problems in the kerma factors, which could result from overall inconsistencies in the data. Use with care!!!

Chapter 11

Collision tape

11.1 What is a collision tape and what is its purpose

A "collision tape" is a file where quantities describing selected events are recorded in the course of a FLUKA run.

This file is the standard output of the MGDRAW user routine, that can be customised by the user to get different and/or more complete output (see description of user routine MGDRAW in 12.2.13)

Note that "event" would be a more appropriate word than "collision", and "file" better than "tape". For historical reasons, however, the expression "collision tape" is used in Monte Carlo jargon rather than "event file". It is true that most interesting events are generally collision events (but also boundary crossings, decays, etc.), and that the large size of the file may require the use of a magnetic tape (or at least, that was often the case in the past). Recently, the expression "phase space file" has also been used.

There are several reasons for which the user might decide to write a collision tape. Some examples are:

1) to perform a non-standard analysis or scoring. In general, this is not recommended because the available FLUKA scoring facilities are reliable, efficient and well tested. However, there may be special cases where a user-written scoring is necessary.
2) to save details of transport for a new independent analysis. In this case, however, the user must make sure that no phase-space region of interest be undersampled because of biasing options in the corresponding run. As a general rule, writing of a collision file is not recommended in non-analogue (biased) calculations.
3) to connect FLUKA to other radiation transport codes (now less likely than in the past, since FLUKA covers most energy ranges and transports most particles which can be of interest).
4) to split the transport problem into two or more sequential phases. A technique used in deep penetration calculations, which can be considered as an extension of splitting, consists in recording all particles crossing a given boundary (with their energy, weight, coordinates and direction cosines at the point of crossing), and to sample repeatedly source particles from that set in a successive run [50]. A special subroutine SOURCE (p. 314) must be written for this purpose.
5) to perform some manipulation in an intermediate phase of Monte Carlo transport. An example is saving photon histories to be processed by some program which prepares a photomuon source for a successive run (a special case of Note 3)) where connection is of FLUKA to itself). Another example is a variation of Note 4), in which a user program interpolates some smooth analytical distribution through the recorded quantities, from which source particles are sampled in the next FLUKA run.
6) to trace suspected errors in transport
7) to connect to a graphical display program

FLUKA allows to write a complete dump of each source particle, of each trajectory and of each energy deposition event, possibly under event-driven conditions specified by the user.

11.2 How to write a collision tape

To obtain a collision tape, option USERDUMP must be input with WHAT(1) $\geq$ 100.0.

The user can choose to dump all data concerning

1) particle trajectories,
2) data concerning continuous energy deposition

3) data concerning local (point) energy deposition
4) any combination of the three above

Data are written on the collision tape in single precision and unformatted, but it is also possible for the user to modify the MGDRAW subroutine and to obtain a more customised output file (see 12.2.13).

The variables written by the default version of MGDRAW, and their number, differ in the three cases. The sign of the first (integer) variable dumped at an event indicates how to interpret the following ones:

Case **1** (First variable $>$ 0): continuous energy deposition
Case **2** (First variable $=$ 0): point energy deposition
Case **2** (First variable $<$ 0): source particles

In Case **1**, the following variables are written:

First record:
NTRACK, MTRACK, JTRACK, ETRACK, WTRACK (three integers and two real variables)

Next record:
(XTRACK(I), YTRACK(I), ZTRACK(I), I = 0, NTRACK),
(DTRACK(J), J = 1, MTRACK), CTRACK
(that is, NTRACK + MTRACK + 1 real variables)

where:

NTRACK = number of trajectory segments
MTRACK = number of energy deposition events along the trajectory
JTRACK = particle type (see 5.1)
ETRACK = total energy of the particle (rest + kinetic)
WTRACK = particle weight
XTRACK(I), YTRACK(I), ZTRACK(I) = coordinates defining the upstream end of the (I+1)th segment; for I = NTRACK, the end of the trajectory
DTRACK(J) = energy deposition in the Jth deposition event along the trajectory
CTRACK = total curved path

In Case **2**, the following variables are written:

First record:
0, ICODE, JTRACK, ETRACK, WTRACK (three integers and two real variables)

Next record:
XSCO, YSCO, ZSCO, RULL (4 real variables)

where:

JTRACK, ETRACK, WTRACK have the meaning explained above,
XSCO, YSCO, ZSCO = coordinates of the energy deposition point
RULL = amount of energy deposited
ICODE indicates the type of point event giving rise to energy deposition, as explained below:

ICODE = 1x: call from subroutine KASKAD (hadronic part of FLUKA);

= 10: elastic interaction recoil
= 11: inelastic interaction recoil
= 12: stopping particle
= 14: escape

ICODE = 2x: call from subroutine EMFSCO (electromagnetic part of FLUKA);
= 20: local energy deposition (i.e. photoelectric)
= 21: below user-defined transport cut-off (but larger than EMF production cut-off)
= 22: below both user-defined transport cut-off and EMF production cut-off
= 23: escape

ICODE = 3x: call from subroutine KASNEU (low-energy neutron part of FLUKA)
= 30: target recoil
= 31: neutron below threshold
= 32: escape

ICODE = 4x: call from subroutine KASHEA (heavy ion part of FLUKA)
= 40: escape

ICODE = 5x: call from subroutine KASOPH (optical photon part of FLUKA)
= 50: optical photon absorption
= 51: escape

In Case **3**, the following variables are written:

First record:
-NCASE, NPFLKA, NSTMAX, TKESUM, WEIPRI, (three integers and two real variables)

Next record:
(ILOFLK(I), ETOT(I), WTFLK(I), XFLK(I), YFLK(I), ZFLK(I), TXFLK(I), TYFLK(I), TZFLK(I), I = 1, NPFLKA) (NPFLKA times (that is, one integer + 8 real variables))

where:

NCASE = number of primaries treated so far (including the current one)
NPFLKA = number of particles in the stack
NSTMAX = maximum number of particles in stack so far
TKESUM = total kinetic energy of the primaries of a user written SOURCE
WEIPRI = total weight of the primaries handled so far
ILOFLK(I) = type of the Ith stack particle (see 5.1)
ETOT(I) = total energy of Ith stack particle
XFLK(I), YFLK(I), ZFLK(I) = source coordinates for the Ith stack particle
TXFLK(I), TYFLK(I), TZFLK(I) = direction cosines of the Ith stack particle

Chapter 12

User routines

Unlike some other Monte Carlo particle transport codes, FLUKA gets its input mainly from a simple file. It offers a rich choice of options for scoring most quantities of possible interest and for applying different variance reduction techniques, without requiring the users to write a single line of code. However, although normally there is no need for any "user code", there are special cases where this is unavoidable, either because of the complexity of the problem, or because the desired information is too unusual or too problem-specific to be offered as a standard option.

And on the other hand, even when this is not strictly necessary, experienced programmers may like to create customised input/output interfaces. A number of user routines (available on LINUX and UNIX platforms in directory `usermvax`) allow to define non-standard input and output, and in some cases even to modify to a limited extent the normal particle transport. Most of them are already present in the FLUKA library as dummy or template routines, and require a special command in the standard input file to be activated. Users can modify any one of these routines, and even insert into them further calls to their own private ones, or to external packages (at their own risk!). This increased flexibility must be balanced against the advantage of using as far as possible the FLUKA standard facilities, which are known to be reliable and well tested.

12.1 How to write, compile and link a user routine

To implement their own code, users must perform the following steps:

1) make a modified copy of one or more of these routines. It is recommended that each modified routine should always print an informative message when called for the first time, to confirm that it has been successfully activated, for future documentation, and to avoid misinterpretations of the standard output. It is important to remember that when calling modified user routines, the units, titles etc. reported in the normal FLUKA output become often meaningless.
   A typical way to do this is:

```
      ..............
      LOGICAL LFIRST
      SAVE LFIRST
      DATA LFIRST /.TRUE./
* return message from first call
      IF (LFIRST) THEN
         WRITE(LUNOUT,*) 'Version xxx of Routine yyy called'
         LFIRST = .FALSE.
      ENDIF
      ..............
```

   IMPORTANT: The user should not modify the value of any argument in a routine calling list, except when marked as "returned" in the description of the routine here below.
   Similarly, no variable contained in `COMMON` blocks should be overwritten unless explicitly indicated.
2) compile the modified routines (with the `fff` script on LINUX/UNIX):
   `$FLUPRO/flutil/fff yyy.f` (produces a new file `yyy.o`)
3) link them (with the `lfluka` script on LINUX/UNIX) to the FLUKA library and any additional library of interest (for instance CERNLIB):
   `$FLUPRO/flutil/lfluka -o myfluka -m fluka yyy.o`
   This will produce a new executable (indicated here as `myfluka`).
   To run the new executable, launch the usual `rfluka` script with the option `-e myfluka`.

12.1.1 INCLUDE files

It is recommended that at least the following lines be present at the beginning of each routine:

```
INCLUDE '(DBLPRC)'
INCLUDE '(DIMPAR)'
INCLUDE '(IOUNIT)'
```

Each INCLUDE file contains a COMMON block, plus related constants. Additional INCLUDEs may be useful, in particular BEAMCM, CASLIM, EMFSTK, SOURCM, EVTFLG, FHEAVY, GENSTK, LTCLCM, FLKMAT, RESNUC, SCOHLP, SOUEVT, FLKSTK, SUMCOU, TRACKR, USRBIN, USRBDX, USRTRC, USRYLD. Note the parentheses which are an integral part of the FLUKA INCLUDE file names.

Files flukaadd.add and emfadd.add, or directory $FLUPRO/flukapro, contain a full documentation about the meaning of the variables of these INCLUDE files.

Here is a summary of their content:

DBLPRC : included in *all* routines of FLUKA, contains the declaration
```
IMPLICIT DOUBLE PRECISION (A-H,O-Z)
```
and sets many mathematical and physical constants. Users are strongly encouraged to adhere to "FLUKA style" by using systematically double precision (except for very good reasons such as calling external single precision scoring packages), and to use constants defined in this file for maximum accuracy.

DIMPAR : dimensions of the most important arrays

IOUNIT : logical input and output unit numbers

BEAMCM : properties of primary particles as defined by options BEAM and BEAMPOS

CASLIM : number of primary particles followed (needed for normalisation)

EMFSTK : particle stack for electrons and photons

SOURCM : user variables and information for a user-written source

EVTFLG : event flags

FHEAVY : stack of heavy secondaries created in nuclear evaporation

GENSTK : properties of each secondary created in a hadronic event

LTCLCM : LaTtice CeLl CoMmon (needed when writing symmetry transformations for Lattice Geometry)

FLKMAT : material properties

FLKSTK : main FLUKA particle stack

RESNUC : properties of the current residual nucleus

SCOHLP : SCOring HeLP (information on current estimator type). It contains a flag ISCRNG, indicating the quantity being scored by the current estimator, and one JSCRNG corresponding to the binning/detector number. Binnings and detectors are sequentially numbered according to their order of appearance in standard input. Note that several detectors can have the same JSCRNG number (for instance Binning N. 3 and Track-length estimator N. 3). They can be distinguished based on the value of ISCRNG. However, note that the same value of ISCRNG may have different meanings in functions FLUSCW and COMSCW

SOUEVT : SOUrce EVenT (useful when source particles are obtained from an external event generator)

SUMCOU : total numbers and total weights relative to many physical and Monte Carlo events (needed for normalisation, energy balance etc.)

TRACKR : TRACK Recording (properties of the currently transported particle and its path)

USRBIN, USRBDX, USRSNC, USRTRC, USRYLD: parameters of the requested estimators

12.2 Description of available user routines

12.2.1 ABSCFF: user defined ABSorption CoeFFicient

Argument list (all variables are input only)	
WVLNGT :	photon wavelength (in cm)
OMGPHO :	angular frequency ($\omega = 2\pi\nu$) of the photon (in s^{-1})
MMAT :	material index

Function ABSCFF returns a user-defined absorption coefficient for optical photons. It is activated by setting WHAT(2) < -99 in command OPT–PROP, with SDUM = blank. See p. 164 and Chap. 13 for more information.

12.2.2 COMSCW: weighting deposited energy or stars

Argument list (all variables are input only)	
IJ :	particle type (1 = proton, 8 = neutron, etc.: see code in 5.1)
XA,YA,ZA :	current particle position
MREG :	current geometry region
RULL :	amount to be deposited (unweighted)
LLO :	particle generation
ICALL :	internal code calling flag (not for general use)

This function is activated by option USERWEIG, (p. 209) with WHAT(6) > 0.0. Energy and star densities obtained via SCORE (p. 195) and USRBIN(p. 218), energy and stars obtained via EVENTBIN (p. 111) and production of residual nuclei obtained via RESNUCLEi (p. 187) are multiplied by the value returned by this function. The user can implement any desired logic to differentiate the returned value according to any information contained in the argument list (particle type, position, region, amount deposited, particle generation), or information available in COMMON SCOHLP (binning number, type of scored quantity). The scored quantity is given by the flag ISCRNG (in SCOHLP):

ISCRNG = 1 ⟶ Energy density binning
ISCRNG = 2 ⟶ Star density binning
ISCRNG = 3 ⟶ Residual nuclei scoring

The binning/detector number is given by JSCRNG (in SCOHLP) and is printed in output between the estimator type and the detector name:

```
Res. nuclei n. 3  "any-name" , "high" energy products, region n. 4
R-Phi-Z  binning n. 5  "other-name" , generalised particle n.    1
```

Note that an detector of residual nuclei can have the same JSCRNG number as a binning (use the value of ISCRNG to discriminate). Further information can be obtained including COMMON TRACKR (for instance particle's total energy, direction cosines, age). TRACKR contains also special user variables (both integer and in double precision) which can be used to save information about particles which have undergone some particular event. If data concerning the current material are needed, it can be accessed as MEDIUM(MREG) if file (FLKMAT) is included. Indeed, a common simple application of COMSCW is to score dose according to the local density (especially useful to get the correct average dose in bins straddling a boundary between two different media):

```
        ..................
        INCLUDE '(FLKMAT)'
        INCLUDE '(SCOHLP)'
        ..................
* ========== In order to compute doses ========= *
```

```
*        Medium(n) is the material number of region n
*        Rho(m) is the density of material m (in g/cm3)
*        Iscrng = 1 means we are depositing energy (not stars)
         IF ( ISCRNG .EQ. 1 ) THEN
*           to get dose in Gy (elcmks is the electron charge in C)
            COMSCW = ELCMKS * 1.D12 / RHO (MEDIUM(MREG))
         ELSE
*           oneone is defined as 1.D0 in include DBLPRC
            COMSCW = ONEONE
         ENDIF
         ...................
```

Note: setting the variable `LSCZER = .TRUE.` before `RETURN` (`LSCZER` is in `COMMON SCOHLP`), will cause zero scoring whatever the value returned by `COMSCW`. This is more efficient than returning a zero value.

12.2.3 DFFCFF: user defined DiFFusion CoeFFicient

Argument list (all variables are input only)	
WVLNGT :	photon wavelength (in cm)
OMGPHO :	angular frequency ($\omega = 2\pi\nu$) of the photon (in s^{-1})
MMAT :	material index

Function DFFCFF returns a user-defined diffusion coefficient for optical photons. It is activated by setting WHAT(3) < -99 in command OPT–PROP, with SDUM = blank. See Sec. 7.51 and Chap. 13 for more information.

12.2.4 ENDSCP: ENergy density DiStributed — Change of Positions

Argument list	
IJ :	particle type (input only)
NTRUCK :	number of step points (input only)
XTRUCK, YTRUCK, ZTRUCK :	particle step points, can be modified by user
MREG :	region number (input only)
LLO :	particle generation (input only)
ICALL :	internal code calling flag (not for general use)

Subroutine ENDSCP allows to shift by a user-defined distance the energy which is being deposited along a step or several step binning portions, by providing new segment endpoints. A typical application is to simulate an instrument drift.

12.2.5 FLDSCP: FLuence DiStributed — Change of Positions

Argument list	
IJ :	particle type (input only)
PLA :	particle momentum (if > 0), or kinetic energy (if < 0) (input only)
TXX, TYY, TZZ :	particle direction cosines, can be modified by user
NTRUCK :	number of step points (input only)
XTRUCK, YTRUCK, ZTRUCK :	particle step points, can be modified by user
NRGFLK :	new region number (input only)
IOLREG :	old region number (input only)
LLO :	particle generation (input only)
ICALL :	internal code calling flag (not for general use)

Subroutine FLDSCP allows to shift by a user-defined distance the track whose length is being scored as fluence along a step or several step binning portions, by providing new segment endpoints. A typical application is to simulate an instrument drift.

12.2.6 FLUSCW: weighting fluence, current and yield

Argument list (all variables are input only)

IJ	:	particle type
PLA	:	particle momentum (if > 0.0) or -PLA = kinetic energy (if < 0.0)
TXX, TYY, TZZ	:	particle current direction cosines
WEE	:	particle weight
XX, YY, ZZ	:	particle position
NRGFLK	:	current region (after boundary crossing)
IOLREG	:	previous region (before boundary crossing). Useful only with boundary crossing estimators (for other estimators it has no meaning)
LLO	:	particle generation
NSURF	:	internal code calling flag (not for general use)

Function FLUSCW is activated by option USERWEIG (p. 209), with WHAT(3) > 0.0. Yields obtained via USRYIELD (p. 236), fluences calculated with USRBDX, USRTRACK, USRCOLL, USRBIN (respectively p. 211, 231, 227, 218), and currents calculated with USRBDX are multiplied by the value returned by this function. The user can implement any desired logic to differentiate the returned value according to any information contained in the argument list (particle type, energy, direction, weight, position, region, boundary, particle generation), or information available in COMMON SCOHLP (binning or detector number, estimator type). The estimator type is given by the flag ISCRNG (in COMMON SCOHLP):

ISCRNG = 1 ⟶ Boundary crossing estimator
ISCRNG = 2 ⟶ Track-length binning
ISCRNG = 3 ⟶ Track-length estimator
ISCRNG = 4 ⟶ Collision density estimator
ISCRNG = 5 ⟶ Yield estimator

The binning/detector number is given by JSCRNG (in COMMON SCOHLP) and is printed in output:

```
Bdrx n. 2  "bdxname" , generalised particle n. 8, from region n. 22 to region n. 78
Track n. 6  "trkname" , generalised particle n. 14, region n.   9
```

Note that a track-length detector can have the same JSCRNG number as a boundary crossing one or a binning etc. (use the value of ISCRNG to discriminate the different estimators). Further information can be obtained including COMMON TRACKR (for instance particle age). TRACKR contains also special user variables (both integer and in double precision) which can be used to save information about particles which have undergone some particular event.

Function FLUSCW has many applications. A common one in shielding calculations is to multiply selected scored fluences by particle/energy-dependent fluence-to-dose equivalent conversion coefficients, or by some instrument response, radiation damage curve, etc. Another application is conditional scoring (score only if within a certain distance from a point, etc.): for instance it is possible to implement a sort of 2-dimensional fluence binning on a plane boundary.

Other interesting applications are based on the fact that FLUSCW is called at every boundary crossing, provided that at least one USRBDX detector has been requested. Although the function has been designed mainly to weight scored quantities, it can be "cheated" to do all sorts of side things, even not directly connected with scoring.

Note: setting the variable LSCZER = .TRUE. before RETURN (LSCZER is in COMMON SCOHLP), will cause zero scoring whatever the value returned by COMSCW or FLUSCW. This is more efficient than returning a zero value.

12.2.7 FORMFU: nuclear FORM Factor User-defined

Argument list (all variables are input only)		
IJ	:	particle code, except that it is set to 3 for both e^+ and e^-
QU2	:	squared momentum transfer $(\mathrm{GeV/c})^2$
ZMEDIU	:	atomic number of target nucleus
AMEDIU	:	atomic mass of target nucleus

Function FORMFU can be used to override the standard value of the nuclear charge form factor. It must return the squared value of the nuclear charge form factor for particle IJ. The default version computes the form factor in Born approximation for a medium of given composition, using the simple expression given by Tsai [170], and accounts also for the contribution of incoherent scattering.
The function is called by the multiple and single scattering routines if option MULSOPT (p. 153) has been issued with WHAT(3) $<$ 0.0 for electrons and positrons, or WHAT(2) $<$ 0.0 for hadrons and muons. See Note 2 to option MULSOPT, p. 155.

12.2.8 FRGHNS: material roughness (for optical photons)

Argument list (all variables are input only)		
TXX, TYY, TZZ	:	particle direction cosines
UXSRFC, UYSRFC, UZSRFC	:	direction of the normal to the surface
MREG	:	region from which the particle is coming
NEWREG	:	region to which the particle is going
MMAT	:	material of the region from which the particle is coming
MMATNW	:	material of the region to which the particle is going

Function FRGHNS can be used to return a non-zero value for the roughness of a boundary between two materials, relevant for optical photon transport (default roughness is zero for all boundaries). Meaningful only if options OPT–PROP or OPT–PROD (p. 164, 160) have been requested. See Sec. 7.51 and Chap. 13 for more information.

12.2.9 MUSRBR, LUSRBL, FUSRBV: user defined quantities for special binning

These three functions are used to define 3-dimensional fluence distributions to be calculated by special user-defined binnings (see option USRBIN with WHAT(1) = 8.0 in the first card).

MUSRBR defines a discrete (integer) variable (by default: region number).

Argument list (all variables are input only)		
IJ	:	particle type
PCONTR	:	particle momentum
XA, YA, ZA	:	particle position
MREG	:	current region
LATCLL	:	current lattice cell
ICALL	:	internal code calling flag (not for general use)

LUSRBL defines another discrete (integer) variable (by default: lattice number)

Argument list (all variables are input only)

IJ : particle type
PCONTR : particle momentum
XFLK, YFLK, ZFLK : particle position
MREG : current region
LATCLL : current lattice cell
ICALL : internal code calling flag (not for general use)

FUSRBV defines a continuous (double precision) variable (by default: pseudorapidity with respect to the Z axis)

Argument list (all variables are input only)

IJ : particle type
PCONTR : particle momentum
XFLK, YFLK, ZFLK : particle position
MREG : current region
ICALL : internal code calling flag (not for general use)

The 3 functions are called at track-length events. What is scored is the particle track-length multiplied by the particle's weight, possibly modified by a user-written FLUSCW (see 12.2.6), as a function of the 3 variables defined by MUSRBR, LUSRBL and FUSRBV.

12.2.10 LATTIC: symmetry transformation for lattice geometry

Subroutine LATTIC is activated by one or more LATTICE cards in the geometry input (see 8.2.9). It is expected to transform coordinates and direction cosines from any lattice cell (defined by card LATTICE) to the reference system in which the basic structure has been described.

The user is expected to provide a transformation of coordinates and vector direction cosines from each lattice cell to the corresponding basic structure (in ENTRY LATTIC) and of direction cosines from the basic structure to each corresponding lattice cell (in ENTRY LATNOR).

Entries:

LATTIC (position and direction symmetry transformation from lattice cell to prototype structure)

Argument list

XB(1), XB(2), XB(3) : actual physical position coordinates in IRLTGG lattice cell
WB(1), WB(2), WB(3) : actual physical direction cosines in IRLTGG lattice cell
DIST : reserved variable
SB(1), SB(2), SB(3) : transformed coordinates in prototype cell
UB(1), UB(2), UB(3) : transformed cosines in prototype cell
IR : region number in prototype cell
IRLTGG : lattice cell number
IRLT : array containing region indices corresponding to lattice cells
IFLAG : reserved variable

LATTIC returns the tracking point coordinates (SB) and direction cosines (UB) in the reference prototype geometrical structure, corresponding to real position/direction XB, WB in the actual cell IRLTGG (defined as input region IR by a LATTICE card).

When the lattice option is activated, the tracking proceeds in two different systems: the "real" one, and that of the basic symmetry unit. Particle positions and directions are swapped from their real values to their symmetric ones in the basic cell, to perform the physical transport in the regions and materials that form the prototype geometrical structure and back again to the real world. The correspondence between "real" and "basic" position/direction depends on the symmetry transformation and on the lattice cell number.

LATNOR (LATtice cell NORmal transformation from prototype structure to lattice cell)

Argument list

UN(1), UN(2), UN(3) : direction cosines of the vector normal to the surface, in the prototype cell (entry values) and in the lattice cell (returned values)
IRLTNO : present lattice cell number

ENTRY LATNOR transforms the direction cosines stored in the vector UN(3) from the system of the basic prototype unit to that of the real world in lattice cell number IRLTNO. Therefore, this cosine transformation must be the inverse of that performed on the cosines by the LATTIC entry: but while LATTIC maps vector UB to a different vector WB, LATNOR maps the UN vector to itself.

Note that if the transformation implies a rotation, it is necessary to save first the incoming UN cosines to local variables, to avoid overwriting the vector before all transformation statements are executed.

Notes

1) Different symmetry transformations can of course be implemented in the same LATTIC routine (each being activated by a different cell number or range of cell numbers).
2) The advantage of the lattice geometry is to avoid describing in detail the geometry of repetitive multi-modular structures. It must be realised, however, that a penalty is generally paid in computer efficiency.
3) Also, a region contained in the prototype cell and all those "mirrored" to it inside lattice cells are treated by the program as if they were connected with "non-overlapping ORs" (see 8.2.6.2, 8.2.6.4) into a single region. Therefore, any region-based scoring (options SCORE, USRTRACK, etc.) can only provide quantities averaged over the whole structure. More detailed information must be obtained by region-independent options such as USRBIN or by user-written routines (MGDRAW, see 12.2.13). The USRBIN and EVENTBIN options (p. 218, 111) can also be set with the special binning type selected with WHAT(1) = 8, which activates the MUSRBR, LUSRBL, FUSRBV user routines to recover lattice information (see 12.2.9).

12.2.11 MAGFLD: definition of a magnetic field

Argument list

X, Y, Z : current position (input only)
BTX, BTY, BTZ : direction cosines of the magnetic field vector (returned)
B : magnetic field intensity in tesla (returned)
NREG : current region (input only)
IDISC : if returned = 1, the particle will be discarded

MAGFLD is activated by option MGNFIELD (p. 151) with WHAT(4–6) = 0.0 and is used to return intensity and direction of a magnetic field based on the current position and region. It is called only if the current region has been flagged as having a non-zero magnetic field by option ASSIGNMAt (p. 62), with WHAT(5) = 1.0.

The magnetic field spatial distribution is often read and interpolated from an external field map. Note that in any case the direction cosines *must* be properly normalised in double precision (e.g., `BTX = SQRT(ONEONE - BTY**2 - BTZ**2)`), even if `B = 0.0`.

Please read carefully the notes on option MGNFIELD (p. 151).

12.2.12 MDSTCK: management of the stack of secondaries

Argument list		
IFLAG	:	type of nuclear interaction which has produced secondaries: 1: inelastic 2: elastic 3: low-energy neutron
NUMSEC	:	number of secondary particles produced in the interaction

MDSTCK is called after a nuclear interaction in which at least one secondary particle has been produced, before any biasing is applied, to decide which secondary will be loaded in the main stack for further transport. The properties of the secondaries are stored in the secondary stack (COMMON GENSTK). With MDSTCK, users can analyse those secondaries, write them to a file, or even modify the content of GENSTK (for instance applying their own biasing). In the latter case, however, it is their responsibility to make sure that energy is conserved, the various physical quantities are still consistent, etc.

12.2.13 MGDRAW: general event interface

Subroutine MGDRAW, activated by option USERDUMP (p. 208) with WHAT(1) $\geq$ 100.0, usually writes a "collision tape", i.e., a file where all or selected transport events are recorded. The default version (unmodified by the user) offers several possibilities, selected by WHAT(3) in USERDUMP. Details are given in Chap. 11.

Additional flexibility is offered by a user entry USDRAW, interfaced with the most important physical events happening during particle transport. The user can modify of course also any other entry of this subroutine (BXDRAW called at boundary crossings, EEDRAW called at event end, MGDRAW for trajectory drawing, ENDRAW for recording of energy deposition events, and SODRAW for recording of source events): for instance the format of the output file can be changed, and different combinations of events can be written to file.

But the most interesting aspect of the routine is that the six entries (all of which, if desired, can be activated at the same time by setting USERDUMP with WHAT(3) = 0.0 and WHAT(4) $\geq$ 1.0) constitute a complete interface to the whole FLUKA transport. Therefore, MGDRAW can be used not only to write a collision tape, but to do any kind of complex analysis (for instance studying correlations between events).

Entries:

MGDRAW (trajectory dumping for drawing)

Argument list (all variables are input only)		
ICODE	:	FLUKA physical compartment originating the call = 1: call from subroutine KASKAD (hadrons and muons) = 2: call from subroutine EMFSCO (e^-, e^+ and photons) = 3: call from subroutine KASNEU (low-energy neutrons) = 4: call from subroutine KASHEA (heavy ions) = 5: call from subroutine KASOPH (optical photons)
MREG	:	current region

MGDRAW writes by default, for each trajectory, the following variables (contained in COMMON TRACKR):
NTRACK : number of track segments

MTRACK : number of energy deposition events along the track
JTRACK : type of particle
ETRACK : total energy of the particle
WTRACK : weight of the particle
NTRACK values of XTRACK, YTRACK, ZTRACK: end of each track segment
MTRACK values of DTRACK: energy deposited at each deposition event
CTRACK : total length of the curved path

Other variables are available in TRACKR (but not written by MGDRAW unless the latter is modified by the user: particle momentum, direction cosines, cosines of the polarisation vector, age, generation, etc. (see a full list in the comment in the INCLUDE file).

BXDRAW (boundary crossing dumping)

Argument list (all variables are input only)

ICODE : physical compartment originating the call, as in the MGDRAW entry
MREG : region from which the particle is exiting
NEWREG : region the particle is entering
XSCO, YSCO, ZSCO : point where the boundary crossing occurs

No output by default.

EEDRAW (event end dumping)

Argument list (all variables are input only)

ICODE : physical compartment originating the call, as in the MGDRAW entry

No output by default.

ENDRAW (energy deposition dumping)

Argument list (all variables are input only)

ICODE : type of event originating energy deposition
ICODE = 1x: call from subroutine KASKAD (hadrons and muons);
= 10: elastic interaction recoil
= 11: inelastic interaction recoil
= 12: stopping particle
= 14: particle escaping (energy deposited in blackhole)
ICODE = 2x: call from subroutine EMFSCO (electrons, positrons and photons)
= 20: local energy deposition (i.e. photoelectric)
= 21 or 22: particle below threshold
= 23: particle escaping (energy deposited in blackhole)
ICODE = 3x: call from subroutine KASNEU (low-energy neutrons)
= 30: target recoil
= 31: neutron below threshold
= 32: neutron escaping (energy deposited in blackhole)
ICODE = 4x: call from subroutine KASHEA (heavy ions)
= 40: ion escaping (energy deposited in blackhole)
ICODE = 5x: call from subroutine KASOPH (optical photons)
= 50: optical photon absorption
= 51: optical photon escaping (energy deposited in blackhole)
MREG : current region
RULL : energy amount deposited
XSCO, YSCO, ZSCO : point where energy is deposited

ENDRAW writes by default, for each energy deposition point:
0 : flag identifying ENDRAW output from that of other entries

ICODE : see argument list
JTRACK, ETRACK, WTRACK : see MGDRAW above
XSCO, YSCO, ZSCO, RULL : see argument list.

SODRAW (source particle dumping)

Argument list
No arguments

SODRAW writes by default, for each source or beam particle:
-NCASE (in COMMON CASLIM), with a minus sign to identify SODRAW output): number of primaries followed so far
NPFLKA (in COMMON FLKSTK): stack pointer
NSTMAX (in COMMON FLKSTK): highest value of the stack pointer encountered so far
TKESUM (in COMMON SOURCM): total kinetic energy of the primaries of a user written source (see user subroutine SOURCE in 12.2.19), if applicable. Otherwise = 0.0
WEIPRI (in COMMON SUMCOU): total weight of the primaries handled so far

NPFLKA times:	ILOFLK:	type of source particle
(all variables in	TKEFLK + AM:	total particle energy (kinetic+mass)
COMMON FLKSTK)	WTFLK:	source particle weight
	XFLK, YFLK, ZFLK:	source particle position
	TXFLK, TYFLK, TZFLK:	source particle direction cosines

USDRAW (user-defined dumping)

Argument list (all variables are input only)
ICODE : type of event
ICODE = 10x: call from subroutine KASKAD (hadron and muon interactions);
= 100: elastic interaction secondaries
= 101: inelastic interaction secondaries
= 102: particle decay secondaries
= 103: delta ray generation secondaries
= 104: pair production secondaries
= 105: bremsstrahlung secondaries
ICODE = 20x: call from subroutine EMFSCO (electron, positron and photon interactions)
= 208: bremsstrahlung secondaries
= 210: Møller secondaries
= 212: Bhabha secondaries
= 214: in-flight annihilation secondaries
= 215: annihilation at rest secondaries
= 217: pair production secondaries
= 219: Compton scattering secondaries
= 221: photoelectric secondaries
= 225: Rayleigh scattering secondaries
ICODE = 30x: call from subroutine KASNEU (low-energy neutron interactions)
= 300: neutron interaction secondaries
ICODE = 40x: call from subroutine KASHEA (heavy ion interactions)
= 400: delta ray generation secondaries
MREG : current region
XSCO, YSCO, ZSCO : interaction point

USDRAW is called after each particle interaction (if requested by the user with option USERDUMP, WHAT(4) $\geq$ 1.0). There is no default output: any output must be supplied by the user.

Information about the secondary particles produced is available in COMMON GENSTK, except that concerning delta rays produced by heavy ions (in which case the properties of the single electron produced are available in COMMON EMFSTK, with index NP).

Information about the interacting particle and its trajectory can be found in COMMON TRACKR (see description under the MGDRAW entry above). In TRACKR there are also some spare variables at the user's disposal: LLOUSE (integer), ISPUSR (integer array) and SPAUSR (double precision array). Like many other TRACKR variables, each of them has a correspondent in the particle stacks, i.e., the COMMONs from which the particles are unloaded at the beginning of their transport: FLKSTK, EMFSTK and OPPHST (respectively, the stack of hadrons/muons, electrons/photons, and optical photons). The correspondence with TRACKR is shown below under STUPRF/STUPRE (12.2.20). When a particle is generated, its properties (weight, momentum, energy, coordinates etc., as well as the values of the user flags) are loaded into one of the stacks. The user can write a STUPRF or STUPRE subroutine (see description below in 12.2.20) to change anyone of such flags just before it is saved in stack.

When a particle starts to be transported, its stack variables are copied to the corresponding TRACKR ones. Unlike the other TRACKR variables, which in general become modified during transport due to energy loss, scattering etc., the user flags keep their original value copied from stack until they are changed by the user himself (generally under the USDRAW entry).

One common application is the following: after an interaction which has produced secondaries, let USDRAW copy some properties of the interacting particle into the TRACKR user variables. When STUPRF is called next to load the secondaries into stack, by default it copies the TRACKR user variables to the stack ones. In this way, information about the parent can be still carried by its daughters (and possibly by further descendants). This technique is sometimes referred to as "latching".

12.2.14 OPHBDX: user defined Optical PHoton BounDary-(X)crossing properties

Argument list (all variables are input only)

OMGPHO :	angular frequency ($\omega = 2\pi\nu$) of the photon (in s^{-1})
WVLNGT :	photon wavelength (in cm)
MREG :	old region number
NEWREG :	new region number
SIGANW :	absorption coefficient in the new region (cm^{-1})
SIGDNW :	diffusion coefficient in the new region (cm^{-1})
RFNDPR :	refractive index in the new region
VGRPNW :	group velocity in the new region (cm s^{-1})
LPHKLL :	if .TRUE., the photon will be absorbed on the boundary

Subroutine OPHBDX sets the optical properties of a boundary surface. The call is activated by command OPT–PROP, with SDUM = SPEC–BDX. See Sec. 7.51 and Chap. 13 for more information.

12.2.15 QUEFFC: user defined QUantum EFFiCiency

Argument list (all variables are input only)

WVLNGT :	photon wavelength (in cm)
OMGPHO :	angular frequency ($\omega = 2\pi\nu$) of the photon (in s^{-1})

Function QUEFFC returns a user-defined quantum efficiency for an optical photon of the given wavelength or frequency.
It is activated with option OPT–PROP with SDUM = SENSITIV, by setting the 0^{th} photon sensitivity parameter to a value < -99. See Sec. 7.51 and Chap. 13 for more information.

12.2.16 RFLCTV: user defined ReFLeCTiVity

Argument list (all variables are input only)	
WVLNGT :	photon wavelength (in cm)
OMGPHO :	angular frequency ($\omega = 2\pi\nu$) of the photon (in s^{-1})
MMAT :	material index

Function RFLCTV returns a user-defined of the current material for an optical photon of the given wavelength or frequency..
It is activated by command OPT–PROP with SDUM = METAL and WHAT(3) < -99. See Sec. 7.51 and Chap. 13 for more information.

12.2.17 RFRNDX: user defined ReFRaction iNDeX

Argument list (all variables are input only)	
WVLNGT :	photon wavelength (in cm)
OMGPHO :	angular frequency ($\omega = 2\pi\nu$) of the photon (in s^{-1})
MMAT :	material index

Function RFRNDX returns a user-defined refraction index of the current material for an optical photon of the given wavelength or frequency.
It is activated by command OPT–PROP with SDUM = blank and WHAT(1) < -99. See Sec. 7.51 and Chap. 13 for more information.

12.2.18 SOEVSV: SOurce EVent SaVing

Argument list
No arguments

Subroutine SOEVSV is always called after a beam particle is loaded onto stack, but a call to SOEVSV can be inserted by the user anywhere in a user routine.

SOEVSV copies the whole COMMON FLKSTK to another COMMON, SOUEVT, which can be included in other user routines. In other words, this routine is used to "take a snapshot" of the particle bank at a particular time for further use (interfacing to independent generators, etc.)

12.2.19 SOURCE: user-written source

Argument list	
NOMORE :	if set = 1, no more calls will occur (the run will be terminated after exhausting the primary particles loaded onto stack in the present call). The history number limit set with option START (p. 199) will be overridden

Subroutine SOURCE is probably the most frequently used user routine. It is activated by option SOURCE (p. 197) and is used to sample primary particle properties from distributions (in space, energy, time, direction or mixture of particles) too complicated to be described with the BEAM, BEAMPOS and BEAMAXES cards alone. For each phase-space variable, a value must be loaded onto COMMON FLKSTK (particle bank) before returning control. These values can be read from a file, generated by some sampling algorithm, or just assigned.

12.2.19.1 Reading from a file

Reading from a file is needed, for instance, when the particle data are taken from a collision file, written by FLUKA or by another program (see 11). The user must open the file with a unit number > 20.0 (unit numbers lower than 20 are reserved), in one of the following ways:

1) Using option OPEN (p. 158), with SDUM = OLD
2) In a user subroutine USRINI (see 12.2.26 below), with a Fortran OPEN statement. Option USRICALL (p. 229) is needed to activate the call to the routine.
3) With an OPEN statement in the initialisation part of subroutine SOURCE itself.

Then, a READ statement in SOURCE can be used to get the data to load in stack, for instance:

```
READ(21,*) IPART, X, Y, Z, COSX, COSY, COSZ, ENERGY, WEIGHT
ILOFLK (NPFLKA) = IPART
XFLK   (NPFLKA) = X
YFLK   (NPFLKA) = Y
ZFLK   (NPFLKA) = Z
TXFLK  (NPFLKA) = COSX
```

. . . etc. . .
(NPFLKA is the current stack index).

12.2.19.2 Direct assignment

Direct assignment can be done explicitly, for instance:

```
PMOFLK (NPFLKA) = 305.2D0
```

or implicitly, leaving unmodified values input with BEAM (p. 64) or BEAMPOS (p. 69):

```
PMOFLK (NPFLKA) = PBEAM
```

(PBEAM is the momentum value input as WHAT(1) in option BEAM). A set of direct assignments, one for each of several different stack entries, can be useful, for example, to define a series of RAYs through the geometry (see 14):

```
      DO 10 I = 1, 20
         NPFLKA = NPFLKA + 1
         ILOFLK (NPFLKA) = 0               ! (0 is the RAY particle id number)
         XFLK   (NPFLKA)  = 500.D0 + DBLE(I) * 40.D0
         YFLK   (NPFLKA)  = 200.D0
         ...etc...
   10 CONTINUE
```

12.2.19.3 Sampling from a uniform distribution

To sample from a uniform distribution, the user must use the function FLRNDM(DUMMY), which returns a double precision pseudo-random number uniformly distributed between 0 (included) and 1 (not included). Actually, DUMMY can be any variable name. A simple example of sampling from a uniform distribution is that of a linear source along the Z axis, between Z = 10 and Z = 80:

```
Z1 = 10.D0
Z2 = 80.D0
ZFLK (NPFLKA) = 10.D0 + (Z2 - Z1) * FLRNDM(XXX)
```

12.2.19.4 Sampling from a generic distribution

One way to sample a value X from a generic distribution $f(x)$ is the following.
First integrate the distribution function, analytically or numerically, and normalise to 1 the obtained cumulative distribution:

$$F(x) = \frac{\int_{x_{min}}^{x} f(x)\mathrm{d}x}{\int_{x_{min}}^{x_{max}} f(x)\mathrm{d}x}$$

Then, sample a uniform pseudo-random number ξ using `FLRNDM` and get the desired result by finding the inverse value $X = F^{-1}(\xi)$ (analytically or most often by interpolation).

A FLUKA subroutine is available to sample directly from a Gaussian distribution:

```
CALL FLNRRN (RGAUSS)
```

or, if two independent Gaussian distributed numbers are needed:

```
CALL FLNRR2 (RGAUS1, RGAUS2)
```

(faster than calling `FLNRRN` twice).

12.2.19.5 Sampling from a biased distribution

The technique for sampling from a generic distribution described above can be extended to modify the probability of sampling in different parts of the interval (importance sampling). We replace $f(x)$ by a weighted function $g(x) = f(x)\,h(x)$, where $h(x)$ is any appropriate function of x we like to choose. We normalise $g(x)$ in the same way as $f(x)$ before:

$$G(x) = \frac{\int_{x_{min}}^{x} g(x)\mathrm{d}x}{\int_{x_{min}}^{x_{max}} g(x)\mathrm{d}x} = \frac{\int_{x_{min}}^{x} f(x)\mathrm{d}x\,h(x)\mathrm{d}x}{B}$$

and we need also the integral of $f(x)$ over the whole interval:

$$A = \int_{x_{min}}^{x_{max}} f(x)\mathrm{d}x$$

All the sampling is done using the biased cumulative normalised function G instead of the original unbiased F: we sample a uniform pseudo-random number ξ as before, and we get the sampled value X by inverting $G(x)$:

$$X = G^{-1}(\xi)$$

The particle is assigned a weight $\frac{B}{A\,h(X)}$.

A special case of importance sampling is when the biasing function chosen is the inverse of the unbiased distribution function:

$$h(x) = \frac{1}{f(x)}g(x) = f(x)\,h(x) = 1G(x) = \frac{x - x_{min}}{x_{max} - x_{min}}$$

In this case we sample a uniform pseudo-random number t using `FLRNDM` as shown above. The sampled value X is simply given by:

$$X = x_{min} + (x_{max} - x_{min})\,t$$

and the particle is assigned a weight $= f(X)(x_{max} - x_{min})$. Because X is sampled with the same probability over all possible values of x, independently of the value $f(X)$ of the function, this technique is used to ensure that sampling is done uniformly over the whole interval, even though $f(x)$ might have very small values somewhere. For instance it may be important to avoid undersampling in the high-energy tail of a spectrum, steeply falling with energy but more penetrating, such as that of cosmic rays or synchrotron radiation.

Option SOURCE (p. 197) allows the user to input up to 12 numerical values (WHASOU(1),(2)...(12)) and one 8-character string (SDUSOU) which can be accessed by the subroutine by including the following lines:

```
      INCLUDE '(SOURCM)'
      INCLUDE '(CHEPSR)'
```

These values can be used as parameters or switches for a multi-source routine capable to handle several cases, or to identify an external file to be read, etc., without having to compile and link again the routine.

In the SOURCE routine there are a number of mandatory statements, (clearly marked as such in accompanying comments) which must not be removed or modified. The following IF block initialises the total kinetic energy of the primary particles and sets two flags: the first to skip the IF block in all next calls, and the second to remind the program, when writing the final output, that a user source has been used:

```
*  +-------------------------------------------------------------------*
*  |  First call initialisations:
      IF ( LFIRST ) THEN
*  |  *** The following 3 cards are mandatory ***
         TKESUM = ZERZER
         LFIRST = .FALSE.
         LUSSRC = .TRUE.
*  |  *** User initialisation ***
      END IF
*  |
*  +-------------------------------------------------------------------*
```

The user can insert into the above IF block any other initialisation needed, for instance the preparation of a cumulative spectrum array from which to sample the energy of the source particles. Note that user initialisation can take place also in routine USRINI (activated at input time by input option USRICALL (p. 229), see 12.2.26) and USREIN (called before unloading from stack the first source particle of an event, i.e., just after the call to SOURCE: see 12.2.24)

At the time SOURCE is called, the particle bank FLKSTK is always empty and the stack pointer NPFLKA has value 0.

The user can load onto the FLKSTK stack one or more source particles at each call: for each particle loaded the pointer must be increased by 1. The template version of SOURCE loads only one particle: if several are loaded the following sequence, until the statement CALL SOEVSV not included, must be repeated once for each particle, possibly inside a DO loop:

```
      NPFLKA = NPFLKA + 1           !  increases the pointer
```

The following statements assign a value to each of the FLKSTK stack variables concerning the particle being loaded.

`WTFLK (NPFLKA) = ONEONE`
sets the weight of the particle = 1.0
This must be changed if the sampling of one or more of the particle properties are biased. In that case, generally the weight must be set after the sampling, and its value depends on the sampling outcome.

`WEIPRI = WEIPRI + WTFLK (NPFLKA)`
updates the total weight of the primaries (don't change)

`ILOFLK (NPFLKA) = IJBEAM`
by default sets the type of particle equal to the one defined by the BEAM card (p. 64). If no BEAM card is given in input, IJBEAM is = 1 (proton).

The above statement is followed by several others that must not be changed or removed. In the template routine, they are encompassed by the comment lines:

```
From this point .... / ... to this point: don't change anything
```

These statements are:

```
* From this point ....
      LOFLK (NPFLKA) = 1             !  Generation is 1 for source particles
      LOUSE (NPFLKA) = 0             !  User variables: the user can set
      DO 100 ISPR = 1, MKBMX1        !  different values in the STUPRF or
         SPAREK (1,NPFLKA) = ZERZER  !  STUPRE routine, but it is better
 100  CONTINUE                       !  not to do it here
      DO 200 ISPR = 1, MKBMX2
         ISPARK (ISPR,NPFLKA) = 0    !  More user variables (integer)
 200  CONTINUE
      NPARMA = NPARMA + 1            !  Updating the maximum particle number
      NUMPAR (NPFLKA) = NPARMA       !  Setting the current particle number
      NEVENT (NPFLKA) = 0            !  Resetting the current event number
      DFNEAR (NPFLKA) = +ZERZER      !  Resetting the distance to the
                                     !  nearest boundary
* ... to this point: don't change anything
```

The following statements can be overridden or rewritten by the user, assigning new values or sampling them from problem-dependent distributions.

First three statements which are rarely modified:

```
      AGESTK (NPFLKA) = +ZERZER       !  Particle age is zero by default
      AKNSHR (NPFLKA) = -TWOTWO       !  Resets the Kshort component of
                                      !  K0/K0bar. Usually not to be changed.
      IGROUP (NPFLKA) = 0             !  Group number for low-energy
                                      !  neutrons: if set to 0, the program
                                      !  derives it from the kinetic energy
```

Then the most frequently changed lines: both energy and momentum of the particle must be loaded onto the FLKSTK stack, but the two cannot be defined independently. Appropriate kinematical (relativistic) relations must be applied to derive one from the other.

In the template routine, the momentum is assumed to be assigned by BEAM option (its value, PBEAM, is taken from COMMON BEAMCM, which contains all values defined by options BEAM and BEAMPOS).

```
      PMOFLK (NPFLKA) = PBEAM
```

Therefore, the kinetic energy (in GeV) must be derived:

```
      TKEFLK (NPFLKA) = SQRT ( PBEAM**2 + AM (IJBEAM)**2 ) - AM (IJBEAM)
```

(where AM is the rest mass, in COMMON PAPROP, and IJBEAM is the particle type, in COMMON BEAMCM).

If instead the energy had been sampled first from some spectrum, and ENSAMP would be the sampled value, the two statements above would become:

```
      TKEFLK (NPFLKA) = ENSAMP
      PMOFLK (NPFLKA) = SQRT(ENSAMP * (ENSAMP + TWOTWO * AM(IJBEAM)))
```

The direction cosines are loaded next:

```
      TXFLK (NPFLKA) = UBEAM             !  Assumed here to be the same as
      TYFLK (NPFLKA) = VBEAM             !  defined by option BEAMPOS. UBEAM,
      TZFLK (NPFLKA) = WBEAM             !  VBEAM, WBEAM are some among the beam
                                         !  properties in COMMON BEAMCM
```

(If BEAMPOS is not given, by default UBEAM = VBEAM = 0.0, WBEAM = 1.0)

Remember to make sure that the cosines are normalised! One could replace the last statement by:

```
TZFLK (NPFLKA) = SQRT ( ONEONE - TXFLK(NPFLKA)**2 - TYFLK(NPFLKA)**2 )
```

The polarisation cosines are not set by default:

```
TXPOL (NPFLKA) = -TWOTWO        ! -2 is a flag for "no polarisation"
TYPOL (NPFLKA) = +ZERZER
TZPOL (NPFLKA) = +ZERZER
```

but appropriate values need to be given in some cases, for instance in synchrotron radiation shielding problems.

Finally the particle coordinates, set again by default equal to those input with BEAMPOS:

```
XFLK (NPFLKA) = XBEAM             !  Assumed here to be the same as
YFLK (NPFLKA) = YBEAM             !  defined by option BEAMPOS. XBEAM,
ZFLK (NPFLKA) = ZBEAM             !  YBEAM, ZBEAM are also in COMMON BEAMCM
```

(If BEAMPOS is not given, by default `XBEAM = YBEAM = ZBEAM = 0.0`).

If for example our problem required instead a linear source uniformly distributed along Z between `Z1` and `Z2`, we could replace the last statement by:

```
ZFLK (NPFLKA) = Z1 + FLRNDM(UGH) * (Z2 - Z1)
```

The following lines in the template `SOURCE` routine should never be changed. They calculate the total energy of the primary particles, define the remaining properties of the particles (starting region and lattice cell) and do some geometry initialisation.
The last line calls the `SOEVSV` user routine (see description in 12.2.18 above) to save the stack for possible further use.

Important!

The values of beam characteristics defined by commands BEAM (p. 64) and POLARIZAti (p. 182) are available in `COMMON BEAMCM`: the angular divergence (variable `DIVBM`), beam width (`XSPOT` and `YSPOT`), and the polarisation vector (`UBMPOL`, `VBMPOL`, `WBMPOL`) can help to set up a scheme to sample the corresponding quantities from user-defined distributions. But sampling from the distributions pre-defined by BEAM and POLARIZAti is not simply inherited by subroutine `SOURCE`: *it is the responsibility of the user to write such a scheme!*
For this task, it may be useful to define a "beam reference frame" by means of option BEAMAXES (see more details on p. 67).

12.2.20 STUPRE, STUPRF: SeT User PRoperties for EMF and FLUKA particles

These two functions are used to assign a value to one or more stack user variables when the corresponding particle is loaded onto one of the stacks (`FLKSTK` for hadrons/muons, and `EMFSTK` for electrons/photons.

In each of these stacks the user has access to one integer variable, one integer array and one double precision array. Each of them is copied to a correspondent variable or array in `COMMON TRACKR` at the beginning of transport:

Correspondence	FLKSTK	EMFSTK		TRACKR
integer variable:	LOUSE	LOUEMF	$\longrightarrow$	LLOUSE
integer array:	ISPARK	IESPAK	$\longrightarrow$	ISPUSR
double precision array:	SPAREK	ESPARK	$\longrightarrow$	SPAUSR

The user can access and modify the `TRACKR` variables via subroutine `MGDRAW` and its entries `ENDRAW`, `SODRAW` and especially `USDRAW` (see description above). `STUPRF` and `STUPRE` can be used to do the reverse, namely to copy `TRACKR` user variables to those of the relevant stack (see `USDRAW` above).

Note that a stack OPPHST exists also for optical photons, containing similar user variables and arrays LOUOPP, ISPORK and SPAROK. They can be used in user routines, but they are not handled by STUPRE or STUPRF.

STUPRE is called before loading into stack electrons, positrons and photons.

Argument list
No arguments

The default version does nothing (the user variables of the parent particle are already set equal to the original projectile by the various electromagnetic interaction routines. Also the region/position etc. are already set inside the stack arrays.

STUPRF is called before loading into stack hadrons, muons, neutrinos and low-energy neutrons

Argument list	
IJ	: type of the parent particle
MREG	: current region
XX, YY, ZZ	: particle position
NUMSEC	: index in COMMON GENSTK of the secondary being loaded onto stack
NPPRMR	: if $>$ 0, the secondary being loaded is actually still the interacting particle (it can happen in some biasing situations)

The default version copies to stack the user flags of the parent.

12.2.21 UBSSET: User BiaSing SETting

Argument list	
IR	: region number
RRHADR	: multiplicity biasing factor to be applied to the secondaries from hadronic interactions in region IR (WHAT(2) of card BIASING, p. 71)
HMPHAD	: Importance of region IR for hadrons and muons (WHAT(3) of card BIASING, with WHAT(1) = 0.0 or 1.0). Actually the routine argument is an integer, IMPHAD, equal to importance multiplied by 10000, but the user should consider only the double precision version HMPHAD (a conversion from and to the integer version is provided at the beginning and at the end of the routine, and should not be changed)
HMPLOW	: Importance of region IR for low-energy neutrons (WHAT(3) of card BIASING, with WHAT(1) = 0.0 or 3.0). Actually the routine argument is an integer, IMPLOW, equal to importance multiplied by 10000, but the user should consider only the double precision version HMPLOW (a conversion from and to the integer version is provided at the beginning and at the end of the routine, and should not be changed)
HMPEMF	: Importance of region IR for electrons and photons (WHAT(3) of card BIASING, with WHAT(1) = 0.0 or 2.0). Actually the routine argument is an integer, IMPEMF, equal to importance multiplied by 10000, but the user should consider only the double precision version HMPEMF (a conversion from and to the integer version is provided at the beginning and at the end of the routine, and should not be changed)

Continues...

... Continuation of the UBSSET argument list

Argument list

IGCUTO : Cut-off group index for low-energy neutrons in region IR (WHAT(1) in card LOW–BIAS, p. 135)
IGNONA : Non-analogue absorption group limit for low-energy neutrons in region IR (WHAT(2) in card LOW–BIAS)
PNONAN : Non-analogue survival probability for low-energy neutrons in region IR (WHAT(3) in card LOW–BIAS)
IGDWSC : Group limit for biased downscattering for low-energy neutrons in region IR (WHAT(1) in card LOW–DOWN, p. 137)
FDOWSC : Biased downscattering factor for low-energy neutrons in region IR (WHAT(2) in card LOW–DOWN)
JWSHPP : Weight Window/importance profile index for low-energy neutrons in region IR (SDUM in WW–FACTOr, p. 244)
WWLOW : Weight Window lower level in region IR (WHAT(1) in card WW–FACTOr, possibly modified by WHAT(4) in WW–THRESh, p. 249 or WHAT(2) in WW–PROFIle, p. 247)
WWHIG : Weight-Window upper level in region IR (WHAT(2) in card WW–FACTOr, possibly modified by WHAT(4) in WW–THRESh or WHAT(2) in WW–PROFIle)
WWMUL : Weight-Window multiplicative factor applied to the two energy thresholds defined with WW–THRESh, for region IR (WHAT(3) in card WW–FACTOr)
EXPTR : Exponential transform parameter for region IR (WHAT(2) in card EXPTRANS, p. 117) (*not implemented yet!!!!!!!!*)
ELECUT : e^+, e^- cut-off in region IR (WHAT(1) in card EMFCUT)
GAMCUT : Photon cut-off in region IR (WHAT(2) in card EMFCUT, p. 102)
LPEMF : Leading Particle Biasing flag in region IR (SDUM = LPBEMF in card EMF–BIAS, p. 98, or WHAT(3) in card EMFCUT)
ELPEMF : Maximum e^+/e^- energy for applying Leading Particle Biasing (WHAT(2) in card EMF–BIAS with SDUM = LPBEMF)
PLPEMF : Maximum photon energy for applying leading particle biasing (WHAT(3) in card EMF–BIAS with SDUM = LPBEMF)

Subroutine UBSSET does not require a special command to be activated: is always called several times for each region: (once for every biasing option or suboption) after the end of input reading and before starting the calculations. The default version is a dummy and does nothing. The user can replace it to override any biasing parameters specified in input.

The UBSSET subroutine is used especially in cases with a large number of regions, because it allows to derive the biasing parameters from simple algorithms instead of entering each input value by hand. Choosing an appropriate numbering scheme for the geometry regions can often facilitate the task.

For instance, assuming a simple slab geometry with an expected exponential hadron attenuation from region 3 to region 20, each region being one half-value-layer thick, one could write the following in order to set importances that would keep the hadron number about constant in all regions:

```
IF(IR .GE. 3 .AND. IR .LE. 20) HMPHAD = ONEONE * TWOTWO**(IR-3)
```

It is important, however, not to do recursive assignments of the type:

```
GAMCUT(5) = GAMCUT(5) * HLFHLF
```

(HLFHLF is a FLUKA constant = 0.5D0) because that would halve the value of the photon cut-off for Region 5 *at every call*, and the number of calls is not known to the user.

12.2.22 UDCDRL: User defined DeCay DiRection biasing and Lambda (for ν only)

Argument list

input:

IJ	:	type of decaying particle
KPB	:	index of the concerned neutrino in the decay products list (may be changed in future releases)
NDCY	:	number of decay products

output:

UDCDRB, VDCDRB, WDCDRB	:	cosines of the preferential outgoing direction for the neutrino
UDCDRL		width of the distribution around the preferential direction

Function UDCDRL is used to bias the direction of a neutrino emitted by a decaying particle of type IJ event by event. The preferential direction axis is returned in the UDCDRB, VDCDRB, WDCDRB variables, and the value returned by UDCDRL is the λ for direction biasing: the zenith angle around the selected axis is sampled according to $\exp[(1-\cos(\theta))/\lambda]$.

12.2.23 USIMBS: USer defined IMportance BiaSing

Argument list

input:

MREG	:	region at the beginning of the step
NEWREG	:	region at the end of the step

output:

FIMP	:	returns the user-defined importance ratio between the position at the end and at the beginning of the step

Subroutine USIMBS is activated by card BIASING (p. 71) with SDUM = USER. The routine is called *at every particle step*. It can be used to implement any importance biasing scheme based on region number and on phase space coordinates and other information provided by COMMON TRACKR.
Warning: The user must balance the very effective biasing power offered by the routine with the important demand on CPU time due to the large number of calls.

12.2.24 USREIN: USeR Event INitialisation (called before each event)

Argument list

No arguments

Subroutine USREIN is called just before the first source particle of an event is unloaded from stack and begins to be transported. An event is the full history of a group of related particles and their descendants. If primaries are loaded into stack by the input option BEAM (p. 64), there is only one source particle per event; but there can be more if the user routine SOURCE is used to load particles into stack. USREIN does not need any special command to be activated, but the default version of USREIN does nothing: the user can write here any kind of initialisation.

12.2.25 USREOU: USeR Event OUtput (called at the end of each event)

Argument list

No arguments

Subroutine USREOU is called at the end of each event, namely after all event primary particles and their descendants have been transported. (See USREIN above for a definition of an event).
USREOU does not need any special command to be activated, but the default version of USREOU does nothing: the user can write here any kind of event analysis, output, etc.

12.2.26 USRINI: USeR INItialisation

Argument list

WHAT(1),(2),(3),(4),(5),(6) : user-provided numerical parameters
SDUM : user-provided character string (8 characters)

Subroutine USRINI is called every time a USRICALL (p. 229) card is read in input. It can be used to do any kind of initialisation: reading and manipulating data from one or more files, calling other private routines, etc.
The calling parameters can carry any kind of useful information or can be used as flags to choose between different possible actions to be performed before any particle transport takes place.

12.2.27 USRMED: USeR MEDium dependent directives

Argument list

IJ : particle type
EKSCO : particle kinetic energy (GeV)
PLA : particle momentum (GeV/c)
WEE : particle weight
MREG : previous region number
NEWREG : current region number
XX, YY, ZZ : particle position
TXX, TYY, TZZ : particle direction

Subroutine USRMED is activated by option MAT–PROP (p. 144) with SDUM = USERDIRE, for one or more materials indicated by the user. It is called every time a particle is going to be transported in one of the user-flagged materials.

Two cases are possible:

1) MREG = NEWREG: the particle is going to move from a point inside the medium. The user is allowed to change only the particle weight. Typical application: simulating attenuation of optical photons in an absorbing medium by reducing the photon weight.
2) MREG $\neq$ NEWREG: the particle is going to move from a point on a boundary between two different regions. The user may change any of the following: particle weight, current region number, direction cosines.

Typical applications:

- simulating refraction, by changing the direction cosines so that the particle is still inside the same region. To do this, one generally needs the direction cosines of the normal to the surface: TXNOR(NPFLKA), TYNOR(NPFLKA), TZNOR(NPFLKA) (COMMON FLKSTK must be included).
- simulating reflection (albedo) at a boundary. The direction cosines must be modified according to some reflection law or albedo angular distribution, and NEWREG must be set = MREG.

In both cases the weight can also be reduced to account for surface reflectivity or similar (if the particle is an optical photon, the FRGHNS user function (12.2.8) can be called to establish a surface roughness)

12.2.28 USROUT: USeR OUTput

Argument list		
WHAT(1),(2),(3),(4),(5),(6)	:	user-given numerical parameters
SDUM	:	user-given character string (8 characters)

Subroutine USROUT is called every time a USROCALL (p. 230) card is read in input. It is used to print special user-written output in addition to the standard one provided by default.
The calling parameters can carry any kind of useful information or can be used as flags to choose between different possible actions to be performed after all particle transport has taken place.

12.2.29 USRRNC: USeR Residual NuClei

Argument list		
IZ	:	Atomic number of the residual nucleus
IA	:	Mass number of the residual nucleus
IS	:	X, Y, Z : particle position
MREG	:	current region
WEE	:	particle weight
ICALL	:	internal code calling flag (not for general use)

Subroutine USRRNC is called every time a residual nucleus is produced

Chapter 13

Generating and propagating optical photons

FLUKA can be used to generate and propagate optical photons of Cherenkov, scintillation and transition radiation light. Light generation is switched off by default and is activated and totally controlled by the user by means of data cards and user routines. This is true also for the optical properties of materials. These include the refraction index as a function of wavelength (or frequency or energy), the reflection coefficient of a given material, etc.

In this respect, the user has the responsibility of issuing the right input directives: the code does not perform any physics check on the assumptions about the light yield and the properties of material.

Optical photons (FLUKA id = -1) are treated according to the laws of geometrical optics and therefore can be reflected and refracted at boundaries between different materials. From the physics point of view, optical photons have a certain energy (sampled according to the generation parameters given by the user) and carry along their polarisation information. Cherenkov photons are produced with their expected polarisation, while scintillation photons are assumed to be unpolarised. At each reflection or refraction, polarisation is assigned or modified according to optics laws derived from Maxwell equations.

At a boundary between two materials with different refraction index, an optical photon is propagated (refracted) or reflected with a relative probability calculated according to the laws of optics.

Furthermore, optical photons can be absorbed in flight (if the user defines a non zero absorption coefficient for the material under consideration) or elastically scattered (Rayleigh scattering) if the user defines a non zero diffusion coefficient for the material under consideration).

In order to deal with optical photon problems, two specific input commands are available to the user:

1) OPT–PROP: to set optical properties of materials.
2) OPT–PROD: to manage light generation.

See p. 160 and 164 for a detailed description of these options and of their parameters.
Some user routines are also available for a more complete representation of the physical problem:

i) RFRNDX: to specify a refraction index as a function of wavelength, frequency or energy
ii) RFLCTV: to specify the reflectivity of a material. This can be activated by card OPT–PROP with SDUM = METAL and WHAT(3) < -99.
iii) OPHBDX: to set optical properties of a boundary surface. The call is activated by card OPT–PROP with SDUM = SPEC–BDX.
iv) FRGHNS: to set a possible degree of surface roughness, in order to have both diffusive and specular reflectivity from a given material surface (**not yet implemented**).
v) QUEFFC: to request a detailed quantum efficiency treatment. This is activated by card OPT–PROP with SDUM = SENSITIV, setting the 0^{th} optical photon sensitivity parameter to a value lesser than -99 (WHAT(1) < -99).

All running values of optical photon tracking are contained in COMMON TRACKR, just as for the other ordinary elementary particles (see 12.1.1, 12.2.13).

13.1 Cherenkov transport and quantum efficiency

In order to use quantum efficiency (via of the QUEFFC routine) the user must input a sensitivity < -100 by means of the OPT–PROP option with SDUM = SENSITIV).

That option sets the quantum efficiency as a function of photon energy *overall* through the problem and it is not material/region dependent. The reason is that it is applied "a priori" at photon generation time (for obvious time saving reasons). Here below is an explanation taken directly from the code.

a) current particle is at a given position, in a given material
b) Cherenkov (or scintillation) photons are going to be produced
c) the photon generation probability is immediately reduced over the whole energy spectrum according to the maximum quantum efficiency of the problem. The latter, set as WHAT(5) in card OPT–PROD (p. 160), *is meaningful even if routine* QUEFFC *is used*, see below
d) the detailed efficiency, set again by the OPT–PROD command with SDUM = SENSITIV or by routine QUEFFC, is used for a further reduction when the actual energy of each individual photon is selected (rejecting it against Q.E.(ω)/Q.E.$_{max}$, where ω = photon angular frequency)

Summarising, the yes/no detection check is done *at production* and *not at detection*: this in order to substantially cut down CPU time. If one wants all photons to be produced the sensitivity must be set = 1. Then it is still possible to apply a quantum efficiency curve *at detection*, by means of the user weighting routine FLUSCW (see 12.2.6) or by a user-written off-line code.

Since the quantum efficiency curve provided by OPT–PROD with SDUM = SENSITIV is applied at production and not at detection, it is not known which material the photon will eventually end up in.

Furthermore, WHAT(5) must be set anyway equal to the maximum quantum efficiency over the photon energy range under consideration. One cannot use the QUEFFC routine as a way to provide an initial screening on the produced photons, i.e., to use a "safe" initial guess for the quantum efficiency (say, for instance 20%) and then, at detection, refine it through more sophisticated curves, i.e., rejecting against the actual quantum efficiency/0.2 (this again can be done in routine FLUSCW). This makes sense of course if the user has different quantum efficiency curves for different detectors (one should use in QUEFFC the curve that maximises all of them and then refine it by rejection case by case), or if the quantum efficiency is position/angle dependent upon arrival on the photomultiplier (again one should use inside QUEFFC the quantum efficiency for the most efficient position/angle and refine by rejection at detection time.

Optical photons are absorbed in those materials where the user selected properties dictate absorption, i.e., metals or materials with a non zero absorption cross-section. These absorption events can be detected in different ways. For instance:

a) through energy deposition by particle -1 (optical photons have always id = -1), photons usually deposit all energy in one step (since only absorption and coherent scattering are implemented). So one can check for JTRACK = -1 and energy deposition (RULL) in a given region (e.g., the photo-cathode of the PMT). One can also apply an extra quantum efficiency selection, e.g., using the COMSCW user routine.
b) through boundary crossing of particles -1 into the given region, however this is correct if and only if absorption is set such that the photon will not survive crossing the region. Again further selections can be performed, e.g., using the FLUSCW user routine.

Several practical examples of handling optical photons are presented in 13.2.

13.2 Handling optical photons

In order to help the user to understand how to deal with optical photons, in the following we describe two input files respectively concerning the production of Cherenkov and Scintillation light in Liquid Argon. A specific user routine, giving the refraction index of Liquid Argon as a function of wavelength is also shown.

It is a very simple case, in which muons are generated inside a box filled with Liquid Argon. Notice that at present it is not yet possible to request optical photons as primary particles via the BEAM card. Therefore light must be generated by a user-written SOURCE routine, or starting from ordinary particles.

The examples presented here consider 0.5 GeV muons in a box of $4 \times 4 \times 4$ m^3. In order to avoid unnecessary complications in the example, secondary particle production by muons is switched off. Of course this is not required in real problems.

As far as the output is concerned, the following example proposes a standard energy spectrum scoring at a boundary (option USRBDX) applied to optical photons, together with a user-specific output built via the MGDRAW user routine (see 12.2.13), where a dump of optical photon tracking is inserted. At the end of this section we will propose the relevant code lines to be inserted in MGDRAW (activated by the USERDUMP card, p. 208), together with an example of readout.

13.2.1 Routine assigning a continuous Refraction Index as a function of Wavelength

Notice that in this example a check is performed on the material number. In the following problems, the light will be generated on material no. 18. In order to avoid problems a FLUKA abort is generated if the routine is called by mistake for a different material.

```
*
*=== Rfrndx ===========================================================*
*
      DOUBLE PRECISION FUNCTION RFRNDX ( WVLNGT, OMGPHO, MMAT )

      INCLUDE '(DBLPRC)'
      INCLUDE '(DIMPAR)'
      INCLUDE '(IOUNIT)'
*
*----------------------------------------------------------------------*
*                                                                      *
*     user defined ReFRaction iNDeX:                                   *
*                                                                      *
*     Created on 19 September 1998 by    Alfredo Ferrari & Paola Sala  *
*                                                Infn - Milan          *
*                                                                      *
*     Last change on 25-Oct-02     by    Alfredo Ferrari               *
*                                                                      *
*                                                                      *
*----------------------------------------------------------------------*
*
      INCLUDE '(FLKMAT)'
*
* Check on the material number
*
      IF ( MMAT .NE. 18 ) THEN
         CALL FLABRT ( 'RFRNDX', 'MMAT IS NOT SCINTILLATOR!' )
      END IF
*
      WL     =  WVLNGT * 1.D+07
      RFRNDX = ONEONE
     &       + 9.39373D+00*(4.15D-08/(0.000087892D+00 - WL**(-2))
     &       + 2.075D-07 / (0.000091012D+00 - WL**(-2))
     &       + 4.333D-06 / (0.00021402 D+00 - WL**(-2)))
      RETURN
*=== End of function Rfrndx ===========================================*
      END
```

13.2.2 Input Example no. 1: Only Cherenkov light is generated

Cherenkov light generation depends on the refraction index. Among the different possibilities, here we have chosen to give the refraction index by means of the user routine shown above. The relevant data cards are commented. The value inserted for light absorption in this example is arbitrary, while the mean free path for Rayleigh scattering is the result obtained from measurements performed in the framework of the ICARUS collaboration.

```
TITLE
Test of Cherenkov light production in Liquid Argon
DEFAULTS                                                              PRECISIO
*...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+
BEAM         -10.000                                                  MUON+
BEAMPOS          0.0       0.0     190.0                              NEGATIVE
DELTARAY        -1.0                            18.0      18.0
PAIRBREM        -3.0                            18.0      18.0
MUPHOTON        -1.0                            18.0      18.0
PHOTONUC        -1.0                             3.0     100.0
EVENTYPE                               6.0
DISCARD         27.0      28.0      43.0      44.0       5.0       6.0
GEOBEGIN                                                              COMBINAT
                              Test
*...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+
* A large box for the blackhole
  RPP    1 -9999999. +9999999. -9999999. +9999999. -9999999. +9999999.
* A smaller box for for liquid argon
  RPP    2    -200.0    +200.0    -200.0    +200.0    -200.0    +200.0
  END
*== Region Definitions ===============================================
*      1) Blackhole
  BL1           +1     -2
*      2) Liquid Argon
  LG3           +2
  END
GEOEND
*  Switch off electron and photon transport
EMF                                                                   EMF-OFF
*
MATERIAL        18.0    39.948     1.400      18.0                    ARGON
*  Select neutron cross-sections at liquid argon temperature
LOW-MAT         18.0      18.0      -2.0      87.0                    ARGON
*
ASSIGNMAT        1.0       1.0      500.       1.0       0.0
ASSIGNMAT       18.0       2.0       2.0
*
* Set Light production/transport properties: from 100 to 600 nm in all materials
OPT-PROP   1.000E-05 3.500E-05 6.000E-05       3.0     100.0          WV-LIMIT
* Set all materials to "metal" with 0 reflectivity:
OPT-PROP                               1.0       3.0     100.0          METAL
* resets all previous properties for material n. 18 (Liquid Argon)
OPT-PROP                                        18.0                    RESET
* switches off scintillation light production in material n. 18 (Liq. Argon)
OPT-PROD                                        18.0                    SCIN-OFF
* defines Cherenkov production for material n. 18 (Liq. Argon)
OPT-PROD   1.100E-05 5.500E-05                  18.0                    CEREN-WV
* The following card restores the wave-length limits for material n. 18
OPT-PROP   1.000E-05 3.500E-05 6.000E-05        18.0                    WV-LIMIT
* The following card, for material n. 18:
* a) calls the RFRNDX user routine (to define the refraction index
*    vs wave-length (WHAT(1)< -99)
* b) sets to 1000 cm the mean free path for absorption.
* c) sets to 90 cm the mean free path for Rayleigh scattering.
OPT-PROP      -100.0     0.001   0.01111      18.0
* The following card defines the "Sensitivity" in order to introduce the
* maximum Quantum Efficiency at generation level.
* Here 1/10 of photons is actually generated.
* Fluctuations are properly sampled
OPT-PROP         0.1                                    0.1          SENSITIV
SCORE          208.0     211.0     201.0     210.0
RANDOMIZ         1.0
```

```
*...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+
USRBDX          1.0      -1.0     -55.0       2.0       1.0          Opt.Phot
USRBDX      12.0E-09       0.0     120.0                               &
USERDUMP         111.                 2.                            MGDRAW
START          10000.0
STOP
```

13.2.3 Input Example no. 2: Only Scintillation light is concerned

Here it is necessary to point out that, at present, FLUKA can generate scintillation lines only for monochromatic lines. A maximum number of 3 different lines is possible. The value inserted here (128 nm) is the correct one for Liquid Argon. The fraction of deposited energy going into scintillation light depends on the degree of recombination after ionisation. Again, the value used here is a parameter justified in the framework of the ICARUS collaboration, where about 20000 photons/MeV of deposited energy have been measured for the electric field of 500 V/cm (the field used in ICARUS). A different electric field intensity will change the degree of recombination and therefore the light yield.

```
TITLE
Test of scintillation light production in Liquid Argon
DEFAULTS                                                              PRECISIO
BEAM      -0.5000                                                     MUON+
BEAMPOS    0.0        0.0      199.0                                  NEGATIVE
*
DELTARAY  -1.0                          18.0      18.0
PAIRBREM  -3.0                          18.0      18.0
MUPHOTON  -1.0                          18.0      18.0
PHOTONUC  -1.0                           3.0      100.0
EVENTYPE                        6.0
DISCARD    27.0      28.0     43.0      44.0       5.0       6.0
GEOBEGIN                                                              COMBINAT
                                Test
*...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+
* A large box for the blackhole
  RPP    1 -9999999. +9999999. -9999999. +9999999. -9999999. +9999999.
* A SMALLER BOX FOR FOR LIQUID ARGON
  RPP    2    -200.0    +200.0    -200.0    +200.0    -200.0    +200.0
  END
*== Region Definitions ================================================
*      1) Blackhole
  BL1          +1     -2
*      2) Liquid Argon
  LG3          +2
  END
GEOEND
*
EMF                                                                   EMF-OFF
*
MATERIAL        18.0    39.948     1.400      18.0                    ARGON
LOW-MAT         18.0      18.0      -2.0      87.0                    ARGON
ASSIGNMAT        1.0       1.0       500.      1.0       0.0
ASSIGNMAT       18.0       2.0       2.0
*
* Set Light production/transport properties: from 100 to 600 nm in all materials
OPT-PROP   1.000E-05 1.280E-05 6.000E-05       3.0     100.0          WV-LIMIT
* Set all materials to "metal" with 0 reflectivity:
OPT-PROP                              1.0       3.0     100.0          METAL
* resets all previous properties for material n. 18 (Liquid Argon)
OPT-PROP                                       18.0                    RESET
* switches off Cherenkov light production in material n. 18 (Liquid Argon)
OPT-PROD                                       18.0                    CERE-OFF
* defines Scint. light production for material n. 18 (Liq. Argon). Parameters:
```

```
* a) wavelength (cm) of first scintillation line.
* b) fraction of deposited energy going into scint. light
*    (in Liquid Argon ~ 2 10**4 photons/MeV)
OPT-PROD   1.280E-05 9.686E-02               18.0                    SCINT-WV
* The following card restores the wave-length limits for material n. 18
OPT-PROP   1.000E-05 1.280E-05 6.000E-05     18.0                    WV-LIMIT
* The following card, for material n. 18:
* a) calls the RFRNDX user routine (to define the refraction index
*    vs wave-length (WHAT(1)< -99)
* b) sets to 1000 cm the mean free path for absorption.
* c) sets to 90 cm the mean free path for Rayleigh scattering
OPT-PROP      -100.0     0.001   0.01111      18.0
* The following card defines the "Sensitivity" in order to introduce the
* maximum Quantum Efficiency at generation level. Here 1/10 of photons are
* actually generated.
* Fluctuations are properly sampled
*...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+
OPT-PROP         0.1                                     0.1          SENSITIV
SCORE          208.0     211.0     201.0     210.0
RANDOMIZ         1.0
*
USRBDX           1.0      -1.0     -55.0       2.0       1.0          Opt.Phot
USRBDX      12.0E-09       0.0     120.0                                &
USERDUMP        111.                 2.                               MGDRAW
START         10000.
STOP
```

13.2.4 User output for optical photons from USERDUMP

The user can request any kind of standard FLUKA output for optical photons and also a user specific output, starting for instance from the MGDRAW user routine. Here an example follows, where a few variables are simply recorded in the output "collision tape" (dump file) at each step in the tracking only for particle id = -1 (optical photons).

It can be useful, in order to exploit the flags available in this routine, and explained in the different comments, to know that the FLUKA routine which drives the transport of optical photons is KASOPH.

The content of COMMON TRACKR can be used to take full advantage of the possibilities offered by the MGDRAW routine (see 12.2.13).

Warning: in the present version of FLUKA, there is not yet the possibility of using the User Particle Properties for optical photons (variables SPAUSR, ISPUSR and the STUPFR user routine)

```
      SUBROUTINE MGDRAW ( ICODE, MREG )

      INCLUDE '(DBLPRC)'
      INCLUDE '(DIMPAR)'
      INCLUDE '(IOUNIT)'
*
*----------------------------------------------------------------------*
*                                                                      *
*     MaGnetic field trajectory DRAWing: actually this entry manages   *
*                                        all trajectory dumping for    *
*                                        drawing                       *
*                                                                      *
*     Created by                     Alfredo Ferrari                   *
*                                    INFN - Milan                      *
*     last change  10-Nov-02  by     Alfredo Ferrari                   *
*                                    INFN - Milan                      *
*                                                                      *
*----------------------------------------------------------------------*
```

```
*
      INCLUDE '(CASLIM)'
      INCLUDE '(COMPUT)'
      INCLUDE '(FHEAVY)'
      INCLUDE '(FLKSTK)'
      INCLUDE '(GENSTK)'
      INCLUDE '(MGDDCM)'
      INCLUDE '(PAPROP)'
      INCLUDE '(SOURCM)'
      INCLUDE '(STARS)'
      INCLUDE '(TRACKR)'
*
      CHARACTER*20 FILNAM
      LOGICAL LFCOPE
      SAVE LFCOPE
      DATA LFCOPE / .FALSE. /
*
*=====================================================================*
*                                                                     *
*     Icode = 1: call from Kaskad                                     *
*     Icode = 2: call from Emfsco                                     *
*     Icode = 3: call from Kasneu                                     *
*     Icode = 4: call from Kashea                                     *
*     Icode = 5: call from Kasoph                                     *
*                                                                     *
*=====================================================================*
*                                                                     *
      IF ( .NOT. LFCOPE ) THEN
         LFCOPE = .TRUE.
         IF ( KOMPUT .EQ. 2 ) THEN
            FILNAM = '/'//CFDRAW(1:8)//' DUMP A'
         ELSE
            FILNAM = CFDRAW
         END IF
         WRITE(*,*) 'TRAJECTORY OPEN!'
         WRITE(*,'(A)') 'FILNAM = ',FILNAM
         OPEN ( UNIT = IODRAW, FILE = FILNAM, STATUS = 'NEW', FORM =
     &          'UNFORMATTED' )
      END IF
C
C     Write trajectories of optical photons
C
      IF(JTRACK .EQ. -1) THEN
         WRITE (IODRAW) NTRACK, MTRACK, JTRACK, SNGL (ETRACK),
     &        SNGL (WTRACK)
         WRITE (IODRAW) ( SNGL (XTRACK (I)), SNGL (YTRACK (I)),
     &        SNGL (ZTRACK (I)), I = 0, NTRACK ),
     &        ( SNGL (DTRACK (I)), I = 1,MTRACK ),
     &        SNGL (CTRACK)
         WRITE(IODRAW) SNGL(CXTRCK),SNGL(CYTRCK),SNGL(CZTRCK)
      ENDIF
      RETURN
*
*=====================================================================*
*                                                                     *
*     Boundary-(X)crossing DRAWing:                                   *
*                                                                     *
*     Icode = 1x: call from Kaskad                                    *
*             19: boundary crossing                                   *
*     Icode = 2x: call from Emfsco                                    *
*             29: boundary crossing                                   *
*     Icode = 3x: call from Kasneu                                    *
```

```
*             39: boundary crossing                                    *
*     Icode = 4x: call from Kashea                                     *
*             49: boundary crossing                                    *
*     Icode = 5x: call from Kasoph                                     *
*             59: boundary crossing                                    *
*                                                                      *
*======================================================================*
*                                                                      *
      ENTRY BXDRAW ( ICODE, MREG, NEWREG, XSCO, YSCO, ZSCO )
      RETURN
*
*======================================================================*
*                                                                      *
*     Event End DRAWing:                                               *
*                                                                      *
*======================================================================*
*                                                                      *
      ENTRY EEDRAW ( ICODE )
      RETURN
*
*======================================================================*
*                                                                      *
*     ENergy deposition DRAWing:                                       *
*                                                                      *
*     Icode = 1x: call from Kaskad                                     *
*             10: elastic interaction recoil                           *
*             11: inelastic interaction recoil                         *
*             12: stopping particle                                    *
*             13: pseudo-neutron deposition                            *
*             14: escape                                               *
*             15: time kill                                            *
*     Icode = 2x: call from Emfsco                                     *
*             20: local energy deposition (i.e. photoelectric)         *
*             21: below threshold, iarg=1                              *
*             22: below threshold, iarg=2                              *
*             23: escape                                               *
*             24: time kill                                            *
*     Icode = 3x: call from Kasneu                                     *
*             30: target recoil                                        *
*             31: below threshold                                      *
*             32: escape                                               *
*             33: time kill                                            *
*     Icode = 4x: call from Kashea                                     *
*             40: escape                                               *
*             41: time kill                                            *
*     Icode = 5x: call from Kasoph                                     *
*             50: optical photon absorption                            *
*             51: escape                                               *
*             52: time kill                                            *
*                                                                      *
*======================================================================*
*                                                                      *
      ENTRY ENDRAW ( ICODE, MREG, RULL, XSCO, YSCO, ZSCO )
      RETURN
*
*======================================================================*
*                                                                      *
*     SOurce particle DRAWing:                                         *
*                                                                      *
*======================================================================*
*
      ENTRY SODRAW
```

```
*  |
*  +--------------------------------------------------------------------*
      RETURN
*
*====================================================================*
*                                                                    *
*     USer dependent DRAWing:                                        *
*                                                                    *
*     Icode = 10x: call from Kaskad                                  *
*             100: elastic   interaction secondaries                 *
*             101: inelastic interaction secondaries                 *
*             102: particle decay  secondaries                       *
*             103: delta ray  generation secondaries                 *
*             104: pair production secondaries                       *
*             105: bremsstrahlung  secondaries                       *
*     Icode = 20x: call from Emfsco                                  *
*             208: bremsstrahlung secondaries                        *
*             210: Moller secondaries                                *
*             212: Bhabha secondaries                                *
*             214: in-flight annihilation secondaries                *
*             215: annihilation at rest   secondaries                *
*             217: pair production        secondaries                *
*             219: Compton scattering     secondaries                *
*             221: photoelectric          secondaries                *
*             225: Rayleigh scattering    secondaries                *
*     Icode = 30x: call from Kasneu                                  *
*             300: interaction secondaries                           *
*     Icode = 40x: call from Kashea                                  *
*             400: delta ray  generation secondaries                 *
*  For all interactions secondaries are put on GENSTK common (kp=1,np) *
*  but for KASHEA delta ray generation where only the secondary elec-  *
*  tron is present and stacked on FLKSTK common for kp=lstack        *
*                                                                    *
*====================================================================*
*
      ENTRY USDRAW ( ICODE, MREG, XSCO, YSCO, ZSCO )
* No output by default:
      RETURN
*=== End of subroutine Mgdraw ========================================*
      END
```

13.2.5 Readout of the sample user output

A sample program to readout the output obtained from the previously shown `MGDRAW` routine is presented here below. In this example the key routine in the one called `VXREAD`, where some trivial output is sent to logical units 66 and 67. Of course the user must adapt such a readout program to his own needs.

```
      PROGRAM MGREAD
      CHARACTER FILE*80
*
      WRITE (*,*)' Name of the binary file?'
      READ  (*,'(A)') FILE
      OPEN ( UNIT = 33, FILE = FILE, STATUS ='OLD',
     &       FORM = 'UNFORMATTED' )
 1000 CONTINUE
         WRITE (*,*)' Event number?'
         READ  (*,*) NCASE
         IF ( NCASE .LE. 0 ) STOP
         CALL VXREAD (NCASE)
      GO TO 1000
      END
```

```
      SUBROUTINE VXREAD (NCASE)
      PARAMETER (  MXH = 2000 )
      PARAMETER ( MXPR =  300 )
      DIMENSION XH   (MXH), YH   (MXH), ZH   (MXH), DH   (MXH),
     &          EPR (MXPR), WPR (MXPR), XPR (MXPR), YPR (MXPR),
     &          ZPR (MXPR), TXP (MXPR), TYP (MXPR), TZP (MXPR),
     &          IPR (MXPR)
*
      LUNSCR = 33
      REWIND (LUNSCR)
*
*  +-----------------------------------------------------------------*
*  |
      NEVT=0
      DO 4000 I=1,2000000000
         READ (LUNSCR,END=4100) NDUM,MDUM,JDUM,EDUM,WDUM
         IF(I.EQ.1) WRITE(*,*) 'NDUM,MDUM,JDUM,EDUM,WDUM',NDUM,MDUM,JDUM
     &        ,EDUM,WDUM
         NEVT = NEVT + 1
*  |  +--------------------------------------------------------------*
*  |  |  Real tracking data:
*  |  +--------------------------------------------------------------*
         IF ( NDUM .GT. 0 ) THEN
            NTRACK=NDUM
            MTRACK=MDUM
            JTRACK=JDUM
            ETRACK=EDUM
            WTRACK=WDUM
            IF(NTRACK.GT.1) WRITE(67,*) 'NTRACK = ',NTRACK
            READ (LUNSCR)(XH(J),YH(J),ZH(J),J=1,NTRACK+1),
     &                    (DH(J),J=1,MTRACK), CTRACK
            READ (LUNSCR) CXX,CYY,CZZ
            IF(I.EQ.1) THEN
               WRITE(67,*) (XH(J),YH(J),ZH(J),J=1,NTRACK+1),
     &                     (DH(J),J=1,MTRACK), CTRACK
               WRITE(67,*) CXX,CYY,CZZ
            ENDIF
            DO J=1,NTRACK+1
               WRITE(67,*) XH(J),YH(J),ZH(J),
     &              CXX,CYY,CZZ
            END DO
            IF ( NEVT.EQ.NCASE ) THEN
               WRITE(66,*)' New step:'
               WRITE(66,*)'   Part.id.:',JTRACK,' Kin.En.:',ETRACK,
     &                    '   N.of substep:', NTRACK
               WRITE(66,*)'   X, Y, Z, i=0, # substep'
               WRITE(*,*)' New step:'
               WRITE(*,*)'   Part.id.:',JTRACK,' Kin.En.:',ETRACK,
     &                    '   N.of substep:', NTRACK
               WRITE(*,*)'   X, Y, Z, i=0, # substep'
            END IF
*  |  |
*  |  +--------------------------------------------------------------*
*  |  |  Energy deposition data:
         ELSE IF ( NDUM .EQ. 0 ) THEN
            ICODE1=MDUM/10
            ICODE2=MDUM-ICODE1*10
            IJDEPO=JDUM
            ENPART=EDUM
            WDEPOS=WDUM
            READ (LUNSCR) XSCO, YSCO, ZSCO, ENDEPO
            IF ( NEVT.EQ.NCASE ) THEN
```

```
               WRITE(66,*) ' En. dep. code n.:',MDUM
               WRITE(66,*) IJDEPO,'   Tot. en. proj.:', ENPART,
     &                     ' Weight:',WDEPOS
               WRITE(66,*) '   Position:',XSCO,YSCO,ZSCO,
     &                     ' En. Dep.:',ENDEPO
            END IF
*  |  |
*  |  +-------------------------------------------------------------*
*  |  |  Source particle:
         ELSE
            NEVT   =-NDUM
            LPRIMA = MDUM
            NSTMAX = JDUM
            TKESUM = EDUM
            WEIPRI = WDUM
            READ (LUNSCR) ( IPR(J),EPR(J),WPR(J),XPR(J),YPR(J),
     &                      ZPR(J),TXP(J),TYP(J),TZP(J),J=1,LPRIMA )
            DO J = 1, LPRIMA
               IF ( ABS(IPR(J)) .LT. 10000 ) THEN
                  LPTRUE=J
               END IF
            END DO
            LPROJ  = LPRIMA - LPTRUE
            LPRIMA = LPTRUE
            IF (NEVT .EQ. NCASE) THEN
               WRITE(66,*)' Event #',NEVT
               IF ( LPROJ .GT. 0) THEN
                  WRITE(66,*)
     &            '  Original projectile(s),n. of:',LPROJ
                  DO IJ = 1, LPROJ
                     J=LPRIMA+IJ
                     IPR(J) = IPR(J)/10000
                     WRITE(66,*) '    Part.id.:',IPR(J),' Kin.en.:',
     &                                EPR(J),' Weight:',WPR(J)
                     WRITE(66,*) IPR(J),EPR(J),WPR(J)
                     WRITE(66,*) '    Position :', XPR(J),YPR(J),ZPR(J)
                     WRITE(66,*) '    Direction:', TXP(J),TYP(J),TZP(J)
                  END DO
               END IF
               WRITE(66,*)'  Source particle(s), n. of:',LPRIMA
               DO J = 1, LPRIMA
                  WRITE(66,*) '    Part.id.:',IPR(J),' Kin.en.:',
     &                           EPR(J),' Weight:',WPR(J)
                  WRITE(66,*) '    Position :', XPR(J),YPR(J),ZPR(J)
                  WRITE(66,*) '    Direction:', TXP(J),TYP(J),TZP(J)
C                  WRITE(67,*) XPR(J)/1.E+05,YPR(J)/1.E+05,ZPR(J)/1.E+05
               END DO
            END IF
            IF (NEVT.GT.NCASE) GO TO 4100
         END IF
*  |  |
*  |  +-------------------------------------------------------------*
 4000 CONTINUE
*  |
*  +-------------------------------------------------------------------*
 4100 CONTINUE
      RETURN
      END
```

Chapter 14

Use of RAY pseudo-particles

Pseudo-particles are called RAY and have particle number 0. As for real particles, their energy, starting point and direction cosines can be defined by options BEAM (p. 64) and BEAMPOS (p. 69), or by a user-written SOURCE routine (see option SOURCE, p. 197).

A RAY travels in a straight line at the speed of light without any physical interaction. At the starting point, and at each boundary crossing, FLUKA writes information on a binary (unformatted) file. The file logical unit number is parameterised as LUNRAY in INCLUDE file (IOUNIT) (usually LUNRAY = 10).

An example of user program which can be used to retrieve the tracking information from the binary file is given below. The meaning of the variables read is explained at the end:

```
*     ..............................................................
*     user program or subroutine
*     ..............................................................
      PARAMETER (LUNRAY = 10)
      CHARACTER MATNAM*8, FILNAM*80
      INTEGER NRAYRN, MREG, MLATTC, MMAT, MREGLD, MLATLD, MMATLD, IDISC
      REAL EKIN, XX, YY, ZZ, R2, R3, THETAP, PHIPOS, TXX, TYY, TZZ,
     &     THETAD, PHIDIR, ETADIR, RCM, ALAMDI, ALAMDP, ALAMDN, ALAMDG,
     &     ALAMDR, DEKMIP, GMOCM2, DELAGE, RCMTOT, ALITOT, ALPTOT,
     &     ALNTOT, ALGTOT, ALRTOT, TOTMIP, SRHTOT, AGEAGE
*     ..............................................................
*     here other possible declarations
*     ..............................................................
      WRITE (*,*) ' File name?'
      READ  (*,'(A80)') FILNAM
      OPEN (FILE = FILNAM, UNIT = LUNRAY, STATUS = 'OLD', FORM =
     &      'UNFORMATTED')
*     loop over several rays
    1 CONTINUE
*     read info about ray starting point
        READ (LUNRAY, END = 1000) NRAYRN, MREG, MLATTC, MMAT, EKIN
        READ (LUNRAY, END = 1000) XX, YY, ZZ, R2, R3, THETAP, PHIPOS
        READ (LUNRAY, END = 1000) TXX, TYY, TZZ, THETAD, PHIDIR, ETADIR
*       ..............................................................
*       here possible user code to manipulate values read
*       ..............................................................
*       loop over further positions along the ray path
    2   CONTINUE
*         read info about next point
          READ (LUNRAY, END = 3000) MREGLD, MLATLD, MMATLD,
     &                              MATNAM, IDISC
          READ (LUNRAY, END = 2000) XX, YY, ZZ, R2, R3, THETAP, PHIPOS
          READ (LUNRAY, END = 2000) RCM, ALAMDI, ALAMDP, ALAMDN, ALAMDG,
     &                              ALAMDR, DEKMIP, GMOCM2, DELAGE
          READ (LUNRAY, END = 2000) RCMTOT, ALITOT, ALPTOT, ALNTOT,
     &                              ALGTOT, ALRTOT, TOTMIP, SRHTOT,
     &                              AGEAGE
*         ..............................................................
*         possible user code to manipulate values read
*         ..............................................................
          IF ( IDISC .EQ. 0 ) THEN
*           ..........................................................
*           possible user code at the end of ray step
*           ..........................................................
```

```
            GO TO 2
          END IF
*       ..............................................................
*       possible user code at the end of ray trajectory
*       ..............................................................
*       new ray
      GO TO 1
 1000 CONTINUE
      WRITE(*,*) ' Incomplete data on file about ray starting point'
      GO TO 3000
 2000 CONTINUE
      WRITE(*,*) ' Incomplete data on file about ray trajectory'
 3000 CONTINUE
*     ..............................................................
*     possible user code at the end of analysis
*     ..............................................................
      CLOSE (UNIT = LUNRAY)
      END
```

Meaning of the variables read from the RAY file:

NRAYRN = ray number
MREG = starting region
MLATTC = starting lattice cell
MMAT = material of starting region
EKIN = reference kinetic energy of the ray (GeV)
XX, YY, ZZ = current ray position (the first time: ray starting point)
R2 = distance of current ray position from z-axis (cm)
R3 = distance of current ray position from the origin (cm)
THETAP = polar angle between the current ray position vector and the z-axis (radians)
PHIPOS = azimuthal angle of the current ray position vector around the z-axis (radians)
TXX, TYY, TZZ = ray direction cosines
THETAD = polar angle between the ray and the z-axis (radians)
PHIDIR = azimuthal angle of ray around the z-axis (radians)
ETADIR = pseudo-rapidity of ray direction with respect to the direction defined by option BEAMPOS
MREGLD = number of next region traversed by ray
MLATLD = number of next lattice cell traversed by ray
MMATLD = material of next region
MATNAM = name of material of next region
IDISC = 0 unless next region is blackhole
RCM = distance traversed from last point to the current ray position
ALAMDI = distance traversed from last point to the current ray position, in units of high energy nucleon inelastic mean free paths (at the reference kinetic energy of the ray)
ALAMDP = distance traversed from last point to the current ray position, in units of high energy pion inelastic mean free paths (at the reference kinetic energy of the ray)
ALAMDN = distance traversed from last point to the current ray position, in units of maximum neutron inelastic mean free paths (i.e., at 200 MeV)
ALAMDG = distance traversed from last point to the current ray position, in units of maximum photon mean free paths (i.e., at the so-called Compton minimum). Note: if the EMF option (p. 97) has not been requested, explicitly or implicitly by default, ALAMDG has always zero value
ALAMDR = distance traversed from last point to the current ray position, in units of radiation lengths. Note: if the EMF option has not been requested, explicitly or implicitly by default, ALAMDR is calculated but only in an approximate way
DEKMIP = energy lost from last point to the current ray position by a minimum ionising muon
GMOCM2 = distance traversed from last point to the current ray position in g/cm^2
DELAGE = time elapsed from last point to the current ray position in sec (i.e., distance divided by speed of light)
RCMTOT = cumulative distance traversed so far in cm

ALITOT = cumulative distance traversed so far, in units of high energy nucleon inelastic mean free paths
ALPTOT = cumulative distance traversed so far, in units of high energy pion inelastic mean free paths
ALNTOT = cumulative distance traversed so far, in units of maximum neutron inelastic mean free paths
ALGTOT = cumulative distance traversed so far, in units of maximum photon mean free paths (i.e., at the so-called Compton minimum). Note: if the EMF option has not been requested, explicitly or implicitly by default, ALGTOT has always zero value
ALRTOT = cumulative distance traversed so far, in units of radiation lengths. Note: if the EMF option has not been requested, explicitly or implicitly by default, ALRTOT is calculated but only in an approximate way
TOTMIP = cumulative energy lost so far by a minimum ionising muon
SRHTOT = cumulative distance traversed so far in g/cm^2
AGEAGE = cumulative time elapsed so far in seconds

Chapter 15

Examples on the material/compound definitions

15.1 Use of MATERIAL, LOW-MAT and COMPOUND

```
*  First define the elements used:

*...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....
*  Single element, single isotope CH2-bound Hydrogen is material n. 3, coupled
*  to low-energy neutron cross-sections at normal temperature
MATERIAL         1.0    1.0079       1.0       3.0               1.0 HYDROGEN
* low-mat is necessary (the default correspondence is with H2O-bound hydrogen)
LOW-MAT          3.0       1.0      -3.0     293.0                   HYDROGEN
*
*  Single element, natural isotopic composition Carbon (mat. n. 6),
*  coupled to low-energy neutron cross-sections at normal temperature
MATERIAL         6.0     12.01      2.25       6.0                   CARBON
* actually the following is not necessary (the same correspondence is default)
LOW-MAT          6.0       6.0      -2.0      293.                   CARBON
*
*  Single element, natural isotopic composition Nitrogen and Oxygen
MATERIAL         7.0    14.007  0.001251       7.0                   NITROGEN
* actually the following is not necessary (the same correspondence is default)
LOW-MAT          7.0       7.0      -2.0      293.                   NITROGEN
MATERIAL         8.0    15.999  0.001429       8.0                   OXYGEN
* actually the following is not necessary (the same correspondence is default)
LOW-MAT          8.0       8.0      16.0      293.                   OXYGEN
*
*...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....
*
*  Three Aluminiums:
*   Aluminium material n. 9, normal density, coupled with low-energy
*   neutron cross-sections at 87 K (overrides default mat. 9)
MATERIAL        13.0    26.982      2.70       9.0                   ALUMINUM
* actually the following is not necessary (the same correspondence is default)
LOW-MAT          9.0      13.0      -2.0      87.0                   ALUMINUM
*   Aluminium material n. 10, low density, coupled with low-energy
*   neutron cross-sections at normal temperature
MATERIAL        13.0    26.982      1.35      10.0                   ALUMLOW
* the following is necessary (names are different)
LOW-MAT         10.0      13.0      -2.0     293.0                   ALUMINUM
*   Aluminium material n. 11, normal density, normal temperature
*   neutron cross-sections at 87 K (overrides pre-defined mat. 11)
MATERIAL        13.0    26.982      2.70      11.0                   ALUMCOLD
* the following is necessary (names are different)
LOW-MAT         11.0      13.0      -2.0     293.0                   ALUMINUM
*
*...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....
*
*  Two Irons, overriding default materials 12 and 13
*   Iron material n. 12, at 87 K
MATERIAL        26.0     55.85      7.87      12.0                   IRON-87
* the following is necessary (names are different)
LOW-MAT         12.0      26.0      -2.0      87.0                   IRON
*   Iron material n. 13, at normal temperature
MATERIAL        26.0     55.85      7.87      13.0                   IRON
* actually the following is not necessary (the same correspondence is default)
LOW-MAT         13.0      26.0      -2.0     293.0                   IRON
*  Two Leads:
```

```
*   Lead material n. 16, normal temperature, overrides pre-defined mat. 16
MATERIAL        82.0    207.19     11.35      16.0                     LEAD
* actually the following is not necessary (the same correspondence is default)
LOW-MAT         16.0      82.0      -2.0     293.0                     LEAD
*   Lead material n. 17, at 87 K
MATERIAL        82.0    207.19     11.35      17.0                     LEAD-87
* the following is necessary (names are different)
LOW-MAT         17.0      82.0      -2.0      87.0                     LEAD
*
*...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....
*
*  Two Argons:
*   Argon material n. 18, normal temperature, overrides pre-defined mat. 18
MATERIAL        18.0    39.948     1.400      18.0                     ARGON
* actually the following is not necessary (the same correspondence is default)
LOW-MAT         18.0      18.0      -2.0     293.0                     ARGON
*   Argon material n. 27, at 87 K
MATERIAL        18.0    39.948     1.400      27.0                     LIQARGON
* the following is necessary (names are different)
LOW-MAT         27.0      18.0      -2.0      87.0                     ARGON
*   Calcium material n. 23, normal temp. overrides pre-defined mat. 23
MATERIAL        20.0     40.08      1.54      23.0                     CALCIUM
* actually the following is not necessary (the same correspondence is default)
LOW-MAT         23.0      20.0      -2.0     293.0                     CALCIUM
*   Silicon material n. 26, normal temperature
MATERIAL        14.0    28.086    0.5825      26.0                     SILICON
* actually the following is not necessary (the same correspondence is default)
LOW-MAT         26.0      14.0      -2.0     293.0                     SILICON
*
*...+....1....+....2....+....3....+....4....+....5....+....6....+....7....+....
*
*   Now define compounds and mixtures:
*
*   Air (overrides pre-defined mat. 24): 3 components, N (mat. 7), O (mat. 8),
*   gaseous Ar (mat. 18)
MATERIAL                         0.001225      24.0                     AIR
COMPOUND  -0.9256E-3       7. -0.2837E-3       8.0 -1.572E-5      18. AIR
*    Mat. 28: Mixture of lead and liquid Ar at 87 K (mat. 27 and 17)
MATERIAL                             4.61      28.0                     LAR-LEAD
COMPOUND       2.378     27.0      1.768      17.0                     LAR-LEAD
*    Mat. 30: another mixture of lead and liquid Ar 87 K, in different amounts
MATERIAL                         9.733333      30.0                     HLAR-LEA
COMPOUND    0.351741     27.0    2.76119      17.0                     HLAR-LEA
*    Polyethylene (mat. 29): H (material 3) and C (material 6)
MATERIAL                             0.94      29.0                     POLY-CH2
COMPOUND         2.0       3.0       1.0       6.0                     POLY-CH2
*    G10 (mat. 31): made of H, O, Si, Ca (mat. 3, 8, 26, 23)
MATERIAL                             1.85      31.0                     G10
COMPOUND     -0.0071       3.0     -0.53       8.0     -0.27      26. G10
COMPOUND      -0.191      23.0                                          G10
*    Again Polyethylene, but very low density (mat. 32)
MATERIAL                             0.06      32.0                     POLY-TRD
COMPOUND         2.0       3.0       1.0       6.0                     POLY-TRD
*    Lead-scintillator mixture (mat. 33), made of H, C, Pb (mat. 3, 6, 16)
MATERIAL                        10.502333      33.0                     LEADSCIN
COMPOUND  -0.0808308       3.0 -0.963192       6.0  -9.458333     16. LEADSCIN
```

Chapter 16

History of FLUKA

16.1 Introduction

The history of FLUKA goes back to 1962-1967. During that period, Johannes Ranft was at CERN doing work on hadron cascades under the guide of Hans Geibel and Lothar Hoffmann, and wrote the first high-energy Monte Carlo transport codes.

Starting from those early pioneer attempts, according to a personal recollection of Ranft himself [147], it is possible to distinguish three different generation of "FLUKA" codes along the years, which can be roughly identified as the FLUKA of the '70s (main authors J. Ranft and J. Routti), the FLUKA of the '80s (P. Aarnio, A. Fassò, H.-J. Möhring, J. Ranft, G.R. Stevenson), and the FLUKA of today (A. Fassò, A. Ferrari, J. Ranft and P.R. Sala).

These codes stem from the same root and of course every new "generation" originated from the previous one. However, each new "generation" represented not only an improvement of the existing program, but rather a quantum jump in the code physics, design and goals. The same name "FLUKA" has been preserved as a reminder of this historical development — mainly as a homage to J. Ranft who has been involved in it as an author and mentor from the beginning until the present days — but the present code is completely different from the versions which were released before 1990, and far more powerful than them.

16.2 First generation (the CERN SPS Project, 1962-1978)

The first codes of Ranft [75, 123–126, 129] were originally non-analogue and were used as a tool for designing shielding of high energy proton accelerators. The first analogue cascade code was written and published at Rutherford High Energy Lab, where Ranft worked from 1967 to 1969. His work was supported by Leo Hobbis of RHEL, who at the time was the radiation study group leader for the CERN 300 GeV project. The analogue code was called FLUKA (**FLU**ktuierende **KA**skade), and was used to evaluate the performances of NaI crystals used as hadron calorimeters [126].

Around 1970, J. Ranft got a position at Leipzig University. During the SPS construction phase in the Seventies he was frequently invited by the CERN-Lab-II radiation group, leader Klaus Goebel, to collaborate in the evaluation of radiation problems at the SPS on the basis of his hadron cascade codes. These codes were FLUKA and versions with different geometries and slightly differing names [156]. Jorma Routti, of Helsinki University of Technology, collaborated with Ranft in setting up several of such versions [130, 131]. The particles considered were protons, neutrons and charged pions.

At that time, FLUKA was used mainly for radiation studies connected with the 300 GeV Project [49, 76, 77]. During that time, the development of FLUKA was entirely managed by Ranft, although many suggestions for various improvements came from Klaus Goebel, partly from Jorma Routti and later from Graham Stevenson (CERN). In that version of FLUKA, inelastic hadronic interactions were described by means of an inclusive event generator [131, 134]. In addition to nucleons and charged pions, the generator could now sample also neutral pions, kaons and antiprotons.

Ionisation energy losses and multiple Coulomb scattering were implemented only in a crude way, and a transport cut-off was set at 50 MeV for all particles. The only quantities scored were star density and energy deposited. The electromagnetic cascade and the transport of low-energy particles were not simulated in detail but the corresponding energy deposition was sampled from "typical" space distributions.

16.3 Second generation (development of new hadron generators, 1978-1988)

After the SPS construction phase, a complete re-design of the code was started in 1978 on the initiative of Graham Stevenson (CERN), with the support of Klaus Goebel, then leader of the CERN Radiation

Protection Group, and of Jorma Routti, Chairman of the Department of Technical Physics at the Helsinki University of Technology (HUT), in the form of a collaboration between CERN, HUT and Leipzig University [1, 2, 105, 139]. The goal was to make FLUKA a more user friendly hadron cascade code with flexible geometry and with a modern formulation of the hadron interaction model. The new FLUKA code was started to be written by visitors from Leipzig University (H.-J. Möhring) and Helsinki Technical University (Jorma Sandberg). The project was finished by Pertti Aarnio, also visitor from Helsinki. Other contributions came from Jukka Lindgren (Helsinki) and by Stevenson himself, who was acting as a coordinator.

The existing versions of Ranft's programs (at least 14) were unified into a single code under the name FLUKA. The new code was capable to perform multi-material calculations in different geometries and to score energy deposition, star density and differential "fluxes" (actually, angular yields around a target).

This second generation resulted in the release of several versions. In FLUKA81 [105] only one geometry was available (cylindrical). High-energy hadronic events were still sampled from inclusive distributions, but the low-energy generators HADRIN [79, 84] and NUCRIN [80, 85] were introduced for the first time.

In FLUKA82 [1, 139], Cartesian and spherical geometries were added, and in principle Combinatorial Geometry too (but the latter option was rarely used, since there was no scoring associated with it and it did not support charged particle multiple scattering and/or magnetic fields). After a first release with the old inclusive hadron generator, an update [2] was released soon in which a new quark-chain generator developed by Ranft and his collaborators was introduced in a tentative way [135, 137, 138]. At least four Ph.D. projects at Leipzig University did contribute to this new generator, based on the Dual Parton Model, known as EVENTQ. The model soon turned out to be superior by far to all those used before in hadron Monte Carlo, and various versions of it were later adopted also in other codes (HETC [9, 10], HERMES [40], CALOR [74], and the simulation codes used for the H1 and ZEUS experiments).

The link to the EGS4 program [116] was introduced in the FLUKA86 version by G.R. Stevenson and A. Fassò, as an alternative to the parameterised electromagnetic cascade used before. The link was working both ways, allowing to transport gammas issued from π^0 decay, and also photohadrons. Production of the latter was implemented only for energies larger than the Δ resonance, in the frame of the Vector Meson Dominance model, by J. Ranft and W.R. Nelson [142].

The possibility to work with complex composite materials was introduced in the FLUKA81 version by Möhring and Sandberg. P. Aarnio restructured the code by encapsulating all COMMON blocks into `INCLUDE` files. In that version, and in FLUKA87 which soon followed [4], several other new features were introduced. A first attempt at simulating ionisation fluctuations (with the Landau approach) was contributed by P. Aarnio, and a rudimentary transport of particles in magnetic fields was provided by J. Lindgren (for charged hadrons only). Some fluence estimators (boundary crossing, collision, track-length) were added in a preliminary form by Alberto Fassò, based on the same algorithms he had written for the MORSE code [50]. J. Ranft and his group improved the EVENTQ hadron generator with the inclusion of diffractive events and Fermi momentum and provided a first model (later abandoned) of nucleus-nucleus collisions.

Practically none of these features, however, is surviving today in same form: in all cases, with the exception of the hadron event generator, even the basic approach is now completely different.

16.4 Third generation (the modern multiparticle/multipurpose code, 1989 to present)

At about the time when the last version was frozen (1987), a new generation of proton colliders, with large luminosities and energies of the order of several TeV, started to be planned. Because of its superior high-energy hadron generator, FLUKA became the object of a great interest and began to be employed for shielding calculations and especially to predict radiation damage to the components of the machines and of the experiments. But soon many limitations of the code became evident: the design of the new accelerators (SSC and LHC) and associated experiments needed a capability to handle large multiplicities, strong magnetic fields, energy deposition in very small volumes, high-energy effects, low-energy neutron interactions, which the code was lacking. A. Ferrari (INFN) and A. Fassò set up a plan to transform FLUKA from a high-energy code mostly devoted to radiation shielding and beam heating into a code which could handle most particles of practical interest and their interactions over the widest possible energy range.

This plan was entirely supported by INFN, since after the retirement of K. Goebel, the CERN Radiation Protection Group had decided to stop support to any further FLUKA development. The Leipzig group was dissolved following Germany reunification, but J. Ranft continued to contribute, especially during three 6-months stays in different INFN labs.

Over a period of six years, FLUKA evolved from a code specialised in high energy accelerator shielding, into a multipurpose multiparticle code successfully applied in a very wide range of fields and energies, going much beyond what was originally intended in the initial development reworking plan of Fassò and Ferrari. Just as examples, a few of the fields where the modern FLUKA has been successfully applied are listed in the following:

- Neutrino physics and Cosmic Ray studies: initiated within ICARUS
  - Neutrino physics: ICARUS, CNGS, NOMAD, CHORUS
  - Cosmic Rays: First 3D neutrino flux simulation, Bartol, MACRO, Notre-Dame, AMS, Karlsruhe (CORSIKA)
  - Neutron background in underground experiments (MACRO, Palo Verde)
- Accelerators and shielding: the very first FLUKA application field
  - Beam-machine interactions: CERN, NLC, LCLS, IGNITOR
  - Radiation Protection: CERN, INFN, SLAC, Rossendorf, DESY, GSI, TERA, APS
  - Waste Management and environment: LEP dismantling, SLAC
- Synchrotron radiation shielding: SLAC
- Background and radiation damage in experiments: Pioneering work for ATLAS
  - all LHC experiments, NLC
- Dosimetry, radiobiology and therapy:
  - Dose to Commercial Flights: E.U., NASA, AIR project (USA)
  - Dosimetry: INFN, ENEA, GSF, NASA
  - Radiotherapy: Already applied to real situations (Optis at PSI, Clatterbridge, Rossendorf/GSI)
  - Dose and radiation damage to Space flights: NASA, ASI
- Calorimetry:
  - ATLAS test beams
  - ICARUS
- ADS, spallation sources (FLUKA+EA-MC, C. Rubbia *et al.*)
  - Energy Amplifier
  - Waste transmutation with hybrid systems
  - Pivotal experiments on ADS (TARC, FEAT)
  - nTOF

This effort, mostly done in Milan by Ferrari and Paola Sala (also of INFN), started in 1989 and went off immediately in many directions: a new structure of the code, a new transport package including in particular an original multiple Coulomb scattering algorithm for all charged particles, a complete remake of the electromagnetic part, an improvement and extension of the hadronic part, a new module for the transport of low-energy neutrons, an extension of Combinatorial Geometry and new scoring and biasing facilities. At the end of 1990, most of these goals had been achieved, although only in a preliminary form. All the new features were further improved and refined in the following years.

16.4.1 Code structure

One of the first changes which led to the modern FLUKA was a complete redesign of the code structure. The main change was a general dynamical allocation scheme allowing to obtain a great flexibility while keeping the global memory size requirements within reasonable limits. Other improvements were a re-arrangement of `COMMON` blocks to optimise variable alignment, a parameterisation of constants making the program easier

to maintain and update, the possibility to insert freely comments in input, and a special attention devoted to portability (FLUKA87 could run only on IBM under VM-CMS).

The greatest importance was attached to numerical accuracy: the whole code was converted to double precision (but the new allocation scheme allowed for implementation also in single precision on 64-bit computers). As a result, energy conservation was ensured within 10^{-10}.

A decision was also made to take systematically maximum advantage from the available machine precision, avoiding all unnecessary rounding and using consistently the latest recommended set of the physical constant values [118]. Such an effort succeeded in strongly reducing the number of errors in energy and momentum conservation and especially the number of geometry errors.

A double precision random number generator was also adopted, kindly provided by Fred James (CERN) [90], and based on the algorithm of RANMAR by Marsaglia and Zaman of Florida State University [100, 101]. The possibility to initialise different independent random number sequences was introduced in 2001. In 2005, the newly proposed double-precision generator proposed by Marsaglia and Tsang [102] has been implemented.

A deliberate choice was made at an early stage to give preference to table look-up over analytical parameterisations or rejection sampling. The burden of large file management was more than compensated by the better accuracy and increased efficiency. Cumulative tabulations optimised for fast sampling were initialised at run-time for the materials of the problem on hand, and were obtained mainly from complete binary data libraries stored in external files.

The concern for self-consistency was and still is the main guiding principle in the design of the code structure. The same attention has been devoted to each component of the hadronic and of the electromagnetic cascade, with the aim of obtaining a uniform degree of accuracy. For this reason, FLUKA can now be used just as well to solve problems where only a single component is present (pure hadron, neutron, muon or electromagnetic problems). It has also been tried to give a complete description of the mutual interaction between the different components, preserving the possible correlations.

A set of default settings recommended for several applications (shielding, radiotherapy, calorimetry etc.) was introduced in 1996 to help the user in a difficult task, but essential to get reliable results.

16.4.2 Geometry

The original Combinatorial Geometry (CG) package from MAGI [78, 98] was adopted and extensively improved by Fassò and Ferrari, starting from the one used in their improved version of the MORSE code [44]. In 1990, new bodies were added (infinite planes and cylinders) which made the task of writing geometry input much easier and allowed more accurate and faster tracking.

CG had originally been designed for neutral particles, but charged particles definitely required a fully new treatment near boundaries, especially when magnetic fields were present. Previous attempts to use CG to track charged particles, in FLUKA87, EGS4 and other codes, had always resulted in a large number of particle rejections, due to rounding errors and to the "pathological" behaviour of charged particles scattering near boundaries and in the practical impossibility to use CG for these purposes.

The tracking algorithm was thoroughly redesigned attaining a complete elimination of rejections. A new tracking strategy brought about large speed improvements for complex geometries, and the so-called `DNEAR` facility (minimum distance from any boundary) was implemented for most geometry bodies and all particles. A sophisticated algorithm was written to ensure a smooth approach of charged particles to boundaries by progressively shortening the length of the step as the particle gets closer to a boundary. Boundary crossing points could now be identified precisely even in the presence of very thin layers. The combined effect of multiple scattering and magnetic/electric fields was taken into account.

In 1994, the PLOTGEOM program, written by R. Jaarsma and H. Rief in Ispra and adapted as a FLUKA subroutine by G.R. Stevenson in 1990, was modified by replacing its huge fixed dimension arrays with others, dynamically allocated. The same year, a repetitive (lattice) geometry capability was introduced

in CG by Ferrari, and a powerful debugger facility was made available.

In 1997-1998, following a request from the ATLAS experiment, INFN hired a fellow, S. Vanini, who, together with Sala, developed an interface called FLUGG which allows to use FLUKA using the GEANT4 geometry routines for detector description [32,33]. This interface was further improved by Sala and in recent times I. Gonzalez and F. Carminati from ALICE.

In 2001-2002, in the framework of a collaboration between INFN-Milan and GSF (Germany), Ferrari developed a generalised voxel geometry model for FLUKA. This algorithm was originally developed to allow to use inside FLUKA the human phantoms developed at GSF out of real person whole body CT scans. It was general enough to be coupled with generic CT scans, and it is already used in Rossendorf (Germany) for hadron therapy applications.

16.4.3 Particle transport

The number of particles which can be transported by FLUKA was 25 in 1990; after the muon (anti)neutrinos and several hyperons were added, the number increased to 36. In 1992, transport of light ions (deuterons, tritons, ^{3}He, α) was introduced, without nuclear interactions. In 1996 τ leptons and neutrino transport (and in some cases interactions) were added. In 1999 the transport of optical photons was set up. The same year charmed hadrons brought the total number of transported particles to 63, plus all kinds of heavy ions.

The transport threshold, originally fixed at 50 MeV has been lowered since 1991 to the energy of the Coulomb barrier. Below that value, charged particles were ranged out to rest.

16.4.4 Particle decays

The old FLUKA allowed for two and three body phase-space like particle decays with no account for possible matrix element, particle polarisations and higher multiplicities. Starting since 1995, particle decays have been rewritten from scratch by Ferrari allowing for more than 3 body decays, implementing polarised decays for π's, kaons, τ's and muons when relevant, and introducing proper matrix elements for kaon and muon decays. Among these features, polarised muon decays with the proper V–A matrix element were developed by G. Battistoni (INFN-Milan) and $K_{\mu 3}$ and K_{l3} decays of $K^{\pm}/K_{Long}$ with the proper matrix element were based on the KL3DCAY code kindly provided by Vincenzo Patera (INFN-Frascati).

Various methods of particle decay biasing were also introduced by Ferrari (described later in 16.4.14).

16.4.5 Magnetic fields

Transport in magnetic fields was extended to electrons and positrons in 1990 by Ferrari. In 1992 and again in 1994, the magnetic field tracking algorithm was completely reworked by Ferrari and Sala in order to overcome the many limitations of the previous one. The new algorithm was much more precise and fast, it supported complex particle/boundary interactions typical of multiple scattering, allowed for charged particle polarisation and did no longer miss by construction any geometrical feature, even if small, met along the curved path.

16.4.6 Multiple Coulomb scattering

The version of EGS4 which had been linked to FLUKA87 was an early one, which did not include the PRESTA algorithm for the control of the multiple scattering step and was therefore very sensitive to the step length chosen. In 1989, Ferrari and Sala developed and implemented in FLUKA an independent model which had several advantages even with respect to PRESTA: it was accounting for several correlations, it sampled the path length correction accounting for the first and second moment of its distribution, it allowed the introduction of high-energy effects (nuclear form factors) and could be easily integrated within the Combinatorial Geometry package. The algorithm, which included also higher order Born approximations, was very efficient and was taking care also of special high-energy effects, very grazing angles, correlations between angular, lateral and longitudinal displacements, backscattering near a boundary etc.

The Ferrari-Sala model, which has proved very robust and has been shown to reproduce with good accuracy even backscattering experiments, was applied to both electrons and heavier charged particles. The

final revision and update of the algorithm were made in 1991. In 1995, the Fano correction for multiple scattering of heavy charged particles [48] was introduced.

16.4.7 Ionisation losses

The treatment of ionisation losses was completely re-written in 1991-1992 by Fassò and Ferrari to eliminate many crude approximations, and delta-ray production was added. Ranging of stopping charged particle was also changed. Quenching according to the Birks law was introduced to calculate the response of scintillators.

Application of FLUKA to proton therapy called for further refinements of stopping power routines in 1995, with the inclusion of tabulated data of effective ionisation potentials and density effect parameters. Shell corrections were added. The new treatment was fully compliant with ICRU recommended formulae and parameters and included all corrections, including low-energy shell corrections as worked out by Ziegler *et al.*.

In 1996, a new formalism for energy loss fluctuations by Ferrari [57] replaced the old treatment of Landau fluctuations. This formalism, based on the statistical properties of the cumulants of a distribution, was applied to both heavy charged particles and e^+e^-, and was fully compatible with any user-defined threshold for delta ray emission.

Other improvements concerned the possibility to define materials with local density different from average (porous substances), and the ranging out of particles with energies lower than the transport cut-off.

In 1999-2000, heavy ion dE/dx was improved by the inclusion of effective Z and straggling (Ferrari).

High-energy energy loss mechanisms for heavy charged particles were implemented by Ferrari both as a continuous and as an explicit treatment: bremsstrahlung and pair production in 1992, nuclear interaction via virtual photons in 1993.

16.4.8 Time dependence

Time-dependent calculations were made available in FLUKA for all particles since 1991. Time cut-offs could be set by the user for both transport and scoring.

16.4.9 Nuclear data and cross-sections

All the nuclear data used by the original evaporation routines by Dresner [43] (see below), were replaced by Ferrari at an early stage with the most recent ones available from the NNDC database, thanks to the help of Federico Carminati. In 1990, the isotopic composition of elements was included, taken from modern evaluations.

In 1991, the proton and neutron inelastic cross-sections between 10 and 200 MeV were updated by Ferrari and Sala with fits to experimental data. An accurate treatment of cross-section energy dependence for all charged particles, independent of the step size, was introduced at that stage through the fictitious-σ method.

Hadron-nucleus cross-sections underwent further extensive reworking starting from 1994 by Ferrari and Sala. The present treatment is based on a novel approach blending experimental data, data driven theoretical approaches, PDG fits and phase shift analysis when available.

Elastic scattering on hydrogen of protons, neutrons, and pions was rewritten from scratch in 1994 by Ferrari and Sala. The same work was done for kaons in 1997.

16.4.10 Electron and photon transport (EMF)

The original EGS4 implementation in FLUKA was progressively modified, substituded with new algorithms and increasingly integrated with the hadronic and the muon components of FLUKA, giving rise to a very different code, called EMF (Electro-Magnetic-Fluka). In 2005, the last remaining EGS routine has been eliminated, although some of the structures still remind of the original EGS4 implementation.

The main developments were made according to the following sequence.

The Ferrari-Sala multiple scattering algorithm was the first major addition in 1989. It has already been described elsewhere since it was applied to hadrons and muons as well. Following its implementation, the whole electron/positron transport algorithm had to be reworked from scratch in order to comply with the features (initial and final step deflections, complex boundary crossing algorithm) of the new model.

In 1990, the treatment of photoelectric effect was completely changed. Shell-by-shell cross-sections were implemented, the photoelectron angular distribution [155] was added, taking into account the fine structure of the edges, and production of fluorescence X-rays was implemented.

Many new features were added in 1991. The emission angle of pair-produced electron and positrons and that of bremsstrahlung photons, which were only crudely approximated in the original EGS4 code, were now sampled from the actual physical distributions.

The full set of the electron-nucleus and electron-electron bremsstrahlung cross-sections, differential in photon energy and angle, published by Seltzer and Berger for all elements up to 10 GeV [158] was tabulated in extended form and introduced into the code together with a brand new sampling scheme by Fassò and Ferrari. The energy mesh was concentrated, especially near the photon spectrum tip, and the maximum energy was extended to higher energies. The Landau-Pomeranchuk-Migdal effect [96, 97, 103, 104] for bremsstrahlung and the Ter-Mikaelyan polarisation effect [167] (suppressing soft photon emission) were implemented.

Positron bremsstrahlung was treated separately, using below 50 MeV the scaling function for the radiation integral given by Kim [91] and differential cross-sections obtained by fitting proper analytical formulae to numerical results of Feng *et al.* [63]. The photon angular distribution was obtained by sampling the emission angle from the double differential formula reported by Koch and Motz [92], fully correlated with the photon energy sampled from the Seltzer-Berger distributions.

The Compton effect routines were rewritten in 1993 by Ferrari and Luca Cozzi (University of Milan), including binding effects. At the end of the same year, the effect of photon polarisation was introduced for Compton, Rayleigh and photoelectric interactions by Ferrari.

In 1993 and 1994, A. Fassò and A. Ferrari implemented photonuclear reactions over the whole energy range, opening the way to the use of Monte Carlo in the design of electron accelerator shielding [54]. Giant Dipole Resonance, Delta Resonance and high-energy photonuclear total cross-sections were compiled from published data [58] (further updated in 2000 and 2002), while the quasideuteron cross-section was calculated according to the Levinger model, with the Levinger's constant taken from Tavares *et al.* [166], and the damping factor according to Chadwick *et al.* [39]. The photon interaction with the nucleus was handled in the frame of the FLUKA hadronic event generators PEANUT and DPM (see below).

In 1995, a single Coulomb scattering option was made available for electrons and positrons by Ferrari and Sala. The aim of this option was mainly to eliminate some artefacts which affected the angular distributions of charged particles crossing a boundary, but it turned out very useful also to solve some problems at very low electron energy or with materials of low density (gases). In the same year, the electron transport algorithm was reworked once more by Ferrari and Sala introducing an adaptive scheme which "senses" close boundaries in advance and automatically adapts the step length depending on their distance. Also in 1995 Ferrari discovered that the EGS4 implementation of Møller and Bhabha scattering, still used at that time in FLUKA, was flawed. The bug was duly reported to the EGS4 authors who took corrective actions on their own code, while Ferrari developed a new algorithm for Møller and Bhabha scattering for FLUKA.

In 1997 mutual polarisation of photons emitted in positron annihilation at rest was introduced by Ferrari.

Cherenkov photon production and optical photon transport was implemented in 1999 by Ferrari. In 2002 scintillation photon production was added as well.

In 2005, new implementations of the pair production and Rayleigh scattering processes have been finalized.

In 1998-2001 an improved version of the Ferrari-Sala multiple scattering model was developed, introducing further refinements and the so called "polygonal" step approach. This version is presently fully tested offline and will be soon introduced into the production code.

In 2005, the need for an external data preprocessor has been eliminated, integrating all the needed functionalities into the FLUKA initialisation stage. At the same time, data from the EPDL97 [47] photon data library have become the source for pair production, photoelectric and total coherent cross-section tabulations, as well as for atomic form factor data.

At the same time, Rayleigh scattering has been reworked from scratch by Ferrari with a novel approach, and the photoeletric interaction model has been further improved with respect to the 1993 Ferrari-Sala approach, extending it, among the other modifications, down to a few eV.

Finally the energy sampling for pair production has been completely reworked by Ferrari using a vastly superior approach, which can distinguish between interactions in the nuclear or electron field, and properly samples the element in a compound or mixture on which the interaction is going to occur. Thew algorithm is also capable of producing meaningful results for photon energies close to thresholds where several corrections are important and the symmetry electron/positron is broken, in similar fashion to the bremsstrahlung case.

16.4.11 Low-energy neutrons

The 50 MeV energy cut-off was one of the most important limitations of the FLUKA87 version. The cut-off concerned muons and all hadrons, but it was the absence of neutron transport below 50 MeV which constituted the most serious drawback for many applications. The limitations stemmed from the increasing inadequacy of the hadron interaction models in dealing with interactions below 1 GeV and with the lack of any detailed nuclear physics treatment, i.e., the lack of an evaporation model and low-energy particle production, at all energies.

Actually, several early attempts to overcome these weaknesses of the code had been made by H.-J. Möhring, H. Kowalski and T. Tymieniecka (code NEUKA [93,171], for Uranium/Lead scintillator only) and J. Zazula (code FLUNEV [176,177]), with mixed results. The most promising approach was that of Jan Zazula, of the Institute of Nuclear Physics in Cracow: he had coupled FLUKA87 with the EVAP-5 evaporation module which he had extracted from the HETC/KFA code, and then interfaced the code with the only available multi-group neutron cross section library extending to 50 MeV and beyond, the HILO library.

The main limitations of these approaches was their inability to address the real deficiencies of the FLUKA87 hadron interaction model, their lack of nuclear physics details and therefore the unreliability of their excitation energy predictions, which indeed were never intended by the original authors for any real use.

Furthermore, it became apparent that HILO had several weaknesses: the cross-section set had been put together by extending a low-energy one of rather coarse structure based on evaluated experimental data with the addition of much less accurate data calculated with an intranuclear cascade code (HETC); for the same reason the library did not contain any information on (n,γ) generation above 20 MeV and was based on a different Legendre angular expansion below and above that energy. And because the library contained a very small number of materials, the possibilities of application were rather limited.

The approach followed by Ferrari and Sala to overcome those shortcomings was two-fold:

- improve/substitute the hadronic interaction models in order to describe with reasonable accuracy low-energy particle production and medium-low energy particle interactions
- develop an ad-hoc neutron cross-section library for FLUKA extending up to 20 MeV (in collaboration with G.C. Panini and M. Frisoni of ENEA-Bologna [42])

The former point is discussed in detail in the section on hadronic models, the latter in the following.

Since Ferrari and Sala had started to work on a preequilibrium model (later known as PEANUT, see next section) which was expected to cover intermediate energies more accurately than the traditional intranuclear cascade codes, it was decided to lower the FLUKA energy cut-off to 20 MeV (thus making HILO unnecessary) and to create a dedicated multigroup neutron cross-section library to be used with FLUKA, with the more usual upper energy limit of 20 MeV. The task was carried out with the essential collaboration of G.C. Panini, an expert of an ENEA laboratory in Bologna specialised in nuclear data processing for reactor and fusion applications. Several neutron cross-section libraries have been produced for FLUKA over the years as a result of a contract between INFN-Milan and ENEA [42].

These libraries, designed by Ferrari, had a format which was similar to the ANISN one [46] used for example by MORSE [44], but which was modified to include partial cross-sections and kerma factors for dose calculations (critically revised). Because at that time there was still a US embargo on the most recent ENDF/B evaluated file, the cross-sections were originally derived from the European compilations JEF–1 and JEF–2. (Later, they were regularly updated with the best ones available from JEF, ENDF, JENDL and others). The choice of materials was particularly tailored on detector and machine applications for high-energy colliders, including also cryogenic liquids at various temperatures, and was much wider than in most other libraries: it contained initially about 40 different materials (elements or isotopes), which became soon 70 (in 1991) and are now more than 130. Hydrogen cross-sections were also provided for different H molecular bonds (H gas, water, polyethylene). Doppler reduced broadening was implemented for a few materials at liquid argon (87 K) and liquid helium (approximately 0 K) temperatures.

The incorporation of the neutron multigroup transport module into FLUKA by Ferrari was loosely based on the approach followed in the MORSE and other multigroup codes, Ferrari and Fassò had a deep expertise about. The low-energy neutron transport and interaction routines were rewritten from scratch progressively introducing many extra features which are detailed in the following. Capture and inelastic gamma generation was still implemented in the multigroup framework, but gamma transport was taken care of by the EMF part of FLUKA. Survival biasing was left as an option to the user with the possibility to replace it by analogue survival.

Energy deposition computed via kerma factors was preserved, but in the case of hydrogen the recoiling protons were explicitly generated and transported. The same was done with protons from the ^{14}N(n,p)^{14}C reaction due to its importance for tissue dosimetry, and later for all reaction on ^{6}Li.

The new low-energy neutron transport was ready at the end of 1990 [65]. The corresponding FLUKA version was called FLUKAN for a while to underline the neutron aspect, but the name was soon abandoned.

At the beginning of 1997, the possibility to score residual nuclei produced by low-energy neutrons was introduced. Many improvements were made in that same year. Federico Carminati, who was using FLUKA for calculations related to C. Rubbia's energy amplifier, added to the program a few routines and nuclear data necessary to score low-energy fission products. Pointwise cross-sections were introduced for the neutron interactions with hydrogen. Ferrari and Sala also developed a model to sample gammas from neutron capture in cadmium, an important reaction for which no data were available in evaluated cross-section files. Similar schemes for capture photon generation were established in 2001 for other nuclei (Ar, Xe) [61]. Pointwise neutron transport and interactions for ^{6}Li were also provided.

16.4.12 Hadron event generators

The two Leipzig event generators developed in the 80's, one for intermediate energies and the other for high energies (> 5 GeV), were a remarkable achievement with great potentialities. In particular the high energy model was among the first developed in the world based on partonic ideas and quark degrees of freedom (specifically on the so called Dual Parton Model [34, 35]).

The part of the code concerning hadron-nucleus primary interactions at energies above 4 GeV has been extensively extended and updated since 1987 and is now virtually a new model, even though the physics foundations are still essentially the same. Several bugs and approximations have been removed too. The intermediate energy resonance model has also been deeply modified and its use is currently restricted to few particles over a small energy range. The newly developed preequilibrium-cascade model PEANUT has progressively replaced this model.

The main lines of the work developed mostly in Milan by Ferrari and Sala starting from 1990 can be summarised as follow [41, 70]:

- further develop and improve the high energy DPM-based part of the models. This was performed in 4 main stages, which eventually led to an almost completely new code still based on the same physics foundations with some extensions.
- introduce a self-consistent nuclear environment in both the medium and high energy models, allowing for a physically meaningful excitation energy and excited residual A and Z calculations.
- develop a state-of-the-art evaporation/fission/break-up/deexcitation stage to describe the "slow" part of nuclear interactions. These actually took place in two major steps.
- develop a completely new model (PEANUT) based on a novel approach for describing the low-to-intermediate (up to a few GeV) energy range, while progressively phasing out the improved version of the intermediate energy Leipzig code. This effort took also place in two main steps.

16.4.12.1 High energy model improvements

In all the developments described in this section and also in some other sections, J. Ranft always acted as the main mentor and source of theoretical and often practical support. Even when he did not contributed to the code directly, his ideas, help and suggestions were essential part of its development.

The two models developed by the Leipzig group were initially improved by removing a number of known bugs and approximations (mainly, but not only, in the kinematics). In the years 1990-1991 all hyperons and anti-hyperons were added as possible projectiles, and most important, nuclear effects, previously restricted to Fermi momentum, were expanded and treated more accurately, with an explicit treatment of the nuclear well potential, the inclusion of detailed tables of nuclear masses to account for nuclear binding energy, a consistent exact determination of nuclear excitation energy and an overall "exact" conservation of energy and momentum on an event-by-event basis. These changes were the minimal modifications required for introducing a sensible evaporation module and related low-energy particle production: they made up the first stage of upgrade of the intermediate and high energy event generator and were performed by Ferrari and Sala.

In the following years, Negative Binomial multiplicity distribution, correlations between primary interactions and cascade particles and better energy-angle distributions were implemented. Sea quark distributions were updated, new distributions were used for the number of primary collisions using an improved Glauber cascade approach, and reggeon mediated interactions (single chains) were introduced at the lower energy end of the application range of the Dual Parton Model. An initial improvement of the diffraction treatment as well of the hadronisation algorithm were performed. These developments ended up in the 1993 version, which represented the second stage of the high energy generator development (and which was made available to GEANT3 users, see later).

Several major changes were performed on both the intermediate and high energy hadron generator in the years 1994-1996 by Ferrari and Sala. The latter was extensively improved, bringing its results into much better agreement with available experimental data from as low as 4 GeV up to several hundreds of GeV. A fully new treatment of transverse momentum and of all DPM in general was developed, including a substantially improved version of the hadronisation code and a new driver model for managing two-chain events. The existing treatment of high-energy photonuclear reactions, previously already based on the VMD model but in an approximate way, was improved by implementing the contribution of all different vector mesons, as well as the quasielastic contribution. The simulation of diffractive events was completely reworked distinguishing between resonant, single-chain and two-chain events, and a smeared mass distributions for resonance was introduced. This version of the model was completed in 1996 and performed very well together with the new "sophisticated" PEANUT when applied to a variety of problems, ranging from radiation protection, to cosmic ray showers in the atmosphere and to the test beam of the ATLAS calorimeters.

The latest round of improvements originated by the new interest of Ferrari and Sala for neutrino physics, triggered by their participation in the ICARUS experiment and resulted in several improvements in the high-energy interaction model. In 1998, a new chain fragmentation/hadronisation scheme was put to use, and a new diffraction model was worked out once more according to rigorous scaling, including

low-mass diffraction and antibaryon diffraction. In 1999, charm production was set up by Ranft and Ferrari (reasonable at least for integrated rates), and charmed particle transport and decay were introduced. The chain building algorithm was thoroughly revised to ensure a continuous transition to low energies, and a significant reworking was done on the chain hadronisation process, providing a smooth and physically sound passage to chains made up by only two particles, resulting in an overall better description of particles emitted in the fragmentation region. This model was thoroughly benchmarked against data taken at WANF by NOMAD and the particle production data measured by SPY. It constituted the basis for all calculations performed for CNGS, both in the early physics design stage and later in the optimisation and engineering studies.

16.4.12.2 Cascade-Preequilibrium model (PEANUT)

There were two main steps in the development of the FLUKA preequilibrium cascade model (PEANUT) by Ferrari and Sala:

- The so called "linear" preequilibrium cascade model
- The full preequilibrium cascade model

The first implementation of the FLUKA cascade-preequilibrium model, the "linear" one was finalised in July 1991 [66]. The model, loosely based for the preequilibrium part on the exciton formalism of M. Blann and coworkers called Geometry Dependent Hybrid Model (GDH) [23, 24] now cast in a Monte Carlo form, was able to treat nucleon interactions at energies between the Coulomb barrier (for protons) or 10-20 MeV (for neutrons) and 260 MeV (the pion threshold). The model featured a very innovative concept, coupling a preequilibrium approach with a classical intranuclear cascade model supplemented with modern quantistic corrections. This approach was adopted for the first time by FLUKA and independently by the LAHET code [119] at LANL. Capture of stopping negative pions, previously very crudely handled (the available alternatives being forced decay or energy deposited on the spot) was also introduced in this framework. This first implementation was called "linear" since in the cascade part refraction and reflection in the nuclear mean field were not yet taken into account, resulting in straight ("linear") paths of particles through the nuclear medium. First order corrections for these effects were anyway implemented on the final state angular distributions. This model immediately demonstrated superb performances when compared with nucleon induced particle production data. Its implementation into FLUKA allowed to overcome some of the most striking limitations of the code and permitted the use of the new neutron cross-section library through its ability to produce sound results down to 20 MeV: in this way it opened a huge range of new application fields for the code.

However, despite its nice performances, the "linear" cascade-preequilibrium model was always felt by Ferrari and Sala as a temporary solution for the low end side of particle interactions, while waiting for something even more sophisticated. The work on the "full" cascade-preequilibrium, which in the meantime had been called PEANUT (**P**re-**E**quilibrium **A**pproach to **N**uclear **T**hermalisation) started at the end of 1991 and produced the first fully working version by mid-1993. Despite its improved quality this version was not included into any of the general use FLUKA versions until 1995, due to its complexity and the overall satisfactory results of the "linear" one for most applications. Till 1995, the full version was in use only by a few selected groups, including the EET group led by Carlo Rubbia at CERN, which meanwhile decided to adopt FLUKA as their standard simulation tools above 20 MeV, mostly due to the superior performances of PEANUT full version.

It would be too long to describe in details all features of this model, which represented a quantum jump in the FLUKA performances and a significant development in the field. Actually, PEANUT combines an intranuclear part and a preequilibrium part (very similar in the "linear" and full versions), with a smooth transition around 50 MeV for secondary nucleons and 30 MeV for primary ones. It included nuclear potential effects (refraction and reflection), as well as quantal effects such as Pauli blocking, nucleon-nucleon correlations, fermion antisymmetrisation, coherence length (a new concept introduced by Ferrari-Sala which generalises to low energy and two body scattering the formation zone concept) and formation zone. The model featured a sophisticated pion complex optical potential approach, together with 2 and 3 nucleon absorption processes and took into account the modifications due to the nuclear medium on the pion resonant

amplitudes. For all elementary hadron-hadron scatterings (elastic, charge and strangeness exchanges) extensive use was made of available phase-shift analysis. Particle production was described in the framework of the isobar model and DPM at higher energies, using a much extended version of the original HADRIN code from Leipzig, and the FLUKA DPM model at higher energies.

In 1995, distinct neutron and proton nuclear densities were adopted and shell model density distributions were introduced for light nuclei. The initial model extended the energy range of the original "linear" one from 260 MeV to about 1 GeV in 1994, with the inclusion of pion interactions. Giant Resonance and Quasideuteron photonuclear reactions were added in 1994 and improved in 2000. In 1996-1997 the emission of energetic light fragments (up to α's) in the GINC (Generalised IntraNuclear Cascade) stage emission has been described through the coalescence mechanism.

The upper limit of PEANUT was further increased in 1996 to 1.8 GeV for nucleons and pions, and to 0.6 GeV for K^+/K^0; then again one year later (2.4 GeV for nucleons and 1.6 GeV for pions), and in 2000 (3.5 GeV for both pions and nucleons). In 1998, PEANUT was extended to K^- and $\bar{K}^0$'s induced interactions. In the 2005 version, all nucleon and pion reactions below 5 GeV/c of momentum are treated by PEANUT, while for kaons and hyperons the upper threshold is around 1.5 GeV (kinetic energy). Since 2005 also antinucleon interactions are treated in the PEANUT framework. It is planned to progressively extend PEANUT up to the highest energies by incorporating into its sophisticated nuclear framework the Glauber cascade and DPM part of the high energy model.

One of the fall-outs of the work done for ICARUS was the introduction of nucleon decays and neutrino nuclear interactions in 1997 [38], which prompted improvements in PEANUT, for instance concerning Fermi momentum and coherence length. Quasielastic neutrino interactions can be dealt with by PEANUT natively; in 1999, the code was coupled with the NUX neutrino-nucleon interaction code developed by André Rubbia at ETH Zurich to produce full on-line neutrino-nucleus interactions, including resonance production and deep inelastic scattering. The combined FLUKA(PEANUT)+NUX model gave outstanding results when compared with NOMAD data, therefore giving support to all predictions done for ICARUS.

Negative muon capture was also introduced in 1997 due to ICARUS needs. To much surprise, it turned out to be a key factor in the understanding of the unexpected background at the nTOF facility during its initial operation phase in 2001.

16.4.12.3 Evaporation/Fission and Fermi Break-Up

Evaporation was initially implemented in FLUKA in 1990-1991 by Ferrari and Sala through an extensively modified version of the original Dresner's model based on Weisskopf's theory [43]. Relativistic kinematics was substituted to the original one; fragmentation of small nuclei was also introduced, although initially only in a rough manner. The mass scale was changed to a modern one and the atomic masses were updated from a recent compilation. Improvements included also a more sophisticated treatment of nuclear level densities, now tabulated both with A and Z dependence and with the high temperature behaviour suggested by Ignatyuk [88]. A brand new model for gamma production from nuclear deexcitation was added, with a statistical treatment of E1, E2 and M1 transitions and accounting for yrast line and pairing energy. This "initial capability" evaporation was used together with the first stage improved high energy hadron generator and the HILO library for the very first calculations carried out in 1990 for the LHC detector radiation environment. Later, in 1991, with the introduction of the "linear" preequilibrium model, a full model coverage down to 20 MeV was available and the new neutron cross-section library developed together with ENEA-Bologna [42] started to be used.

In 1993 the RAL high-energy fission model by Atchison [13], kindly provided by R.E. Prael as implemented in the LAHET code, was included after some extensive modifications to remove some unphysical patches which the presence of a preequilibrium stage had now made unnecessary. The model was further developed and improved along the years and little is now left of the original implementation. Competition between evaporation and fission in heavy materials was implemented. This development was set off by a collaboration on energy amplifiers with C. Rubbia's group at CERN. Eventually, Ferrari joined that group in 1998.

In 1995, a newly developed Fermi Break-up model, with a maximum of 6 bodies in the exit channel, was introduced by Ferrari and Sala to describe the deexcitation of light nuclei ($A \leq 17$). This development provided better multiplicities of evaporated neutrons and distributions of residual nuclei. The deexcitation gamma generation model was improved and benchmarked in the following year.

A completely new evaporation treatment was developed by Ferrari and Sala in 1996 and 1997 in substitution of the improved Dresner model. This new algorithm adopted a sampling scheme for the emitted particle spectra which no longer made any Maxwellian approximation, included sub-barrier effects and took the full energy dependence of the nuclear level densities into account. Gamma competition was introduced too. These physics improvements allowed a much more accurate description of the production of residual nuclei. A refinement of this new package took place in 2000/2001. The production of fragments up to mass 24 has been tentatively included around 2003 and subsequently developed and benchmarked [14] and is now available in the distributed version as an option to be activated by the user.

16.4.13 Radioactivity

FLUKA is capable of making predictions about residual nuclei produced in hadronic and electromagnetic showers since late 1994. The accuracy of these predictions steadily improved along the years, in particular after the introduction of the new evaporation/fragmentation and the improvements and extensions of the PEANUT model: versions before the end of 1996 were unlikely to give satisfactory results. Of course, all FLUKA versions prior to 1989 were totally unable to formulate predictions on this issue. Since 1995, an offline code by Ferrari was distributed together with FLUKA, which allowed to compute offline the time evolution of a radionuclide inventory computed with FLUKA, for arbitrary irradiation profiles and decay times. In 2004–2005, this capability has been brought on line by Ferrari and Sala, with an exact analytical implementation (Bateman equations) of the activity evolution during irradiation and cooling down, for arbitrary irradiation conditions. Furthermore, the generation and transport of decay radiation (limited to γ, β^-, and β^+ emissions for the time being) is now possible. A dedicated database of decay emissions has been created by Ferrari and Sala, using mostly informations obtained from NNDC, sometimes supplemented with other data and checked for consistency. As a consequence, results for production of residuals, their time evolution and residual doses due to their decays can now be obtained in the same run, for an arbitrary number of decay times and for a given, arbitrarily complex, irradiation profile.

16.4.14 Biasing

Variance reduction techniques, a speciality of modern FLUKA, have been progressively introduced along the years. Transport biasing under user control is a common feature of low-energy codes, but in the high energy field biasing has generally been restricted to built-in weighted sampling in event generators, not tunable by the user. In addition, Monte Carlo codes are in general either weighted or analogue, but not both. In the modern FLUKA, the user can decide in which mode to run the code, and has the possibility to adjust the degree of biasing by region, particle and energy.

Many different biasing options have been made available. Multiplicity reduction in high-energy hadron-nucleus interactions was the first one to be introduced by Fassò (in 1987), to manage the huge number of secondaries produced by the 20 TeV proton beams of SSC. Ferrari made possible for the user to tune it on a region dependent basis. In 1990 Ferrari added also geometry splitting and Russian Roulette for all particles based on user-defined region importances and several biasing options for low-energy neutrons, inspired by MORSE, but adapted to the FLUKA structure.

Region, energy and particle dependent weight windows were introduced by Fassò and Ferrari in 1992. In this case the implementation was different from that of MORSE (two biasing levels instead of three), and the technique was not applied only to neutrons but to all FLUKA particles. Decay length biasing was also introduced by Ferrari (useful for instance to improve statistics of muons or other decay products, or to amplify the effect of rare short-lived particles surviving at some distance from the production point). Inelastic length biasing, similar to the previous option, and also implemented by Ferrari, makes possible to modify the interaction length of some hadrons (and of photons) in one or all materials. It can be used to force a larger frequency of interactions in a low-density medium, and it is essential in all shielding calculations for electron accelerators.

Two biasing techniques were implemented by Fassò and Ferrari, which are applicable only to low-energy neutrons.

- Neutron Non Analogue Absorption (or survival biasing) was derived from MORSE where it was systematically applied and out of user control. In FLUKA it was generalised to give full freedom to the user to fix the ratio between scattering and absorption probability in selected regions and within a chosen energy range. While it is mandatory in some problems in order to keep neutron slowing down under control, it is also possible to switch it off completely to get an analogue simulation.
- Neutron Biased Downscattering, also for low-energy neutrons, gives the possibility to accelerate or slow down the moderating process in selected regions. It is an option not easily managed by the average user, since it requires a good familiarity with neutronics.

Leading particle biasing, which existed already in EGS4, was deeply modified in 1994 by Fassò and Ferrari, by tuning it by region, particle, interaction type and energy. A special treatment was made for positrons, to account for the penetrating power of annihilation photons.

In 1997, in the framework of his work for ICARUS and CNGS, Ferrari implemented biasing of the direction of decay neutrinos.

16.4.15 Scoring

The stress put on built-in generalised scoring options is another aspect of FLUKA "philosophy" which differentiates it from many other programs where users are supposed to write their own ad-hoc scoring routines for each problem. This characteristics, which was already typical of the old Ranft codes, has allowed to develop in the modern FLUKA some rather sophisticated scoring algorithms that would have been too complex for a generic user to program. For instance the "track-length apportioning" technique, introduced in 1990 by Fassò and Ferrari, used in dose and fluence binning, which computes the exact length of segment travelled by the particle in each bin of a geometry independent grid. This technique ensures fast convergence even when the scoring mesh is much smaller than the charged particle step.

Different kinds of fluence estimators (track-length, collision, boundary crossing) were implemented in 1990-1992, replacing the corresponding old ones. The dimension limitations (number of energy intervals) were removed and replaced by a much larger flexibility due to dynamical memory allocation. Scoring as a function of angle with respect to the normal to a surface at the point of crossing was also introduced. Facilities were made available to score event by event energy deposition and coincidences or anti-coincidences between energy deposition signals in different regions, and to study fluctuations between different particle histories.

The pre-existent option to write a collision file was completely re-written and adapted to the more extended capabilities of the new code. In 1991, time gates became applicable to most scoring facilities, allowing to ignore delayed radiation components such as multiply scattered low-energy neutrons.

In 1994, two new options were added: residual nuclei scoring and scoring of particle yields as a function of angle with respect to a fixed direction. In the latter case, several new quantities can be scored, such as rapidity, various kinematical quantities in the lab and in the centre-of-mass frame, Feynman-x etc.

In 2004–2005, as explained above, the possibility to follow on-line the radiation from unstable residual nuclei has been implemented, together with an exact analytical calculation (Bateman equations) of activity evolution during irradiation and cooling down. As a consequence, results for production of residuals and their effects as a function of time can now be obtained in the same run.

16.4.16 Heavy ions

Heavy ion transport (energy loss, effective charge and associated fluctuations, multiple scattering) was developed by Ferrari as early as 1998 largely based on already existing tools in FLUKA.

There was an increasing demand for extending the FLUKA interaction models to heavy ions, both for basic and applied physics applications (cosmic rays, hadrotherapy, radiation problems in space). A long

standing collaboration has been going on since 1997 with Prof. L. Pinsky, chair of the Physics Department at the University of Houston. This collaboration became formal in 2000 with the issue of a NASA Grant covering three years of FLUKA developments in the field of heavy ion transport and interactions, as well as the development of user friendly tools based on ROOT for a better management of the code (project FLEUR). Further support came from ASI, as a grant to a collaborating group in Milan for hiring a person for one year devoted to these issues.

The DPMJET code has been interfaced to cover the high (> 5 GeV/n) energy range, and an extensively modified version of the RQMD-2.4 code is used at lower energies.

At very low energy, below ≈ 0.1 GeV/n, a treatment based on the Boltzmann Master Equation (BME) is in preparation.

In 2004, a model for electromagnetic dissociation of ions in ion-ion interactions has been implemented [14].

16.4.16.1 DPMJET interface

DPMJET is a high energy hadron-hadron, hadron-nucleus and nucleus-nucleus interaction model developed by J. Ranft, S. Roesler and R. Engel, capable to describe interactions from several GeV per nucleon up to the highest cosmic ray energies. There are strong ties between the FLUKA and the DPMJET teams (with J. Ranft being author of both) and the collaboration is going on since the mid 90's. An interface with DPMJET-2.5 was developed by Toni Empl (Houston), Ferrari and Ranft [45]. The interface allows to treat arbitrary ion interactions in FLUKA at any energy in excess of 5 GeV/n. The excited projectile and target residual leftovers are passed back to the FLUKA evaporation/fission/break-up routines for the final deexcitation and "low" (in the excited residual rest frame) energy particle production. As part of the interface work a new fast multiion/multienergy initialisation scheme has been developed for DPMJET and a new cross-section algorithm has been worked out for runtime calculations based on a fine mesh of DPMJET runs with various ions and energies.

An interface with the new DPMJET-3 [153] has been developed in collaboration of Stefan Roesler (CERN) and is now available.

16.4.16.2 RQMD interface

A very similar interface has been developed by Francesco Cerutti (University of Milan and INFN), Toni Empl (University of Houston), Alfredo Ferrari, Maria Vittoria Garzelli (University of Milan and INFN) and Johannes Ranft, with the Relativistic Quantum Molecular Dynamics code (RQMD) of H. Sorge [160–162]. Also in this case the evaporation and deexcitation of the excited residuals is performed by FLUKA. Significant interventions on the original code were necessary to bring under control the energy/momentum balance of each interaction to allow for a meaningful calculation of excitation energy. This brand new development allows FLUKA to be used for ions from roughly 100 MeV/n up to cosmic ray energies. The results of this modified model can be found in [7,11,62]. Work is in progress to develop a new original code to replace this RQMD interface.

16.4.17 Code size

The present FLUKA alone totals about 400,000 lines of Fortran code (17 MBytes of source code), plus some 60,000 lines (2 MBytes) of ancillary codes used offline to generate and/or test the various data files required for running. Out of these, roughly 1/3 are associated with PEANUT.
As a term of comparison, the latest release of the previous FLUKA generation, FLUKA87, contained roughly 30,000 lines (1.2 MBytes), out of which very few survive in the present code, mostly in the high energy generator and in the old intermediate energy one.

References

[1] P.A. Aarnio, J. Ranft and G.R. Stevenson
A long writeup of the FLUKA82 program
CERN Divisional Report **TIS–RP/106–Rev.** (1984)

[2] P.A. Aarnio, J. Ranft and G.R. Stevenson
First update of FLUKA82, including particle production with a multi-chain fragmentation model (EVENTQ)
CERN **TIS–RP/129** (1984)

[3] P.A. Aarnio, A. Fassò, H.-J. Möhring, J. Ranft, G.R. Stevenson
FLUKA86 user's guide
CERN Divisional Report **TIS–RP/168** (1986)

[4] P.A. Aarnio, J. Lindgren, J. Ranft, A. Fassò, G.R. Stevenson
Enhancements to the FLUKA86 program (FLUKA87)
CERN Divisional Report **TIS–RP/190** (1987)

[5] P.A. Aarnio, A. Fassò, A. Ferrari, H.-J. Möhring, J. Ranft, P.R. Sala, G.R. Stevenson, J.M. Zazula
FLUKA: hadronic benchmarks and applications
Proc. MC93 Int. Conf. on Monte Carlo Simulation in High Energy and Nuclear Physics, Tallahassee (Florida), 22–26 February 1993. Ed. by P. Dragovitsch, S.L. Linn, M. Burbank, World Scientific, Singapore 1994, p. 88–99

[6] P.A. Aarnio, A. Fassò, A. Ferrari, H.-J. Möhring, J. Ranft, P.R. Sala, G.R. Stevenson, J.M. Zazula
Electron-photon transport: always so good as we think? Experience with FLUKA
Proc. MC93 Int. Conf. on Monte Carlo Simulation in High Energy and Nuclear Physics, Tallahassee (Florida), 22–26 February 1993. Ed. by P. Dragovitsch, S.L. Linn, M. Burbank, World Scientific, Singapore 1994, p. 100–110

[7] H. Aiginger, V. Andersen, F. Ballarini, G. Battistoni, M. Campanella, M. Carboni, F. Cerutti, A. Empl, W. Enghardt, A. Fassò, A. Ferrari, E. Gadioli, M.V. Garzelli, K.S. Lee, A. Ottolenghi, K. Parodi, M. Pelliccioni, L. Pinsky, J. Ranft, S. Roesler, P.R. Sala, D. Scannicchio, G. Smirnov, F. Sommerer, T. Wilson and N. Zapp
The FLUKA code: new developments and application to 1 GeV/n Iron beams
Adv. Space Res. **35**, 214–222 (2005)

[8] R.G. Alsmiller Jr., J.M. Barnes, J.D. Drischler
Neutron-photon multigroup cross sections for neutron energies $\leq$ 400 MeV (Revision 1)
Nucl. Instr. Meth. **A249**, 455–460 (1986)

[9] F.S. Alsmiller and R.G. Alsmiller, Jr.
Inclusion of correlations in the empirical selection of intranuclear cascade nucleons from high energy hadron-nucleus collisions
Nucl. Instr. Meth. **A278**, 713–721 (1989)

[10] R.G. Alsmiller, Jr., F.S. Alsmiller and O.W. Hermann
The high-energy transport code HETC88 and comparisons with experimental data
Nucl. Instr. Meth. **A295**, 337–343 (1990)

[11] V. Andersen, F. Ballarini, G. Battistoni, M. Campanella, M. Carboni, F. Cerutti, A. Empl, A. Fassò, A. Ferrari, E. Gadioli, M.V. Garzelli, K. Lee, A. Ottolenghi, M. Pelliccioni, L.S. Pinsky, J. Ranft, S. Roesler, P.R. Sala and T.L. Wilson,
The FLUKA code for space applications: recent developments
Adv. Space Res. 34, 1338–1346 (2004)

[12] M. Antonelli, G. Battistoni, A. Ferrari, P.R. Sala
Study of radiative muon interactions at 300 GeV
Proc. VI Int. Conf. on Calorimetry in High Energy Physics (Calor 96), Frascati (Italy), 8–14 June 1996. Ed. A. Antonelli, S. Bianco, A. Calcaterra, F.L. Fabbri
Frascati Physics Series Vol. VI, p. 561–570 (1997)

[13] F. Atchison
Spallation and fission in heavy metal nuclei under medium energy proton bombardment
Meeting on Targets for neutron beam spallation sources, Ed. G. Bauer, KFA Jülich Germany, **Jül–conf–34** (1980)

[14] F. Ballarini, G. Battistoni, F. Cerutti, A. Empl, A. Fassò, A. Ferrari, E. Gadioli, M.V. Garzelli, A. Ottolenghi, L.S. Pinsky, J. Ranft, S. Roesler, P.R. Sala and G. Smirnov
Nuclear Models in FLUKA: Present Capabilities, Open Problems, and Future Improvements
AIP Conference Proceedings **769**, 1197–1202 (2005)

[15] V.S. Barashenkov, V.D. Toneev
Interactions of High Energy Particles and Nuclei with Nuclei (*in Russian*)
Atomizdat, Moscow (1972)

[16] A. Fassò, A. Ferrari, S. Roesler, J. Ranft, P.R. Sala, G. Battistoni, M. Campanella, F. Cerutti, L. De Biaggi, E. Gadioli, M.V. Garzelli, F. Ballarini, A. Ottolenghi, D. Scannicchio, M. Carboni, M. Pelliccioni, R. Villari, V. Andersen, A. Empl, K. Lee, L. Pinsky, T.N. Wilson and N. Zapp
The FLUKA code: Present applications and future developments
Computing in High Energy and Nuclear Physics 2003 Conference (CHEP2003), La Jolla, CA, USA, March 24–28, 2003, (paper MOMT004), eConf **C0303241** (2003), arXiv:physics/0306162.

[17] T.H. Bauer, R.D. Spital, D.R. Yennie, F.M. Pipkin
The hadronic properties of the photon in high-energy interactions
Rev. Mod. Phys. **50**, 261–436 (1978)

[18] M.J. Berger
Monte Carlo calculation of the penetration and diffusion of fast charged particles
In: B. Alder, S. Fernbach and M. Rotenberg (Eds.), Methods in Computational Physics **1**, 135–215 (1963)

[19] H.A. Bethe
Zur Theorie des Durchgangs schneller Korpuskularstrahlen durch Materie
Ann. Physik **5**, 325–400 (1930)
Selected Works of Hans A. Bethe, World Scientific, Singapore 1996, p. 77–154

[20] H.A. Bethe
Bremsformel für Elektronen relativistischer Geschwindigkeit
Z. Phys. **76**, 293–299 (1932)

[21] H.A. Bethe and W. Heitler
On the stopping of fast particles and on the creation of positive electrons
Proc. Roy. Soc. **A146**, 83–112 (1934)
Selected Works of Hans A. Bethe, World Scientific, Singapore 1996, p. 187–218

[22] H.A. Bethe
Molière's theory of multiple scattering
Phys. Rev. **89**, 1256–1266 (1953)

[23] M. Blann
Hybrid Model for Pre-Equilibrium Decay in Nuclear Reactions
Phys. Rev. Lett. **27**, 337–340 (1971)

[24] M. Blann
Importance of the Nuclear Density Distribution on Pre-equilibrium Decay
Phys. Rev. Lett. **28**, 757–759 (1972)

[25] M. Blann
Preequilibrium decay
Ann. Rev. Nucl. Sci. **25**, 123–165 (1975)

[26] M. Blann and H.K. Vonach
Global test of modified precompound decay models
Phys. Rev. **C28**, 1475–1492 (1983)

[27] M. Blann
Precompound analyses of spectra and yields following nuclear capture of stopped π^-
Phys. Rev. **C28**, 1648–1662 (1983)

[28] M. Blann, H. Gruppelaar, P. Nagel and J. Rodens (eds.)
International Code Comparison for Intermediate Energy Nuclear Data
OECD/NEA Report NSC–DOC94–02 (1994)

[29] F. Bloch
Zur Bremsung rasch bewegter Teilchen beim Durchgang durch Materie
Ann. Phys. **16**, 285–320 (1933)

[30] F. Bloch
Bremsvermögen von Atomen mit mehreren Elektronen
Z. Phys. **81**, 363–376 (1933)

[31] J.B. Birks
The theory and practice of scintillation counting
Pergamon Press, Oxford, 1964

[32] M. Campanella, A. Ferrari, P.R. Sala and S. Vanini,
Reusing Code from FLUKA and GEANT4 geometry
CERN Report ATLAS Internal Note ATL-SOFT 98-039 (1998)

[33] M. Campanella, A. Ferrari, P.R. Sala and S. Vanini,
First Calorimeter Simulation with the FLUGG Prototype
CERN Report ATL-SOFT-99-004 (1999)

[34] A. Capella, U. Sukhatme, J. Tran Thanh Van
Soft multihadron production from partonic structure and fragmentation functions
Z. Phys. **C3**, 329–337 (1980)

[35] A. Capella, J. Tran Thanh Van
A new parton model description of soft hadron-nucleus collisions
Phys. Lett. **B93**, 146–150 (1980)

[36] A. Capella, U. Sukhatme, C.-I. Tan and J. Tran Thanh Van
Dual Parton Model
Phys. Rep. **236**, 225–330 (1994)

[37] L.L. Carter and E.D. Cashwell
Particle-transport simulation with the Monte Carlo method
ERDA Crit. Rev. Ser., National Technical Information Service, Springfield 1975

[38] D. Cavalli, A. Ferrari, P.R. Sala
Simulation of nuclear effects in neutrino interactions
ICARUS–TM-97/18 (1997)

[39] M.B. Chadwick, P. Obložinský, P.E. Hodgson and G. Reffo
Pauli-blocking in the quasideuteron model of photoabsorption
Phys. Rev. **C44**, 814–823 (1991)

[40] P. Cloth, D. Filges, R.D. Neef, G. Sterzenbach, Ch. Reul, T.W. Armstrong, B.L. Colborn, B. Anders and H. Brückmann
HERMES, a Monte Carlo program system for beam-materials interaction studies
Report **KFA/Jül–2203** (1988)

[41] G. Collazuol, A. Ferrari, A. Guglielmi, and P.R. Sala
Hadronic models and experimental data for the neutrino beam production
Nucl. Instr. Meth. **A449**, 609–623 (2000)

[42] E. Cuccoli, A. Ferrari, G.C. Panini
A group library from JEF 1.1 for flux calculations in the LHC machine detectors
JEF–DOC–340 (91) (1991)

[43] L. Dresner
EVAP — A Fortran program for calculating the evaporation of various particles from excited compound nuclei
Oak Ridge National Laboratory report **ORNL–TM–196** (1961)

[44] M.B. Emmett
The MORSE Monte Carlo radiation transport system
Oak Ridge National Laboratory report **ORNL–4972** (1975)
Revision: **ORNL–4972/R1** (1983)
Revision: **ORNL–4972/R2** (1984)

[45] A. Empl, A. Fassò, A. Ferrari, J. Ranft and P.R. Sala,
Progress and Applications of FLUKA
Invited talk at the 12th RPSD Topical Meeting, April 14–18, 2002, Santa Fe, New Mexico, USA ,
electronic proceedings, American Nuclear Society ANS Order No. 700293, ISBN 8-89448-667-5

[46] W.W. Engle, Jr.
A User's Manual for ANISN, A One-Dimensional Discrete Ordinate Transport Code with Anisotropic

Scattering
Oak Ridge Report **K–1693** (1967)

[47] D.E. Cullen, J.H. Hubbell and L. Kissel
EPDL97: the Evaluated Photon Data Library, '97 Version
UCRL–50400, Vol. 6, Rev. 5 (1997)

[48] U. Fano
Inelastic collisions and the Molière theory of multiple scattering
Phys. Rev. **93**, 117–120 (1954)

[49] A. Fassò, G.R. Stevenson
Air activity in the NAHIF complex
CERN Internal Report **HS–RP/IR/78–45** (1978)

[50] A. Fassò
The CERN version of MORSE and its application to strong-attenuation shielding problems
Proc. Topical Conference on Theory and Practices in Radiation Protection and Shielding, Knoxville (Tennessee) 22–24 April 1987, p. 462–471

[51] A. Fassò, A. Ferrari, J. Ranft, P.R. Sala, G.R. Stevenson, J.M. Zazula
FLUKA92
Proc. Workshop on Simulating Accelerator Radiation Environments, Santa Fe (New Mexico) 11–15 January 1993,
Los Alamos report **LA–12835–C** (1994), p. 134–144

[52] A. Fassò, A. Ferrari, J. Ranft, P.R. Sala
FLUKA: present status and future developments
Proc. IV Int. Conf. on Calorimetry in High Energy Physics, La Biodola (Italy) 21–26 September 1993, Ed. A. Menzione and A. Scribano, World Scientific, p. 493–502

[53] A. Fassò, A. Ferrari, J. Ranft, P.R. Sala, G.R. Stevenson, J.M. Zazula
A comparison of FLUKA simulations with measurements of fluence and dose in calorimeter structures
Nucl. Instr. Meth. **A332**, 459–468 (1993)

[54] A. Fassò, A. Ferrari, P.R. Sala
Designing electron accelerator shielding with FLUKA
Proc. of the 8th Int. Conf. on Radiation Shielding, Arlington (Texas) 24–28 April (1994), p. 643–649

[55] A. Fassò, A. Ferrari, J. Ranft and P.R. Sala
FLUKA: performances and applications in the intermediate energy range
Proc. of an AEN/NEA Specialists' Meeting on Shielding Aspects of Accelerators, Targets and Irradiation Facilities, Arlington (Texas) 28–29 April 1994. OECD Documents, Paris 1995, p. 287–304

[56] A. Fassò, A. Ferrari, J. Ranft, P.R. Sala
An update about FLUKA
Proc. 2nd Workshop on Simulating Accelerator Radiation Environments, CERN, Geneva (Switzerland), 9–11 October 1995, Ed. G.R. Stevenson, CERN Report **TIS–RP/97-05**, p. 158–170

[57] A. Fassò, A. Ferrari, J. Ranft, P.R. Sala
New developments in FLUKA modelling hadronic and EM interactions
Proc. 3rd Workshop on Simulating Accelerator Radiation Environments (SARE 3), 7–9 May **1997**, KEK, Tsukuba (Japan). Ed. H. Hirayama
KEK Proceedings 97–5 (1997), p. 32–43

[58] A. Fassò, A. Ferrari, P.R. Scala,
Total giant resonance photonuclear cross sections for light nuclei: a database for the FLUKA Monte Carlo transport code
Proc. 3rd Specialists' Meeting on Shielding Aspects of Accelerators, Targets and Irradiation Facilities (SATIF3), Tohoku University, Sendai, Japan, 12-13 May 1997, OECD-NEA 1998, p. 61

[59] A. Fassò, A. Ferrari, P.R. Sala
Electron-photon transport in FLUKA: status
Proceedings of the Monte Carlo 2000 Conference, Lisbon, October 23–26 2000, A. Kling, F. Barão, M. Nakagawa, L. Távora, P. Vaz eds., Springer-Verlag Berlin, p. 159–164 (2001)

[60] A. Fassò, A. Ferrari, J. Ranft, P.R. Sala
FLUKA: Status and Prospective for Hadronic Applications
Proceedings of the MonteCarlo 2000 Conference, Lisbon, October 23–26 2000, A. Kling, F. Barão,

M. Nakagawa, L. Távora, P. Vaz eds., Springer-Verlag Berlin, p. 955–960 (2001)

[61] A. Fassò, A. Ferrari, P.R. Sala and G. Tsiledakis, "Implementation of Xenon capture gammas in FLUKA for TRD background calculations"
CERN Report ALICE–INT–2001–28 (2001)

[62] A. Fassò, A. Ferrari, S. Roesler, P.R. Sala, G. Battistoni, F. Cerutti, E. Gadioli, M.V. Garzelli, F. Ballarini, A. Ottolenghi, A. Empl and J. Ranft
The physics models of FLUKA: status and recent developments
Computing in High Energy and Nuclear Physics 2003 Conference (CHEP2003), La Jolla, CA, USA, March 24–28, 2003, (paper MOMT005), eConf **C0303241** (2003), arXiv:hep-ph/0306267

[63] I.J. Feng, R.H. Pratt and H.K. Tseng
Positron bremsstrahlung
Phys. Rev. **A24**, 1358–1363 (1981)

[64] A. Ferrari, P.R. Sala, R. Guaraldi, F. Padoani
An improved multiple scattering model for charged particle transport
Presented at the Int. Conf. on Radiation Physics, Dubrovnik, **1991**
Nucl. Instr. Meth. **B71**, 412–426 (1992)

[65] A. Ferrari, P.R. Sala, A. Fassò, G.R. Stevenson
Can we predict radiation levels in calorimeters?
Proc. 2nd Int. Conference on Calorimetry in High-Energy Physics, 14–18 October 1991, Capri (Italy) World Scientific, Singapore 1992, p. 101–116
CERN EAGLE Internal Note **CAL–NO–005** (1991)

[66] A. Ferrari, P.R. Sala
A new model for hadronic interactions at intermediate energies for the FLUKA code
Proc. MC93 Int. Conf. on Monte Carlo Simulation in High Energy and Nuclear Physics, Tallahassee (Florida) 22–26 February 1993. Ed. by P. Dragovitsch, S.L. Linn, M. Burbank, World Scientific, Singapore, 1994, p. 277–288

[67] A. Ferrari, P.R. Sala
Physics of showers induced by accelerator beams
Proc. 1995 "Frédéric Joliot" Summer School in Reactor Physics, 22–30 August **1995**, Cadarache (France). Ed. CEA, Vol. **1**, lecture **5b** (1996)

[68] A. Ferrari, P.R. Sala, J. Ranft, and S. Roesler
The production of residual nuclei in peripheral high energy nucleus–nucleus interactions
Z. Phys. **C71**, 75–86 (1996)

[69] A. Ferrari, P.R. Sala, J. Ranft, and S. Roesler
Cascade particles, nuclear evaporation, and residual nuclei in high energy hadron-nucleus interactions"
Z. Phys. **C70**, 413–426 (1996)

[70] A. Ferrari, and P.R. Sala
The Physics of High Energy Reactions
Proc. of the Workshop on Nuclear Reaction Data and Nuclear Reactors Physics, Design and Safety, International Centre for Theoretical Physics, Miramare-Trieste (Italy) 15 April–17 May 1996, Ed. A. Gandini and G. Reffo, World Scientific, p. 424 (1998)

[71] A. Ferrari, and P.R. Sala
Intermediate and high energy models in FLUKA: improvements, benchmarks and applications
Proc. Int. Conf. on Nuclear Data for Science and Technology, NDST–97, Trieste (Italy), 19–24 May 1997. Ed. G. Reffo, A. Ventura and C. Grandi (Bologna: Italian Phys. Soc.) Vol. **59**, Part **I**, p. 247 (1997)

[72] A. Ferrari, T. Rancati, and P.R. Sala
FLUKA applications in high energy problems: from LHC to ICARUS and atmospheric showers
Proc. 3rd Workshop on Simulating Accelerator Radiation Environments (SARE 3), 7–9 May 1997, KEK, Tsukuba (Japan). Ed. H. Hirayama
KEK Proceedings 97–5 (1997), p. 165–175

[73] A. Ferrari, and P.R. Sala
Treating high energy showers
In: Training course on the use of MCNP in Radiation Protection and Dosimetry, ENEA, Roma (1998), p. 233–264

[74] T.A. Gabriel, J.E. Brau and B.L. Bishop
The Physics of Compensating Calorimetry and the New CALOR89 Code System
Oak Ridge Report **ORNL/TM–11060** (1989)

[75] J.A. Geibel, J. Ranft
Part VI: Monte Carlo calculation of the nucleon meson cascade in shielding materials
Nucl. Instr. Meth. **32**, 65–69 (1965)

[76] K. Goebel, Ed.
Radiation problems encountered in the design of multi-GeV research facilities:

E. Freytag, J. Ranft, "Hadronic and electromagnetic cascades"

K. Goebel, L. Hoffmann, J. Ranft, G.R. Stevenson, "Estimate of the hadronic cascade initiated by high-energy particles"

K. Goebel, J. Ranft, G.R. Stevenson, "Induced radioactivity"

J.H.B. Madsen, M.H. Van de Voorde, J. Ranft, G.B. Stapleton, "Radiation-induced damage to machine components and radiation heating"

J. Ranft, "The interaction of protons in machine components and beam loss distributions"

CERN Yellow Report **71–21** (1971)

[77] K. Goebel, J. Ranft, J.T. Routti, G.R. Stevenson
Estimation of remanent dose rates from induced radioactivity in the SPS ring and target areas
CERN **LABII–RA/73–5** (1973)

[78] W. Guber, J. Nagel, R. Goldstein, P.S. Mettelman, and M.H. Kalos
A geometric description technique suitable for computer analysis of both nuclear and conventional vulnerability of armored military vehicles
Mathematical Applications Group, Inc. Report **MAGI–6701** (1967)

[79] K. Hänssgen, R. Kirschner, J. Ranft, H. Wetzig
Monte Carlo simulation of inelastic hadron-hadron reactions in the medium energy range ($\sqrt{s} \lesssim 3$ GeV). Description of the model used and of the Monte Carlo code HADRIN
University of Leipzig report **KMU–HEP–79–07** (1979)

[80] K. Hänssgen, R. Kirschner, J. Ranft, H. Wetzig
Monte-Carlo simulation of inelastic hadron nucleus reactions. Description of the model and computer code NUCRIN
University of Leipzig report **KMU–HEP–80–07** (1980)

[81] K. Hänssgen, J. Ranft
Hadronic event generation for hadron cascade calculations and detector simulation I. Inelastic hadron nucleon collisions at energies below 5 GeV
Nucl. Sci. Eng. **88**, 537–550 (1984)

[82] K. Hänssgen, H-J. Möhring, J. Ranft
Hadronic event generation for hadron cascade calculations and detector simulation II. Inelastic hadron-nucleus collisions at energies below 5 GeV
Nucl. Sci. Eng. **88**, 551–566 (1984)

[83] K. Hänssgen, S. Ritter
The Monte Carlo code DECAY to simulate the decay of baryon and meson resonances
University of Leipzig report **KMU–HEP–79–14** (1979)
Comp. Phys. Comm. **31**, 411–418 (1984)

[84] K. Hänssgen, J. Ranft
The Monte Carlo code HADRIN to simulate inelastic hadron-nucleon interactions at laboratory energies below 5 GeV
Comp. Phys. Comm. **39**, 37–51 (1986)

[85] K. Hänssgen, J. Ranft
The Monte Carlo code NUCRIN to simulate inelastic hadron-nucleus interactions at laboratory energies below 5 GeV
Comp. Phys. Comm. **39**, 53–70 (1986)

[86] M. Höfert, F. Coninckx, J.M. Hanon, Ch. Steinbach
The prediction of radiation levels from induced radioactivity: discussion of an internal dump target in the PS
CERN Divisional Report **DI/HP/185** (1975)

[87] M. Höfert, A. Bonifas
Measurement odf radiation parameters for the prediction of dose-rates from induced radioactivity
CERN Internal Report **HP–75–148** (1975)

[88] A.V. Ignatyuk, G.N. Smirenkin and A.S. Tishin
Phenomenological Description of the Energy Dependence of the Level Density Parameter
Yad. Fiz. **21**, 485–490 (1975)
Sov. J. Nucl. Phys. **21**, 255–257 (1975)

[89] R. Jaarsma and H. Rief
TIMOC 72 code manual
Ispra report **EUR 5016e** (1973)

[90] F. James
A review of pseudorandom number generators
Comp. Phys. Comm. **60**, 329–344 (1990)

[91] L. Kim, R.H. Pratt, S.M. Seltzer, M.J. Berger
Ratio of positron to electron bremsstrahlung energy loss: an approximate scaling law
Phys. Rev. **A33**, 3002–3009 (1986)

[92] H.W. Koch and J.W. Motz
Bremsstrahlung Cross-Section Formulas and Related Data
Rev. Mod. Phys. **314**, 920–955 (1959)

[93] H. Kowalski, H.-J. Möhring, T. Tymieniecka
High speed Monte Carlo with neutron component — NEUKA
DESY 87–170 (1987)

[94] M. Krell and T.E.O. Ericson
Energy levels and wave functions of pionic atoms
Nucl. Phys. **B11**, 521–550 (1969)

[95] L.D. Landau
On the energy loss of fast particles by ionisation (In Russian)
J. Phys. U.S.S.R., 8, 201 (1944)
Collected papers of L.D. Landau, Pergamon Press, Oxford 1965, p. 417–424

[96] L.D. Landau, I.Ya. Pomeranchuk
The limits of applicability of the theory of bremsstrahlung by electrons and of creation of pairs at large energies (In Russian)
Dokl. Akad. Nauk SSSR **92**, 535 (1953)
Collected papers of L.D. Landau, Pergamon Press, Oxford 1965, p. 586–588

[97] L.D. Landau, I.Ya. Pomeranchuk
Electron-cascade processes at ultra-high energies (In Russian)
Dokl. Akad. Nauk SSSR **92**, 735–738 (1953)
Collected papers of L.D. Landau, Pergamon Press, Oxford 1965, p. 589–593

[98] H. Lichtenstein, M.O. Cohen, H.A. Steinberg, E.S. Troubetzkoy, M. Beer
The SAM-CE Monte Carlo system for radiation transport and criticality calculations in complex configurations (Revision 7.0)
RSIC Computer Code Collection **CCC–187 (1979)–/Lic79**

[99] I. Lux and L. Koblinger
Monte Carlo particle transport methods: neutron and photon calculations
CRC Press, Boca Raton 1991

[100] G. Marsaglia and A. Zaman
Toward a universal random number generator
Florida State University report **FSU–SCRI–87–50** (1987)

[101] G. Marsaglia and A. Zaman
A new class of random number generators
Ann. Appl. Probab. **1**, 462–480 (1991)

[102] G. Marsaglia and W.W Tsang,
The 64-bit universal RNG
Statistics & Probability Letters **66**, 183–187 (2004)

[103] A.B. Migdal

Bremsstrahlung and pair production in condensed media at high energies
Phys. Rev. **103**, 1811–1820 (1956)

[104] A.B. Migdal
Bremsstrahlung and pair production at high energies in condensed media
Zh. Exp. Teor. Fiz. SSSR **32**, 633–646 (1957)
Sov. Phys. JETP **5**, 527–536 (1957)

[105] H.-J. Möhring, J. Ranft and J. Sandberg
FLUKA81 — A new version of the hadron cascade code
CERN **HS–RP/IR/81–55** (1981)

[106] H.-J. Möhring
Hadron-nucleus inelastic cross-sections for use in hadron cascade calculations at high energies
CERN **TIS–RP/116** (1983)

[107] H.-J. Möhring and J. Ranft
Hadronic event generation for hadron cascade calculations and detector simulation III. Application of the model to hadron cascade calculations
Nucl. Sci. Eng. **89**, 247–255 (1985)

[108] H.-J. Möhring, J. Ranft and S. Ritter
Particle production in high energy nucleus-nucleus collisions in dual Monte-Carlo multi-chain fragmentation model
Z. Phys. **C27**, 419–426 (1985)

[109] H.-J. Möhring
On the contribution of electroproduction off nuclei to the generation of energetic hadrons in electromagnetic showers
DESY 89–150 (1989)

[110] H.-J. Möhring and J. Ranft
DTUNUC — Sampling of hadron-nucleus and nucleus-nucleus interactions according to the dual parton model. Version 1.00
University of Leipzig report **UL–HEP 92–02** (1992)

[111] H.-J. Möhring, J. Ranft, A. Capella and J. Tran Thanh Van
Strangeness production in hadron-hadron, hadron-nucleus and nucleus-nucleus collisions in the Dual Parton Model
University of Leipzig report **UL–HEP 92-09** (1992)

[112] H.-J. Möhring, J. Ranft, C. Merino and C. Pajares
String fusion in the Dual Parton Model and the production of antihyperons in heavy ion collisions
University of Leipzig report **UL–HEP 92–10** (1992)

[113] G.Z. Molière
Theorie der Streuung schneller geladener Teilchen I–Einzelstreuung am abgeschirmten Coulomb-Feld
Z. Naturforsch. **2a**, 133–145 (1947)

[114] G.Z. Molière
Theorie der Streuung schneller geladener Teilchen II — Mehrfach und Vielfachstreuung
Z. Naturforsch. **3a**, 78–97 (1948)

[115] G.Z. Molière
Theorie der Streuung schneller geladener Teilchen III — Die Vielfachstreuung von Bahnspuren unter Berücksichtigung der statistischen Kopplung
Z. Naturforsch. **10a**, 177–211 (1955)

[116] W.R. Nelson, H. Hirayama, D.W.O. Rogers
The EGS4 code system
SLAC–265 (1985)

[117] `\url{http://http://t2.lanl.gov/codes/}`

[118] S. Eidelman *et al.* (Particle Data Group)
Review of particle physics
Phys. Lett. **B592**, 1–1109 (2004)
`http://pdg.lbl.gov/`

[119] R.E. Prael and H. Lichtenstein
User Guide to LCS: the LAHET Code System

Los Alamos Report **LA–UR–89–3014** (1989)

[120] R.E. Prael, A. Ferrari, R.K. Tripathi, A. Polanski
Comparison of nucleon cross section parameterization methods for medium and high energies
Proc. 4th Workshop on Simulating Accelerator Radiation Environments (SARE4), 14–16 September 1998, Knoxville (Tenn.), p. 171–181

[121] R.E. Prael, A. Ferrari, R.K. Tripathi, A. Polanski
Plots supplemental to: "Comparison of nucleon cross section parameterization methods for medium and high energies"
Los Alamos report **LA–UR–98–5843** (1998)

[122] S. Qian and A. Van Ginneken
Characteristics of inelastic interactions of high energy hadrons with atomic electrons
Nucl. Instr. Meth. **A256**, 285–296 (1987)

[123] J. Ranft
Monte Carlo calculation of the nucleon-meson cascade in shielding materials initiated by incoming proton beams with energies between 10 and 1000 GeV
CERN Yellow Report **64–47** (1964)

[124] J. Ranft
Improved Monte-Carlo calculations of the nucleon-meson cascade in shielding materials
CERN Report **MPS/Int. MU/EP 66–8** (1966)

[125] J. Ranft
Improved Monte-Carlo calculation of the nucleon-meson cascade in shielding material I. Description of the method of calculation
Nucl. Instr. Meth. **48**, 133–140 (1967)

[126] J. Ranft
Monte Carlo calculation of energy deposition by the nucleon-meson cascade and total-absorption-nuclear-cascade (TANC) counters
Nucl. Instr. Meth. **81**, 29–35 (1970)

[127] J. Ranft, K. Goebel
Estimation of induced radioactivity around high energy accelerators from hadronic cascade star densities obtained from Monte carlo calculations
CERN Internal Report **HP–70–92** (1970)

[128] J. Ranft
The interaction of protons in machine components and beam loss distributions
Chap. II in [76]

[129] J. Ranft
Estimation of radiation problems around high energy accelerators using calculations of the hadronic cascade in matter
Part. Accel. **3**, 129–161 (1972)

[130] J. Ranft, J.T. Routti
Hadronic cascade calculations of angular distributions of integrated secondary particle fluxes from external targets and new empirical formulae describing particle production in proton-nucleus collisions
Part. Accel. **4**, 101–110 (1972)

[131] J. Ranft, J.T. Routti
Monte-Carlo programs for calculating three-dimensional high-energy (50 MeV–500 GeV) hadron cascades in matter
Comp. Phys. Comm. **7**, 327–342 (1974)

[132] J. Ranft, W.R. Nelson
Monte Carlo calculation of photon, electron and positron production from primary proton beams
Nucl. Instr. Meth. **167**, 443–448 (1979)

[133] J. Ranft
Particle production models. Sampling high-energy multiparticle events from inclusive single particle distributions
In: Computer Techniques in Radiation Transport and Dosimetry. Ed. W.R. Nelson, T.M. Jenkins, Plenum Press, (1980), p. 279–310

[134] J. Ranft

The FLUKA and KASPRO hadronic cascade codes
In: Computer Techniques in Radiation Transport and Dosimetry. Ed. W.R. Nelson, T.M. Jenkins, Plenum Press, (1980), p. 339–372

[135] J. Ranft, S. Ritter
Particle production in hadron-nucleus collisions in a multi-chain fragmentation model
Z. Phys. **C20**, 347–355 (1983)

[136] J. Ranft, S. Ritter
The Monte-Carlo codes NUCEVT and HADEVT to simulate hadron production in hadron-nucleus and hadron-hadron collisions
CERN Internal Report **TIS–RP/IR/83–23** (1983)

[137] J. Ranft, S. Ritter
Rapidity ratios, Feynman-x distributions and forward-backward correlations in hadron-nucleus collisions in a dual Monte-Carlo multi-chain fragmentation model
Z. Phys. **C27**, 569–576 (1985)

[138] J. Ranft, S. Ritter
Particle production and correlations in hadron-hadron collisions in the dual Monte-Carlo chain fragmentation model
Z. Phys. **C27**, 413–418 (1985)

[139] J. Ranft, P.A. Aarnio, G.R. Stevenson
FLUKA82
Presented at the Workshop on Shower Simulation for LEP Experiments, CERN, Geneva (Switzerland), 29–31 January 1985
CERN **TIS–RP/156/CF** (1985)

[140] J. Ranft
Transverse energy distributions in nucleus-nucleus collisions in the dual Monte-Carlo multi-chain fragmentation model
Phys. Lett. **B188**, 379–382 (1987)

[141] J. Ranft
The diffractive component of particle production in the dual multistring fragmentation model
Z. Phys. **C33**, 517–523 (1987)

[142] J. Ranft, W.R. Nelson
Hadron cascades induced by electron and photon beams in the GeV energy range
Nucl. Instr. Meth. **A257**, 177–184 (1987)
SLAC–PUB–3959 (1986)

[143] J. Ranft
Hadron production in hadron-nucleus and nucleus-nucleus collisions in the dual Monte Carlo multichain fragmentation model
Phys. Rev. **D37**, 1842–1850 (1988)

[144] J. Ranft
Hadron production in hadron-nucleus and nucleus-nucleus collisions in a dual parton model modified by a formation zone intranuclear cascade
Z. Phys. **C43**, 439–446 (1988)

[145] J. Ranft and S. Roesler
Single diffractive hadron-nucleus interactions within the Dual Parton Model
Z. Phys. **C62**, 329–396 (1994)

[146] J. Ranft
Dual parton model at cosmic ray energies
Phys. Rev. **D51**, 64–84 (1995)

[147] J. Ranft
33 years of high energy radiation Monte Carlo calculations in Europe as seen from CERN
Proc. 2nd Workshop on Simulating Accelerator Radiation Environments (SARE2), CERN, Geneva, 9–11 October 1995
CERN/TIS–RP/97–05 (1997), p. 1–13

[148] S. Ritter, J. Ranft
Simulation of quark jet fragmentation into mesons and baryons on the basis of a chain decay model

University of Leipzig report **KMU–HEP–79–09** 1979
Acta Phys. Pol. **B11**, 259–279 (1980)

[149] S. Ritter
QCD jets in e^+e^-−annihilation and the transition into hadrons
Z. Phys. **C16**, 27–38 (1982)

[150] S. Ritter
Monte-Carlo code BAMJET to simulate the fragmentation of quark and diquark jets
University of Leipzig report **KMU–HEP 83–02** (1983)
Comp. Phys. Comm. **31**, 393–400 (1984)

[151] S. Ritter
Monte-Carlo code PARJET to simulate e^+e^--annihilation events via QCD Jets
Comp. Phys. Comm. **31**, 401–409 (1984)

[152] S. Roesler, R. Engel and J. Ranft
The single diffractive component in hadron-hadron collisions within the two-component Dual Parton Model
Z. Phys. **C59** 481–488 (1993)

[153] S. Roesler, R. Engel and J. Ranft
The Monte Carlo event generator DPMJET-III,
Proc. Monte Carlo 2000 Conference, Lisbon, October 23–26 2000, A. Kling, F. Barão, M. Nakagawa, L. Távora, P. Vaz eds., Springer-Verlag Berlin, p. 1033–1038, 2001.

[154] E. Sartori
Standard Energy Group Structures Of Cross Section Libraries For Reactor Shielding, Reactor Cell and Fusion Neutronics Applications: VITAMIN–J, ECCO–33, ECCO–2000 and XMAS
JEF/DOC–315, Revision 3, 1990

[155] F. Sauter
Über den atomaren Photoeffekt bei grosser Härte der anregenden Strahlung
Ann. Physik **9**, 217–247 (1931)
Über den atomaren Photoeffekt in der K-Schale nach der relativistischen Wellenmechanik Diracs
Ann. Physik **11**, 454–488 (1931).

[156] H. Schönbacher
Short write-up of standard RA-Group Monte Carlo programs available on permanent file on the 7600 computer library
CERN Technical Memorandum **LABII–RA/TM/74–5** (1974)

[157] S.M. Seltzer, M.J. Berger
Bremsstrahlung spectra from electron interactions with screened nuclei and orbital electrons
Nucl. Instr. Meth. **B12**, 95–134 (1985)

[158] S.M. Seltzer, M.J. Berger
Bremsstrahlung spectra from electrons with kinetic energy 1 keV-10 GeV incident on screened nuclei and orbital electrons of neutral atoms with Z = 1–100
At. Data Nucl. Data Tab. **35**, 345–418 (1986)

[159] Qing-biao Shen
Systematics of intermediate energy proton nonelastic and neutron total cross section
IAEA Report **INDC(CPR)–020**, 1991

[160] H. Sorge, H. Stöcker and W. Greiner,
Poincaré invariant Hamiltonian dynamics: Modelling multi-hadronic interactions in a phase space approach
Ann. Phys. **192**, 266–306 (1989)

[161] H. Sorge, H. Stöcker and W. Greiner,
Relativistic quantum molecular dynamics approach to nuclear collisions at ultrarelativistic energies
Nucl. Phys. **A498**, 567c–576 (1989)

[162] H. Sorge,
Flavor production in Pb(160A GeV) on Pb collisions: Effect of color ropes and hadronic rescattering
Phys. Rev. **C52**, 3291–3314 (1995)

[163] R.M. Sternheimer, S.M. Seltzer, M.J. Berger
Density effect for the ionization loss of charged particles in various substances

Phys. Rev. **B26**, 6067–6076 (1982)

[164] R.M. Sternheimer, M.J. Berger, S.M. Seltzer
Density effect for the ionization loss of charged particles in various substances
At. Data Nucl. Data Tab. **30**, 261–271 (1984)

[165] E. Storm, H.I. Israel
Photon cross sections from 1 keV to 100 MeV for elements Z = 1 to Z = 100
At. Data Nucl. Data Tab. **7**, 565–681 (1970)

[166] O.A.P. Tavares and M.L. Terranova
Nuclear photoabsorption by quasi-deuterons and an updated evaluation of Levinger's constant
J. Phys. **G18**, 521–524 (1992)

[167] M.L. Ter-Mikaelyan
Bremsstrahlung radiation spectrum in a medium
Dokl. Akad. Nauk SSSR **94**, 1033 (1954)

[168] M.L. Ter-Mikaelyan
Vliyanie Sredy na Elektromagnitnye Protsessy pri Vysokikh Energiyakh
Izd. Akad. Nauk Arm. SSR, Erevan 1969
English translation: High Energy Electromagnetic Processes in Condensed Media
Wiley & Sons, New York 1972

[169] R.H. Thomas and G.R. Stevenson
Radiological Safety Aspects of the Operation of Proton Accelerators
IAEA Technical Reports Series No. **283**, Vienna 1988

[170] Y.-S. Tsai
Pair production and bremsstrahlung of charged leptons
Rev. Mod. Phys. **46**, 815–851 (1974)
Erratum: Rev. Mod. Phys. 49, 421-423 (1977)

[171] T. Tymieniecka, H.-J. Möhring
NEUKA86 Manual
ZEUS–Note 90–097 (1990)

[172] A. Van Ginneken
CASIM, program to simulate hadronic cascades in bulk matter
Fermilab report **FN–272** (1975)

[173] A. Van Ginneken Energy loss and angular characteristics of high energy electromagnetic processes
Nucl. Instr. Meth. **A251**, 21–39 (1986)

[174] H. Wetzig, K. Hänssgen, J. Ranft
Monte-Carlo simulation of elastic hadron nucleus reactions with the computer code NUCREL
University of Leipzig report **KMU–HEP 81–07** (1981)

[175] M. Zankl and A. Wittmann
The adult male voxel model "Golem" segmented from whole-body CT patient data
Radiat. Environm. Biophys. 40, 153-162 (2001)

[176] J.M. Zazula
Implementation of a low energy neutron transport module into a Monte Carlo hadronic shower code and its applications for accelerator shielding problems
Prog. Nucl. Energy **24**, 385–397 (1990)

[177] J.M. Zazula and K. Tesch
Study of the neutron field from a hadronic cascade in iron: verification of a Monte Carlo calculational model by comparison with measured data
Nucl. Instr. Meth. **A300**, 164–178 (1991)

[178] J.F. Ziegler, H.H. Andersen
The stopping and ranges of ions in matter
Vol. 1–4, Pergamon Press, New York 1977

Index