Forschungszentrum Karlsruhe
in der Helmholtz-Gemeinschaft
Wissenschaftliche Berichte
FZKA 6984

IOTA - a Code to Study Ion Transport and Radiation Damage in Composite Materials

C. H. M. Broeders, A. Yu. Konobeyev, K. Voukelatou
Institut für Reaktorsicherheit

August 2004

Forschungszentrum Karlsruhe

in der Helmholtz-Gemeinschaft

Wissenschaftliche Berichte

FZKA 6984

IOTA - a code to study ion transport and radiation damage in composite materials

C.H.M. Broeders, A. Yu. Konobeyev, K. Voukelatou

Institut für Reaktorsicherheit

Forschungszentrum Karlsruhe GmbH, Karlsruhe

2004

Als Manuskript gedruckt
Für diesen Bericht behalten wir uns alle Rechte vor

Forschungszentrum Karlsruhe GmbH
Postfach 3640, 76021 Karlsruhe

Mitglied der Hermann von Helmholtz-Gemeinschaft
Deutscher Forschungszentren (HGF)

ISSN 0947-8620

urn:nbn:de:0005-069844

IOTA - ein Programm für Untersuchungen von Ionentransport und Strahlenschäden in Verbundwerkstoffen

Zusammenfassung

Der Code IOTA wurde mit dem Ziel entwickelt, primäre Schäden durch Bestrahlung mit Ionen in Verbundwerkstoffen untersuchen zu können.

Der Code kann für die Berechnung der Gesamtzahl der primären Schäden in den Materialien, für Displacement-Wirkungsquerschnitte und für die räumlichen Verteilungen der Strahlenschäden benützt werden. Die Berechnungen für den Transfer der kinetischen Energie des bewegenden Teilchens zum Atom im Gitter, basieren auf der Anwendung von tabellierten Daten oder auf analytischen Formeln für die differentiellen Wirkungsquerschnitte. Die Simulation wurde mit der „binary collision approximation“ (BCA) und mit dem Monte Carlo Verfahren realisiert.

Unter Verwendung von Rechnungsergebnissen nach der Methode der Molekular Dynamik (MD) kann der IOTA Code auch für hochenergetische Rechnungen in gekoppelten BCA-MD Rechnungen angewandt werden.

Abstract

The code IOTA has been elaborated to study primary radiation defects in composite materials irradiated by ions.

The code may be used for the calculation of the total number of primary defects created in materials, for the displacement cross-section and for the spatial defect distribution. The calculations are based on the use of tabulated data or of analytical expressions for the differential cross-section for the transfer of the kinetic energy from the moving ion to a lattice atom. The simulation is based on the binary collision approximation and the Monte Carlo method.

Using the results obtained with the help of the method of molecular dynamics (MD), the IOTA code may also be applied for high energy calculations in joint BCA-MD calculations.

Content

The code IOTA (IOn TrAnsport) has been created for the investigation of the primary radiation defects in composite materials irradiated by ions.

The code calculates the total number of primary defects created in materials, the displacement cross-section and the spatial defect distribution. The calculations are performed with the help of the different approaches described below. The simulation of the ion movement in the media is based on the binary collision approximation (BCA) and the Monte Carlo method.

The IOTA code can be applied for high energy calculations using the results obtained with the help of the method of molecular dynamics (joint BCA-MD calculations).

History. The creation of the IOTA code was initiated in January, 2003 and the first version of the code was approved in April, 2003. The current version is realized in December, 2003.

The service code SL_LNS4A used for the input data preparation was written in 2000. Now it was checked and rewritten.

1. *Specific features of the code*

The IOTA code can be applied for composite materials having up to 21 different components. The code is based on the use of any justified and approved differential cross-section $d\sigma(E,T)/dT$ [1-3] for the transfer of the kinetic energy T from the moving ion to the lattice atom.

The IOTA code does not repeat any parts and algorithms of the popular codes based on the BCA approach [4-7].

The advantages of the IOTA code are

i) the possibility to simulate the damage process based on the different ap proaches with the help of the Monte Carlo method, and by the numerical inte gration of the corresponding distributions in a single computer run;

ii) the algorithm allows easily incorporation of the results obtained by the MD method and to perform combined BCA-MD calculations;

iii) the faster running time comparing with the other BCA codes.

2. *Brief description of the method of calculations*

For the incident ion with kinetic energy E the radiation damage dose rate is defined by the particle flux and the atomic displacement cross-section $\sigma_d(E)$. This cross-section is calculated as follows

$$\sigma_d(E) = \int \frac{d\sigma(E,T)}{dT} \nu(T) dT \,, \qquad (1)$$

where $d\sigma(E,T)/dT$ is the differential cross-sections for the transfer of a kinetic energy T from the incident ion to a primary knocked-on atom (PKA), $\nu(T)$ is the cascade function equal to the total number of displaced atoms produced by the PKA of kinetic energy T.

The IOTA code calculates the value of the cascade function $\nu(T)$, the space distribution of the defects created and the displacement cross-section $\sigma_d(E)$.

A single computer run for the incident energy E gives the $\nu(T)$ values for the whole energy range below this energy.

2.1. Monte Carlo simulation

The simulation of the ion transport can be described by the following scheme.

1. Firstly the mean free path L of the primary ion is calculated with the help of the expression

$$L = \frac{1}{\sigma_{int}(E) N_0} \,, \qquad (2)$$

where E is the kinetic energy of the projectile, N_0 is the number density of atoms in a material.

The cross-section $\sigma_{int}(E)$ in Eq.(2) corresponds to the interaction of the primary ion with the lattice atom and the transfer to it of the portion of the kinetic energy exceeding the threshold displacement energy E_d. It is calculated as follows

$$\sigma_{int}(E) = \sum_{i=1}^{N} w_i \, \sigma_{int}^{(i)}(E) = \sum_{i=1}^{N} w_i \int_{E_d^{(i)}}^{T_{max}^{(i)}} d\sigma_i(E,T) \,, \qquad (3)$$

where w_i is the atomic fraction of the i-th component of the target material, $E_d^{(i)}$ is the effective threshold displacement energy for i-th component, $T_{max}^{(i)}$ is the maximal energy transferred from the ion to the PKA defined by the kinematics.

In the present version of the IOTA code the differential cross-section for the transfer of the kinetic energy T from the primary ion to the lattice atom dσ(E,T) is calculated as follows

$$d\sigma_i(E,T) = \pi a^2 f(t^{1/2}) \frac{dt}{2t^{3/2}}, \tag{4}$$

where the function f($t^{1/2}$) and the screening length “a” are calculated based on the different approaches depending on the user choice by input.

i) The f($t^{1/2}$) function and screening length are defined according to Ref.[2] (the IOTA code parameter IDSDT on Card 2 with valid values 0, 10-15 from Table 1)

$$f(t^{1/2}) = \lambda t^{1/2-m}\left[1+(2\lambda t^{1-m})^q\right]^{-1/q},$$

$$a = a_0(9\pi^2/128)^{1/3}\left(Z_1^{2/3}+Z_2^{2/3}\right)^{-1/2},$$

where the corresponding available sets of the λ, m and q parameters are listed in Table 1.

Table 1

Sets of the λ, m and q parameters for the f($t^{1/2}$) function calculation and the IOTA code parameter IDSDT from Ref.[2]

λ	M	q	Reference	IDSDT
1.309	1/3	2/3	[2]	0 or 10
1.7	0.311	0.588	[13]	11
2.37	0.103	0.570	[13]	12
2.92	0.191	0.512	[13]	13
3.07	0.216	0.530	[13]	14
3.35	0.2328	0.4445	[14]	15

ii) The f($t^{1/2}$) function and ‘a” are calculated according to Ref.[3]

(IDSDT =1 on Card 2)

$$f(t^{1/2}) = a_1 t^{j/4} \frac{x + a_2}{(x + a_3)^2}$$

j=1, a_1=0.088, a_2=1, a_3=0.2 for $t^{1/2} \le 0.04$

j=1, a_1=0.23, a_2=1.3, a_3=0.4 for $0.04 < t^{1/2} \le 1$

j=0, a_1=0.4563, a_2=0, a_3=0.3 for $1 < t^{1/2} \le 40$

j=0, a_1=0.5, a_2=0, a_3=0 for $t^{1/2} > 40$

$$a = a_0 (9\pi^2/128)^{1/3} \left(Z_1^{1/2} + Z_2^{1/2}\right)^{-2/3},$$

For both cases i) and ii)

$$t = \varepsilon T / \alpha^2 \varepsilon_0,$$

$$\varepsilon = E/\varepsilon_0 = \frac{a A_2 E}{Z_1 Z_2 e^2 (A_1 + A_2)},$$

$$\alpha^2 = \frac{4 A_1 A_2}{(A_1 + A_2)^2}$$

where Z_1, A_1 and Z_2, A_2 are the atomic number and mass number of the projectile and the target material, respectively, a_0 is the Bohr radius.

The values of the $\sigma_{int}(E)$ cross-section are calculated by the SL_LNS4A code before the IOTA code run and stored on special files.

2. The trajectory of the ion is divided in small steps. The step Δl is defined as a minimal value from the part of the mean free path L/n and the distance δx, where the ion energy loss is less than the part of the ion kinetic energy γE. The values of parameters “n” and γ adopted in the current version of the code are equal to 10 and 0.02 correspondingly. The value of δx is obtained from the total stopping power.

3. For the defined Δl value the elastic interaction of primary ion with a lattice atom is simulated by Monte Carlo. The elastic interaction corresponds to the transfer of the energy to PKA more than the threshold displacement energy E_d.

4. If the elastic event occurs, the type of the PKA from the composite material is defined by the stochastic method based on $w_i\sigma^{(i)}_{int}$ values (Eq.(3)). Than the kinetic energy T of the PKA is obtained with the help of the $d\sigma(E,T)$ distribution (Eq.(4) by the Monte Carlo “rejection” method.

5. The primary ion passes the distance Δl and losses the energy transferred to the PKA and for the electronic excitation. The last energy is defined from the electronic stopping power $(dE/dx)_{el}$.

The PKA characteristics are stored for the further treatment.

The new mean free path is defined for the primary ion and the procedure described above is repeated.

6. If the elastic collision with the energy transfer exceeding E_d does not occur, the primary ion moves on the distance Δl. Its energy is decreased corresponding to the experimental electronic stopping power $(dE/dx)_{el}$ and the part of the nuclear stopping power $(dE/dx)_n^{0-E_d}$, which corresponds to the energy transfer from the ion to the lattice atom from 0 to E_d value. The data $(dE/dx)_n^{0-E_d}$ are prepared by the SL_LNS4A code before the IOTA code run.

7. The primary ion is tracing while its kinetic energy is less than the minimal effective threshold displacement energy $E_d^{(i)}$ from all N-components of the composite material.

8. The stored PKA's are treated as primary ions producing new knock-on atoms and procedure described above repeats.

9. The process described above is repeated until the energy of all PKA's created is less than the minimal E_d energy and the computer bank of PKA's is empty.

2.2. *Integration*

Along with the Monte Carlo simulation the IOTA code calculates the number of the primary defects $\nu(E)$ by the numerical integration of the ratio of the losses for damage production to the total stopping power. The total number of Frenkel pairs produced by the ion with the energy E_0 in the material is calculated using the following expression [1]

$$\nu(E_0) = \frac{0.8}{2E_d}\tilde{T}(E_0) = \frac{0.8}{2E_d}\int_0^{E_0}\frac{(dE/dx)_{dam}}{(dE/dx)_{el} + (dE/dx)_n}dE \ , \qquad (5)$$

where E_0 is the primary ion energy, $\tilde{T}$ is the energy going to the production of displaced atoms.

The value $(dE/dx)_{dam}$ in Eq.(5) is the specific energy loss for the damage production, $(dE/dx)_{el}$ is the electronic stopping power and $(dE/dx)_n$ is the specific energy loss for the elastic scattering (nuclear loss).

The specific energy loss for the damage production is calculated as follows

$$\left(\frac{dE}{dx}\right)_{dam} = N_0 \int_{E_d}^{T_{max}} \frac{d\sigma(E,T)}{dT} \tilde{T}(T)\, dT\,, \tag{6}$$

where N_0 is the number density of atoms in a material, $\tilde{T}$ (T) is the energy for defects production, dσ(E,T) is calculated by Eq.(4).

The energy for the damage production $\tilde{T}$(T) is calculated according to the NRT-approach [4,8]

$$\tilde{T}_{NRT}(T) = \frac{T}{1 + k\, g(\varepsilon)} \quad \text{(keV)}, \tag{7}$$

$g(\varepsilon)=3.4008\ \varepsilon^{1/6} + 0.40244\ \varepsilon^{3/4} + \varepsilon$,

$$k = \frac{32}{3\pi}\left(\frac{m_e}{M_2}\right)^{1/2} \frac{(A_1 + A_2)^{3/2} Z_1^{2/3} Z_2^{1/2}}{A_1^{3/2} (Z_1^{2/3} + Z_2^{2/3})^{3/4}}\,,$$

$$\varepsilon = \left[A_2 T / (A_1 + A_2)\right]\left[a / (Z_1 Z_2 e^2)\right],$$

where m_e is the mass of an electron, M_2 is the mass of the atom of target material and the other values are defined above, e is the electron charge.

Note, that the value of "k" in Eq.(7) is defined according to Robinson [4] and for the case of Z_1=Z_2, A_1=A_2 it coincides with the NRT-value [8] equal to $k = 0.1337\ Z_1^{1/6}\ (Z_1 / A_1)^{1/2}$.

2.3. Simple evaluation

For the comparison with other calculations, the value of the cascade function ν(E) is evaluated with the help of the NRT-formula [8]

$$\nu(E)_{NRT} = \frac{0.8}{2E_d}\ \tilde{T}_{NRT}(E)\,, \tag{8}$$

where $\tilde{T}_{NRT}$ (E) is calculated by Eq.(7).

2.4. Combined BCA-MD calculation

The simulation described in Sect.2.1 is performed up to a critical ion energy T_{crit}, below this energy the results of MD calculations are used to obtain the total number of defects. This energy is applied as for the primary ion, as for PKA's produced in the collision cascade with the energy above T_{crit}.

The typical value of T_{crit} is between 5 keV and 100 keV depending on the MD simulation.

The results of the MD calculations are introduced in the code in the form of the defect production efficiency. The efficiency is equal to the ratio of the total number of survived point defects (Frenkel pairs) to the number of defects calculated by the NRT model (Eq.(8)).

The general expression for the defect production efficiency is as follows

$$\eta(E_{MD}) = \alpha_1 E_{MD}^{\alpha_2} + \alpha_3 E_{MD} \,, \tag{9}$$

where the energy E_{MD} is supposed equal to the $\tilde{T}_{NRT}$ energy in Eq.(7), α_i are fitting coefficients.

To perform the calculation the user should supply the values of the α_i parameters in Eq.(9) or use the default values which are available in the code for a limited number of elements (Fe, Cu, W).

2.5. Flowchart of the code

The flowchart of the code is shown in Fig.1

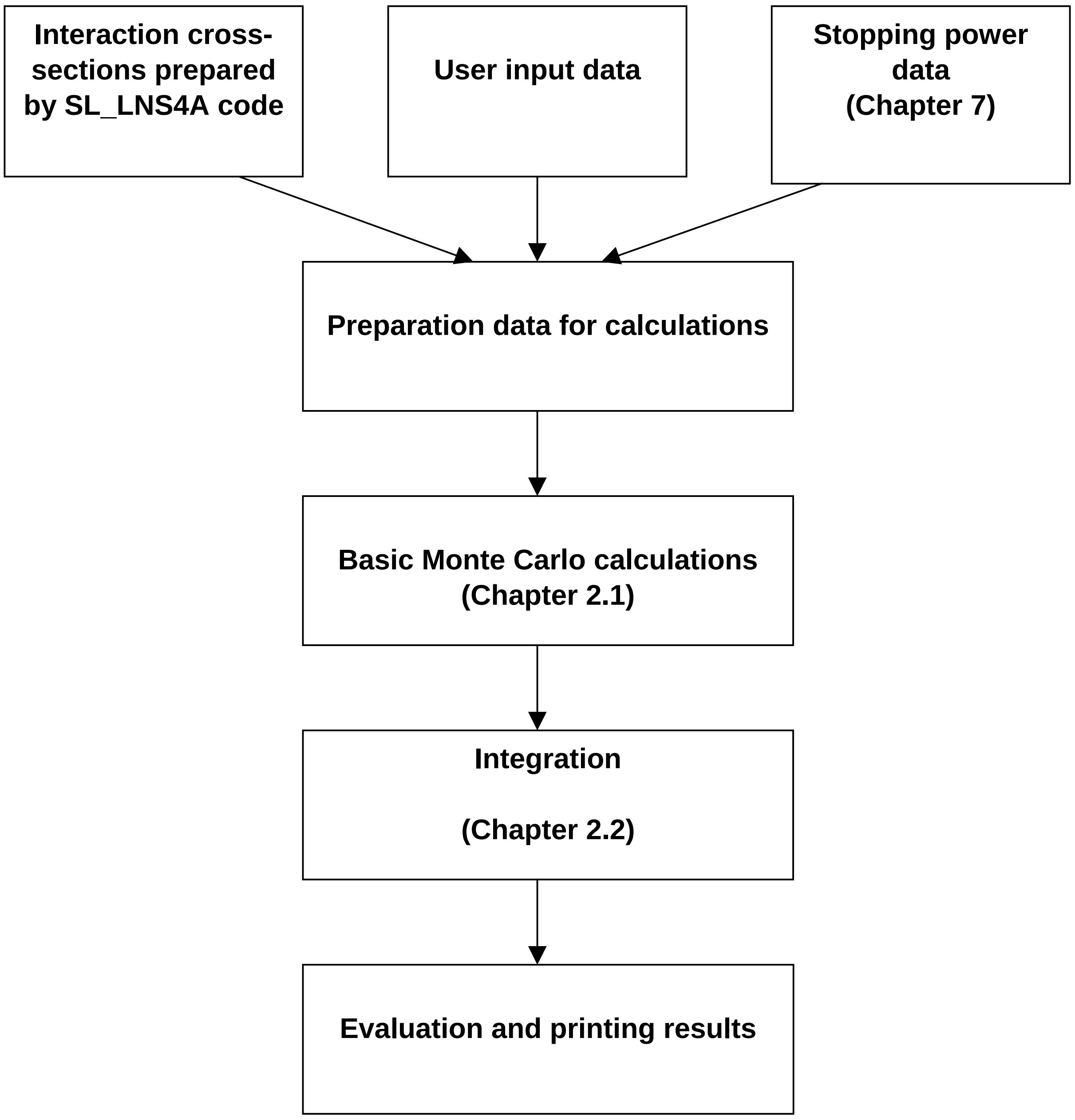
Interaction cross-sections prepared by SL_LNS4A code
User input data
Stopping power data (Chapter 7)
Preparation data for calculations
Basic Monte Carlo calculations (Chapter 2.1)
Integration (Chapter 2.2)
Evaluation and printing results

Fig.1

3. *Computer compatibility*

The IOTA code and the service code SL_LNS4A are written in FORTRAN.

The IOTA code uses a random number generator, which can be different for different FORTRAN compilers. The current version of the IOTA code uses the GNU FORTRAN [9] built-in random number generator RAND. To substitute it by the other one it is necessary to check the FUNCTION RANDOM in the IOTA code.

4. *Execution of the IOTA code*

The execution of the IOTA code includes several steps.

1. The preparation of the main input file. The name is INPUT. The procedure is described below in Sect.5.

2. The stopping power calculation by the SRIM code [7] if the option 'SRIM' is chosen in the input file (Sect.5). The details are discussed in Sect.7.

If the 'SPAR' option is selected in INPUT, the stopping power is calculated by the IOTA code and point 2. is omitted.

3. Run SL_LNS4A code. The input file is the same as for the IOTA code (the file INPUT). After the SL_LNS4A code execution the files with the information needed for the IOTA code run are created. After the file creation it is not necessary to execute this code again for the other IOTA code runs with the same INPUT and other set of the primary ion energies, which do not exceed the TLIM value (Sect.5).

4. Run the IOTA code. Analysis of the results.

5. *Preparation of the main input file for the IOTA code*

The main input file is called INPUT. It is located in the same directory with the executable modules of the IOTA code and the SL_LNS4A code.

The file INPUT

All data are introduced in the file in the free format form including the names of the external files. Each line beginning from the symbol ‘*’, ‘c’ or ‘!’ defines the comment card. The length of the external file names should not exceed 12 characters.

The structure of the INPUT file is described below.

(***Card 1***) The title of the task described in 80 characters including blank spaces.

(***Card 2***) Parameters IGRAPH, MAXPLAY, TLIM, IDSDT.

Parameter IGRAPH is the damage profile printing option.

The value IGRAPH = 0 suppresses the printing of the damage profiles (space damage distribution).

IGRAPH = 1 means the printing.

The parameter MAXPLAY defines the maximal number of attempts to get the kinetic energy T of PKA from the dσ(E,T)/dT distribution in the FUNCTION TPKA. The play of T value is performing by the Monte Carlo “rejection” method. If after MAXPLAY attempts the definition of T failed, the kinetic energy of PKA is taken according to the ratio

$$T_{mean} = \int_{E_d}^{T_{max}} \frac{d\sigma(E,T')}{dT'} T'dT' \Bigg/ \int_{E_d}^{T_{max}} d\sigma(E,T') \;, \qquad (10)$$

the T_{mean} values are calculated by the SL_LNS4A code before the IOTA code run.

The possible values of MAXPLAY are

i)actual number of attempts

ii) MAXPLAY=1. In this case the kinetic energy T of PKA is substituted by T_{mean} from Eq.(10). It greatly reduces the time of the calculation at the high ion energies, but should be referred to a approximate calculations

iii) MAXPLAY=0. It means the use of the default value of MAXPLAY equal to 250,000

iv) MAXPLAY < 0 means the unlimited number of attempts. It incredibly increases the running time of the code at the high energies of the primary ions.

The value of TLIM defines the maximal energy (MeV) to be treated by the SL_LNS4A code. It should exceed the maximal energy of the primary ion introduced in the IOTA code input file (see below). The default value (TLIM=0) is equal to 5 GeV.

Parameter IDSDT defines the method of the $f(t^{1/2})$ function calculation (Eq.(4)). The values IDSDT=0, 10-15 correspond to the $f(t^{1/2})$ function from Ref.[2] (see Table 1 for IDSDT definition). The value IDSDT=1 means the calculation of $f(t^{1/2})$ with the help of the approach from Ref.[3].

(***Special cards 1' and 2'***) These cards are used only for the combined BCA-MD calculations.

[***Card 1'***] To initiate the calculations the card should contain the words 'BCA' and 'MD' in any combination printed in the same line (See Example 3)

[***Card 2'***] CMD(1), CMD(2), CMD(3), ELOW, EHIGH, ECRIT

The CMD(i) values correspond to the coefficients α_i in Eq.(9) for the efficiency of the defect production obtained from the MD calculations.

ELOW and EHIGH are the lowest and the highest E_{MD} (or $\tilde{T}_{NRT}$) energy in keV, where the efficiency supposed to be a constant, $\eta(E_{MD} < ELOW) = \eta(ELOW)$, $\eta(E_{MD} > EHIGH) = \eta(EHIGH)$.

The energy ECRIT (keV) is equal to the value of the E_{MD} energy in Eq.(9) where the MD results for the defect production are used instead of the BCA calculations (Sect.2.4). Usually, ECRIT is equal to EHIGH. The critical energy T_{crit} mentioned in Sect.2.4 is calculated by the code basing on the ECRIT value for the incident ion and the ions of the target material.

The default values are taken if CMD(i),i=1,3; ELOW; EHIGH and ECRIT are set to zero. They are available for Fe, Cu and W in the current code version

(***Card 3***) The atomic number (Z) and the atomic weight (A) (amu) of the projectile.

If A=0 the atomic weight is taken for the natural mixture of isotopes for this element (Z) from internal code tables.

(***Card 4***) The density of the target composite material (RO), the number of the components of the compound (NK) and the stopping power option (SO).

The RO value is introduced in g/cm^3. If RO=0, the density is taken from the internal code table for this material.

The NK value is equal to the number of unique atoms of the target. For example, the NK value is equal to 2 for Li_2O, and equal to 3 for $Al_2(SO_4)_3$. It is for the case if only the single isotope of each element is considered in the calculation. For the mixture of the isotopes the NK value should include their amount. If the lithium from the Li_2O compound is considered as a mixture of ^6Li and ^7Li isotopes, the NK value is equal to 3.

The stopping power option (SO) is a character variable describing the method of the stopping power calculation. It can be equal as follows

i) ‘SRIM’ (or ‘Ziegler’). The data for the stopping power are taken from the special files prepared by the SRIM code [7]. (See *Card 6 ,9* and Sect.7*).* This option is recommended.

ii) ‘SPAR’ (or ‘Armstrong’). The stopping power is calculated in the IOTA code by the SPAR routine [15,16]. It should be noted that for the heavy ions and high energies the accuracy of this method is less than of the SRIM code.

(***Card 5,** 6 or more if necessary*) Names of files with the cross-sections and other information to be prepared by the SL_LNS4A code. Each name corresponds to the interaction of the primary ion with the certain material component. For example, for an α-particle projectile and Li_2O target the names are introduced for α+O and α+Li interactions[1].

The sequence of the file names should be the same as for the description of the components of the composite material (*Card 7* and others). The names are separated by at least one blank space. Any number of cards can be used for the total list of the file names.

(***Card 6**[2]*) Name of file with stopping power data corresponding to the primary ion interaction with the compound (e.g. with Li_2O). The format of the file corresponds to the SRIM code output [7] (see Sect. 7).

If the ‘SPAR’ option is selected for the stopping power calculation (*Card 4*) the *Card 6* should be omitted.

(***Card 7***) The atomic number (Z), the atomic weight (A) (amu), the atomic concentration (W) and the effective threshold displacement energy (ED) for the first component of the composite material (target).

If the zero value is introduced instead of A value (A=0), the atomic weight is taken for the natural mixture of isotopes with the atomic number Z from the internal code table.

The atomic concentration (W) is introduced in the normalized or the unnormalized form. For example, in the description of the oxygen from Li_2O compound both values 0.333333 or 1 are correct.

The displacement energy (ED) is introduced in eV. The default value (ED=0) is equal to 40 eV.

(***Card 8,** 9 or more if necessary*) Names of files with the data to be prepared by the SL_LNS4A code for the interaction of the ion described in *Card 7* with each component of the composite material. For example, if the *Card 7* describes the oxygen from

[1] it is supposed that Li_2O consists of single Li and single O isotope.

[2] the number depends on the amount of cards used to describe the names of files prepared by the SL_LNS4A code (see *Card number 5,6 or more*)

Li_2O, the current card should include the file names for the O+O and O+Li interactions.

Note, the sequence of such file names should coincide with the sequence of the different material components description (given in the cards similar to the *Card 7*).

(***Card 9***) Name of file with the stopping power data from the SRIM code corresponding to the interaction of the ion describing in the *Card 7* with the compound. For example, if the *Card 7* describes the oxygen from Li_2O compound, the name of file given here is for the stopping power data for the O+Li_2O interaction.

This card is omitted for the ‘SPAR’ option chosen for the stopping power calculation in *Card 4*.

(***Next cards similar to Cards 7-9***). The same for other components of the compound. For example, if the oxygen from Li_2O is described in Cards 7-9 these cards are used for the lithium description.

(Card after the material description). Names of the output files.

Two file names are introduced for the usual calculations and three file names for the joint BCA-MD calculations.

The first output file contains the detail information about the calculation performed. The second file repeats partly the data from the main output for the special use, e.g. for the graphics representation. The third file contains the results of the combined BCA-MD simulation.

(***Next cards***). The energy of primary ion (E) and the number of Monte Carlo events to perform the calculation at this energy (NHIST).

The energy is introduced on the separate cards following one after another in MeV along with the unique NHIST number.

The negative value shown in the list is considered as the last energy in the simulation. For example, to perform the calculation for incident energy equal to 10 keV with 999 Monte Carlo events and the primary energy equal to 1 GeV with 100 Monte Carlo events, one should introduce two successive cards 0.010 999 and –1000.0 100

6. *Examples of the INPUT file*

A) Use the ‘SRIM’ option for the stopping power calculation

The example of the INPUT file for the interaction of ^{28}Si with the silicon carbide is given below (Example 1)

Example 1

The INPUT file for Si+SiC interactions with the stopping power from SRIM code

```
* IOTA code input
* name of the task
                 Test calculations    Si+Silicon Carbide
 1  0  10.  0
* Z, A of the projectile ion
 14  28
* density (g/cm3), number of components and stopping option
  3.21    2   SRIM
* name of files with the cross-sections from SL_LNS4A code
   Si_Si.dat    Si_C.dat
* dE/dx for projectile + composite media
   Si_SiC.Zie
*
*  description of the components for compound
* Z,  A,  atomic fraction, Ed (eV)
 14  28   0.5000   40.0
* cross-sections
  Si_Si.dat   Si_C.dat
*   dE/dx
  Si_SiC.Zie
  6  12   0.5000   20.0
  C_Si.dat    C_C.dat
  C_SiC.Zie
*
* names of output files
   messages  mess2.dat
*
*  energy of projectile and number of Monte Carlo events
0.010  1000
0.1    1000
-1.0    1000
```

B) Use the ‘SPAR’ option for the stopping power calculation

The example of the INPUT file is given here (Example 2).

Example 2

The INPUT file for Si+SiC with the stopping power from the SPAR calculations

```
* IOTA code input
* name of the task
                 Test calculations    Si+Silicon Carbide
*
 1  0  10.  0
* Z, A of the projectile ion
 14  28
* density (g/cm3), number of components and stopping option
  3.21    2   SPAR
*
* name of files with the cross-sections from SL_LNS4A code
   Si_Si.dat    Si_C.dat
*
*  description of the components for compound
* Z,  A,  atomic fraction, Ed (eV)
 14  28   0.5000   40.0
* cross-sections
  Si_Si.dat   Si_C.dat
*
  6  12   0.5000   20.0
  C_Si.dat    C_C.dat
*
* names of output files
   messages  mess2.dat
*
*  energy of projectile and number of Monte Carlo events
0.010  1000
0.1    1000
-1.0    1000
```

C) Combined BCA-MD calculations

The example of the INPUT file for the irradiation of natural tungsten by ^{167}Tm ions is given below (Example 3). The stopping power is obtained by the SRIM code. The default parameters are used for the efficiency calculations at low energies.

Example 3

The INPUT file for joint BCA-MD calculation for Tm+W interaction

```
* IOTA code input
* name of the task
                 Test calculations    Tm+W
 0 700000 500.1  0
*
   BCA + MD
 0  0  0  0  0  0
* Z, A of the projectile ion
 69.  167.
* density (g/cm3), number of components and stopping option
  0    1    Ziegler
* name of files with the cross-sections from SL_LNS4A code
   tm_w.dat
* dE/dx for projectile + media
   tm_w.zie
*
* Z,  A,  atomic fraction, Ed (eV)
 74  0   1.0000   90.0
* cross-sections
  w_w.dat
* <  dE/dx   >
  w_w.zie
*
* names of output files
   messages  mess2.dat  bcamd.out
*
*  energy of projectile and number of Monte Carlo events
0.01   100
1.0    100
 10.   100
100.   50
-500.  50
*End
```

7. Preparation of the data for the stopping power

This section discusses the preparation of the stopping power data with the help of the SRIM code. It corresponds to the recommended input option ‘SRIM’ or ‘Ziegler’ (see Sect.5, *Card 4*) for the stopping power calculation.

If the ‘SPAR’ option is chosen in *Card 4* of the INPUT file this section should be omitted.

The current version of the IOTA code works with the output format of the SRIM code [7]. The code is distributed by the author [7] freely.

To use the SRIM code one should download it from Ref.[7] and install it on PC.

If the SRIM code is installed, the instruction to prepare the data for the stopping power suitable for the IOTA code calculation is the following.

1. Select “Stopping/Range Tables” in the main SRIM menu (general panel).

2. Select the appropriate projectile and the target material. In the case of the compound one should introduce the correct density of the compound. It is the same as in the *Card 4* of the INPUT file of the IOTA code (see Sect.5).

3. Set the energy range of the projectile from 1 eV to 5.2 GeV (the maximal energy allowable in the IOTA code).

4. Select the units for the stopping power equal to MeV/(mg/cm^2)) (default in the main SRIM menu).

5. Perform the operation “Calculate table” and give the same name for the output file as shown in the file INPUT for the IOTA code (Sect.5). Store the output file in the same directory where there are the executable modules of the SL_LNS4A code and the IOTA code.

6. Repeat this procedure for all atoms of the compound, considered as a projectile.

The example of the output of the SRIM code for the case of Si+SiC interaction is given below (Example 4).

Note. If the calculations are performed for the same material and the various projectiles with the equal atomic number and the different atomic weights it is not necessary to prepare the SRIM stopping power data for each isotope.

If the data are prepared for the ion with Z and A and the actual projectile in the task is Z and A’, the electronic stopping power is correctly recalculated for a new energy grid (A’/A)E. The nuclear stopping power is renormalized according to the following formula

$$\frac{(dE/dx)_n^{Z,A'}}{(dE/dx)_n^{Z,A}} = \left(\frac{A'}{A}\right) \frac{-\sqrt{t'}/\sqrt{1+t'} + \ln(\sqrt{t'}+\sqrt{1+t'})}{-\sqrt{t}/\sqrt{1+t} + \ln(\sqrt{t}+\sqrt{1+t})} \,,$$

$$t' = 1.948\cdot 10^4 \left(\frac{E\,A_T}{Z\,Z_T(A'+A_T)(Z^{2/3}+Z_T^{2/3})^{1/2}}\right)^{8/9} ,$$

$$t = 1.948\cdot 10^4 \left(\frac{E\,A_T}{Z\,Z_T(A+A_T)(Z^{2/3}+Z_T^{2/3})^{1/2}}\right)^{8/9} ,$$

where E is the energy of projectile (MeV), Z_T and A_T are the atomic number and the atomic weight of the target material.

The formula shown above is obtained from the analytical integration of $T\cdot d\sigma(E,T)/dT$, where $d\sigma(E,T)$ is calculated by Eq.(4) and the $f(t^{1/2})$ function is defined by the λ, m and q parameters from the first line of Table 1.

New data for the stopping power are written in the output file RENORM.STO.

Example 4

<u>The SRIM code output for the S+SiC interaction</u>

```
==================================================================
              Calculation using SRIM-2003
              SRIM version ---> SRIM-2003.10
              Calc. date   ---> March 24, 2003
 ==================================================================

 Disk File Name = SRIM Outputs\Si_SiC.Zie

 Ion = Silicon [14] , Mass = 27.977 amu

 Target Density =  3.2100E+00 g/cm3 = 9.6419E+22 atoms/cm3
 ======= Target  Composition ========
    Atom   Atom   Atomic    Mass
    Name   Numb   Percent   Percent
    ----   ----   -------   -------
     Si     14    050.00    070.05
      C      6    050.00    029.95
 ====================================
 Bragg Correction = 0.00%
 Stopping Units =  MeV / (mg/cm2)
 See bottom of Table for other Stopping units

   Ion        dE/dx      dE/dx     Projected  Longitudinal   Lateral
  Energy      Elec.      Nuclear     Range     Straggling   Straggling
-----------  ---------- ---------- ----------  ----------  ----------
    1.1 eV    6.406E-03  3.765E-02       1 A         1 A           A
    1.2 eV    6.406E-03  3.986E-02       1 A         1 A           A
    1.3 eV    6.667E-03  4.201E-02       1 A         1 A           A
    1.4 eV    6.919E-03  4.408E-02       1 A         1 A           A
    1.5 eV    7.162E-03  4.611E-02       1 A         1 A           A
    1.6 eV    7.397E-03  4.808E-02       1 A         1 A           A
    1.7 eV    7.624E-03  5.000E-02       1 A         1 A           A
    1.8 eV    7.845E-03  5.187E-02       1 A         1 A           A
      2 eV    8.270E-03  5.551E-02       1 A         1 A         1 A
   2.25 eV    8.771E-03  5.985E-02       1 A         1 A         1 A
2.49999 eV    9.246E-03  6.400E-02       1 A         1 A         1 A
2.74999 eV    9.697E-03  6.799E-02       1 A         1 A         1 A
2.99999 eV    1.013E-02  7.182E-02       2 A         1 A         1 A
3.24999 eV    1.054E-02  7.553E-02       2 A         1 A         1 A
3.49999 eV    1.094E-02  7.912E-02       2 A         1 A         1 A

      LINES SKIPPED

   1.80 GeV   1.658E+00  6.462E-04    1.94 mm    69.09 um    12.42 um
   2.00 GeV   1.529E+00  5.872E-04    2.33 mm    88.62 um    14.79 um
   2.25 GeV   1.398E+00  5.276E-04    2.86 mm   116.43 um    17.99 um
   2.50 GeV   1.291E+00  4.794E-04    3.44 mm   142.51 um    21.45 um
   2.75 GeV   1.203E+00  4.395E-04    4.07 mm   167.81 um    25.16 um
   3.00 GeV   1.128E+00  4.060E-04    4.73 mm   192.74 um    29.11 um
   3.25 GeV   1.065E+00  3.774E-04    5.44 mm   217.50 um    33.28 um
   3.50 GeV   1.009E+00  3.527E-04    6.20 mm   242.21 um    37.66 um
   3.75 GeV   9.611E-01  3.312E-04    6.99 mm   266.93 um    42.24 um
   4.00 GeV   9.186E-01  3.122E-04    7.82 mm   291.70 um    47.03 um
   4.50 GeV   8.471E-01  2.803E-04    9.58 mm   384.34 um    57.15 um
   5.00 GeV   7.893E-01  2.546E-04   11.49 mm   469.70 um    67.95 um
   5.20 GeV   7.702E-01  2.456E-04   12.29 mm   483.24 um    72.45 um
   4.50 GeV   8.471E-01  2.803F-04    9.58 mm   384.34 um    57.15 um
   5.00 GeV   7.893F-01  2.546E-04   11.49 mm   469.70 um    67.95 um
   5.20 GeV   7.702E-01  2.456E-04   12.29 mm   483.24 um    72.45 um
-----------------------------------------------------------
 Multiply Stopping by        for Stopping Units
 -------------------        -----------------
  3.2099E+01                 eV / Angstrom
```

8. Description of the output file of the IOTA code

The following information is printed in the main output listing of the IOTA code. See Example 5.

a. The name of the exercise introduced in the first non-comment card of the INPUT file (Sect.5).

b. The characteristics of the primary ion and the target material (**`Primary particle (Z,A) Material at % Ed (eV)`**).

c. The density of the material and the number of atoms in cm^3 (**`n0`**).

d. The effective Z and A numbers for the compound (**`Z(eff)`** and **`A(eff)`**) calculated as the sum of Z and A for all components of the composite material weighted with their atomic fractions. The values Z(eff) and A(eff) are used for the crude estimation of the number of defects created in material with the help of the NRT-formula.

e. The energy of the projectile in MeV (**`Primary energy`**).

f. The number of Monte Carlo simulations for the incident ion energy (**`Number of events`**).

g. The number of primary defects calculated with the help of the NRT-formula (Eq.(8),(7)) (**`Simple NRT-formula prediction`**). For the composite materials the effective values Z(eff) and A(eff) (see Sect."d") are used instead of Z_2 and A_2 in Eq.(7).

h. The number of primary defects calculated for all successive elastic interactions of the primary ion with the lattice atoms simulated by Monte Carlo (**`Number of vacancies (Lindhard, free path)`**). For the single interaction of the ion with kinetic energy E with the lattice atom the number of defects is estimated by the following expression

$$\Delta\nu = \frac{0.8}{2E_d} \int_{E_d}^{T_{max}} \frac{d\sigma(E,T')}{dT'} \tilde{T}(T')\,dT' \Bigg/ \int_{E_d}^{T_{max}} d\sigma(E,T') \ , \qquad (11)$$

where the energy for damage production $\tilde{T}$(T) is calculated according to Eq.(7). The $\tilde{T}$(T) value is defined using the effective Z(eff) and A(eff) numbers for the compound.

The statistical error (%) (in the Example 5 it is equal to 1.02%), the ratio to the number of defects calculated by the NRT-formula (0.667) and the reference number (2) are printed on this line.

i. Number of defects calculated by the integration of Eq.(5) (**`Number of vacancies (Lindhard, integration)`**). The reference number is equal to (3).

j. Number of defects (Frenkel pairs) calculated by the direct counting of all PKA's created (**`Number of Frenkel pairs (direct counting)`**). It is a main characteristic obtained. Method of the calculation is described in Sect.2.1. The statistical error (%), the ratio to the number of defects obtained by the NRT-formula and the reference number (4) are also printed.

k. The value described above in Sect."j" multiplied on the coefficient equal to 0.8 (**`Modified counting`**).

l. The number of defects calculated with the help of the NRT formula (Eq.(8),(7)) for each component of the compound and weighted with their atomic fractions (**`NRT-formula for pure material components weighted`**).

Such estimation repeats widely used and rather incorrect counting of the number of defects for compounds. Here the total number of defects is equal to

$$\nu(E) = \sum_{i=1}^{N} w_i \, \nu_{NRT}(Z_1, A_1, Z_2^{(i)}, A_2^{(i)}) \qquad (12)$$

where the sum is for all components of the composite material, w_i is the atomic fraction, $\nu_{NRT}(Z_1, A_1, Z_2^{(i)}, A_2^{(i)})$ is calculated according to Eq.(8),(7) with the atomic and mass numbers $Z_2^{(i)}$, $A_2^{(i)}$ corresponding to the separate isolated i-th component of the compound.

m. The approximate number of defects calculated (**`Number of vacancies from Tmean calc`** and **`Number of vacancies from Tplay calc`**). The values are of secondary importance.

n. The displacement cross-section (σ_d) for the incident ion energy calculated by the SL_LNS4A code in barns (**`Displacement cross-section (Lindhard, weighted)`**). The value of σ_d is calculated as follows

$$\sigma_d(E) = \sum_{i=1}^{N} w_i \, \sigma_d^{(i)}(E) = \sum_{i=1}^{N} w_i \int \frac{d\sigma(E, T, Z_1, A_1, Z_2^{(i)}, A_2^{(i)})}{dT} \nu_{NRT}(T, Z_2^{(i)}, A_2^{(i)}, Z(eff), A(eff)) dT \, , \qquad (13)$$

where $d\sigma(E,T)$ is calculated by Eq.(4), the cascade function ν is defined by Eq.(8),(7) with Z_1 and A_1 in Eq.(7) substituted by $Z_2^{(i)}$ and $A_2^{(i)}$ and with Z_2, A_2 in Eq.(7) substituted by Z(eff) and A(eff).

The separate non-weighted $\sigma_d^{(i)}(E)$ values are printed below in barns for each component of the composite material (**`Components:`**).

o. The displacement cross-section (σ_d) obtained from the direct counting of the displaced atoms by the Monte Carlo method (Sect.2.1) in barns (**`Displacement cross-section (direct counting)`**).

The value of σ_d corresponds to the initial energy of the primary ion. It is calculated with the help of the following expression

$$\sigma_d(E) = \frac{\Delta N_{dam}}{\varphi N_0 \Delta V}$$

where ΔN_{dam} is the number of PKA's created in the volume ΔV, φ is the flux of the primary ions, N_0 is the number of atoms per cm^3.

The value of ΔV is taken small enough to minimize the energy attenuation of the primary ion.

The statistical error (%) and the reference number (7) are also printed for this σ_d value.

p. The displacement cross-section (σ_d) described in Sect."o" multiplied on the coefficient equal to 0.8 (**`modified count`**).

q. The average kinetic energy of PKA's obtained with the help of different approaches (**`Tmean =…, Tplay=…`**)

r. The range of the primary ion from the SRIM code tables (**`Path (Ziegler,evl)=`**) and the same value multiplied on the coefficient equal to $(1+0.33333A_2/A_1)$ (**`total=`**). The last value has the physical meaning only at the low primary ion energies, where the energy loss is defined by the elastic collisions.

s. The calculated range of the primary ion (**`Path (total,calc )`**). This is the approximate estimation of the range.

t. The relative number (%) of calls of the FUNCTION TPKA, when the definition of the kinetic energy T failed after MAXPLAY attempts (**`T play was cut (TPKA rout)`**). The MAXPLAY value is also printed.

u. The spatial distribution of dpa calculated basing on the approach described in Sect."h" (**`Profile of damage from Lindhard dS/dx (free path )`**). The data are normalized on the primary ion fluence equal to $1/cm^2$. The units are cm for X-axis and dpa for the Y-axis. Note, that in the current version of the code "X" refers to the distance along the ion trajectory rather than for the distance counting from the plane normal to the primary momentum of the ion (z-axis).

v. The space distribution of dpa calculated by the direct counting of PKA's created (**`Profile of damage from direct counting`**). See other notes for Sect."u".

w. The number of defects $\nu(T)$ at the different primary energies of the ion calculated below the initial ion energy E (**`Number of vacancies at different primary energies from current calculation`**).

The first column is the energy of the primary ion in MeV. The second column is the cascade function calculated by the approach described in Sect.”h”. The third column is for the number of defects obtained with the help of the integration procedure described in Sec.2.2. The last column shows the number of defects calculated from the counting of all PKA’s created. The columns have the reference numbers (2),(3),(4) corresponding to the type of the calculation described mentioned above in the IOTA listing.

The data shown in the table for each energy $E^{(i)}$ correspond to the average values for the range from $0.5(E^{(i)} + E^{(i-1)})$ to $0.5(E^{(i+1)} + E^{(i)})$. The values of the cascade function shown in the table have the maximal error at the small ion energies. To get the correct numbers of defects for such energies it is necessary to run the IOTA code for the same energies considered as the primary ones.

Example 5

Main output of the IOTA code

```
I O T A   C O D E
                 Test calculations    Si+Silicon Carbide

 Primary particle (Z,A) 14  28
 Material    at %    Ed (eV)
   14  28    50.00     40.0
    6  12    50.00     20.0
 Density =  3.2100 g/cm3    n0= 9.66531E+22 1/cm3
 Z(eff) = 10.00    A(eff)=  20.00
 Note. Nuclear nonelastic interactions are not considered here
******************************************************************************
 Primary energy   10.000     MeV
  Number of events     100
                                                        error %      /NRT
 Simple NRT-formula prediction               =  3454.1               1.000  (1)
 Number of vacancies (Lindhard, free path )  = 2303.28  (  1.02)     0.667  (2)
 Number of vacancies (Lindhard, integration) = 2407.12               0.697  (3)
----------------------------------------------------------------------------
 Number of Frenkel pairs (direct counting)   = 3041.97  (  1.96)     0.881  (4)
----------------------------------------------------------------------------
 Modified counting                           = 2433.58  (  1.96)     0.705

 NRT-formula for pure material components weighted = 3354.17                (5)
 Number of vacancies from Tmean calc               = 3159.10         0.915
 Number of vacancies from Tplay calc               = 2412.17         0.698

 Displacement cross-section (Lindhard, weighted) (b) = 1.12019E+07          (6)
 Components:  1.30498E+07 9.35397E+06
 Displacement cross-section (direct counting)    (b) = 1.43044E+07 ( 13.35) (7)
                            (modified count )    (b) = 1.14435E+07 ( 13.35)

 Tmean = 3.10542E-01 MeV     Tplay= 0.32180
 Path (Ziegler,evl)= 3.140E-04 cm,  total= 3.888E-04 cm
 Path (total,calc )                      = 3.205E-04 cm
  Histories interrupted (subr model)=           0
  T play was cut (TPKA rout) =           1.030  %    Maxplay=   125000

 Profile of damage from Lindhard dS/dx (free path )
 X=(cm)  Y=(dpa), result is normalized on primary ion fluence 1/cm**2

      X           Y           .........!.........!.........!.........!
 0.00000E+00  0.00000E+00     I                  !                   I
 1.94381E-05  1.21775E-17     *                  !                   I
 3.88762E-05  1.27821E-17     *                  !                   I
 5.83143F-05  1.38325E-17     *                  !                   I
 7.77524E-05  1.55073F-17     I*                 !                   I
 9.71905E-05  1.63149E-17     I*                 !                   I
 1.16629E-04  1.97523F-17     I*                 !                   I
 1.36067E-04  2.27565E-17     I *                !                   I
 1.55505E-04  2.57695E-17     I *                !                   I
 1.74943F-04  3.18961E-17     I--*---------------!-------------------I
 1.94381E-04  3.76942E-17     I   *              !                   I
 2.13819E-04  4.75221E-17     I    *             !                   I
 2.33257E-04  6.29700E-17     I      *           !                   I
 2.52695E-04  8.36897E-17     I         *        !                   I
 2.72133F-04  1.15629E-16     I              *   !                   I
 2.91571E-04  1.64671E-16     I                  !  *                I
 3.11010E-04  2.60224E-16     I                  !               *   I
 3.30448E-04  2.81363F-16     I                  !                   *
 3.49886E-04  1.12125E-18     I                  !                   I
 3.69324E-04  0.00000E+00     I------------------!-------------------I
 3.88762E-04  0.00000E+00     I..................!...................I
 Sum(Y)=  1.22596E-15 dpa    (unnormalized)=   2303.3


 Profile of damage from direct counting
 X=(cm)  Y=(dpa), result is normalized on primary ion fluence 1/cm**2
```

```
      X           Y           .........!.........!.........!.........!
 0.00000E+00  6.91949E-20     I                  !                   I
 1.94381E-05  2.19986E-17     I*                 !                   I
 3.88762E-05  2.01464E-17     I*                 !                   I
 5.83143F-05  2.06680E-17     I*                 !                   I
 7.77524E-05  2.14451E-17     I*                 !                   I
 9.71905E-05  2.33293F-17     I*                 !                   I
 1.16629E-04  2.92269E-17     I *                !                   I
 1.36067E-04  4.10751E-17     I  *               !                   I
 1.55505E-04  4.30126E-17     I  *               !                   I
 1.74943F-04  4.21663F-17     I--*---------------!-------------------I
 1.94381E-04  5.54996E-17     I    *             !                   I
 2.13819E-04  7.39693F-17     I      *           !                   I
 2.33257E-04  1.07625E-16     I         *        !                   I
 2.52695E-04  1.30379E-16     I            *     !                   I
 2.72133F-04  1.32343F-16     I            *     !                   I
 2.91571E-04  1.97541E-16     I                  !*                  I
 3.11010E-04  2.88415E-16     I                  !          *        I
 3.30448E-04  3.64763F-16     I                  !                  *I
 3.49886E-04  5.47172E-18     I                  !                   I
 3.69324E-04  0.00000E+00     I------------------!-------------------I
 3.88762E-04  0.00000E+00     I..................!...................I
 Sum(Y)=  1.61914E-15 dpa    (unnormalized)=   3042.0

 Number of vacancies at different primary energies from current calculation
                                      [explanation (2),(3),(4) see above]
-------------------------------------------------------------------------
    Energy     dS/dx(free path)  dS/dx(integr)     Direct counting
     (MeV)          (2)              (3)               (4)
  1.2500E-02      148.09           232.30            198.51
  5.0000E-02      409.05           522.47            502.49
  0.1000          596.70           711.68            702.07
  0.1500          748.28           861.76            895.04
  0.2000          874.16           983.53            1025.2
  0.2500          981.76           1087.1            1145.3
  0.3000          1071.0           1178.2            1247.0
  0.3500          1147.6           1254.3            1330.7
  0.4000          1214.8           1319.6            1424.8
  0.4500          1275.1           1378.8            1507.0
  0.5000          1328.7           1431.9            1591.3
```

LINES SKIPPED

```
   8.750          2271.7           2374.9            2987.7
   8.800          2273.3           2377.2            2992.5
   8.850          2274.5           2379.5            2994.2
   8.900          2275.7           2379.5            2995.3
   8.950          2277.0           2381.8            2996.8
   9.000          2278.5           2381.8            2998.6
   9.050          2279.6           2384.1            2999.9
   9.100          2280.7           2384.1            3001.8
   9.150          2282.4           2386.4            3004.3
   9.200          2282.9           2386.4            3005.0
   9.250          2284.4           2388.7            3007.2
   9.300          2286.3           2391.0            3010.2
   9.350          2288.1           2391.0            3012.8
   9.400          2289.5           2393.3            3015.0
   9.450          2290.7           2393.3            3017.0
   9.500          2291.9           2395.6            3020.4
   9.550          2293.3           2395.6            3026.8
   9.600          2294.5           2397.9            3032.7
   9.650          2295.7           2397.9            3034.1
   9.700          2297.6           2400.2            3035.9
   9.750          2298.3           2400.2            3036.9
   9.800          2299.4           2402.5            3038.0
   9.850          2300.2           2402.5            3038.6
   9.900          2301.3           2404.8            3039.8
   9.950          2302.2           2404.8            3041.4
   10.00          2303.3           2407.1            3042.0

********************************e*n*d**********************************
```

9. *Example of the calculation*

Fig.2 and Table 2 show the examples of the calculations performed with the help of the IOTA code. For the comparison the result of the calculation obtained by the MARLOWE code in Ref.[10] and by the TRIM-2003 code are also shown.

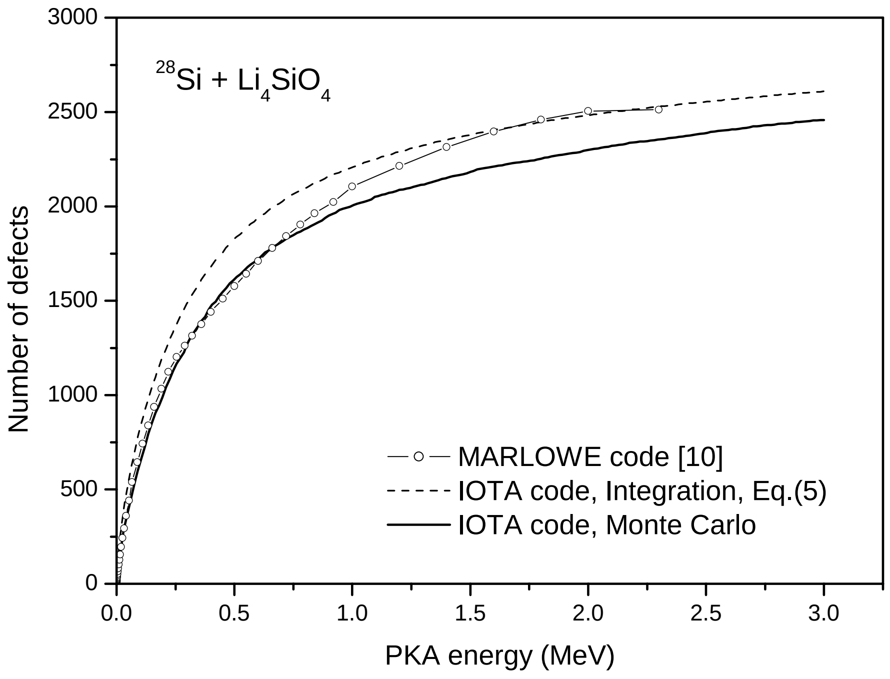

Fig.2 Number of defects calculated with the help of the IOTA code and the MARLOWE code [10] for Li_4SiO_4 irradiated with the ^{28}Si. Effective threshold energy is the same as for Table 2 data.

Table 2

Number of defects calculated with the help of the different codes and by the NRT formula. Effective threshold energy is equal to 17 eV for Be, 33 eV for Li, 19 eV for O and 112 eV for Si (taken for the purpose of comparison only).

Code	^{9}Be in ^{9}Be E_{Be}=6 MeV	^{4}He in ^{9}Be E_{He}=13 MeV	^{28}Si in Li_4SiO_4 E_{Si}=2.3 MeV
IOTA (Integration,Eq.(5))	481	165	2530
IOTA (Monte Carlo)	507	180	2360
MARLOWE [10]	580	140	2500
TRIM -2003	577	217	2200
NRT formula	436	96	2780

10. References

1. J.Lindhard, M.Scharff, H.E.Schiott, Range Concepts and Heavy Ion Ranges, *K. Dan. Vidensk. Selsk. Mat. Fys. Medd.*, **33,** N14 (1963)

2. K.B.Winterbon, P.Sigmund, J.B.K.Sanders, Spatial Distribution of Energy De posited by Atomic Particles in Elastic Collisions, *K. Dan. Vidensk. Selsk. Mat. Fys.Medd.*, **37**, N14 (1970)

3. A.F.Burenkov, F.F.Komarov, M.A.Kumahov, M.M.Temkin, Special Distribution of Energy Deposited in Atomic Collision Cascades in Matter, Energoatomizdat, Moscow, 1985

4. M.T.Robinson, Basic Physics of Radiation Damage Production, *J. Nucl. Mat.* 216 (1994) 1-28

5. L.R.Greenwood, Neutron Interactions and Atomic Recoil Spectra, *J. Nucl. Mat.* 216 (1994) 29-44

6. J.P.Biersack, L.G.Haggmark, A Monte Carlo Computer Program for the Transport of Energetic Ions in Amorphous Targets, *Nucl. Instr. Meth.* 174 (1980) 257-269

7. J.F.Ziegler, Particle Interactions with Matter, http://www.srim.org

8. M.J.Norgett, M.T.Robinson, I.M.Torrens, A Proposed Method of Calculating Displacement Dose Rates, *Nuclear Engineering and Design*, **33** (1975) 50-54.

9. GNU Fortran G77 for Windows 95/98/Me/NT4/2000/XP, http://kkourakis.tripod.com

10. D.Leichte, Displacement Damage Parameters for Fusion Breeder Blanket Materials Based on BCA Computer Simulations J. Nucl. Mat. 307-311 (2002) 793-797

11. A.Yu.Konobeyev, Yu.A.Korovin, V.N.Sosnin, Neutron Displacement Cross-Sections for Structural Materials below 800 MeV, J. Nucl. Mat. 186 (1992) 117

12. Yu.A.Korovin, A.Yu.Stankovsky, A.Yu.Konobeyev, P.E.Pereslavtsev BISERM-2. Nuclear Data Library for Evaluation of Radiation Effects in Materials Induced by Neutrons of Intermediate Energies, IAEA-NDS-203 (March 1997).

13. W.Eckstein, Computer Simulation of Ion-Solid Interactions, Springer-Verlag, 1991, p.59.

14. U.Littmark, J.F.Ziegler, Handbook of Range Distributions, Pergamon Press, 1985.

15. T.W.Armstrong, K.C.Chandler, Nucl. Instr. Meth. 113 (1973) 313

16. H.G.Hughes, H.W.Egdorf, F.C.Gallmeier, J.S.Hendricks et al: MCNPX^{TM} User's manual, Version 2.4.0, September 2002, LA-CP-02-408

11. *Description of the diskette with the IOTA code distributed with this report. Windows version using GNU FORTRAN Compiler*

The diskette contains the following information

Directory	Content
DESCRIPTION	this report
FORTRAN	the source of the IOTA and the SL_LNS4A codes
EXE	executable modules of the IOTA code and the SL_LNS4A code (Windows)
EXAMPLE	example of the input data
GNU_FORTRAN_G77	reference with instruction about GNU FORTRAN installation

To perform the test calculation

1) create the working directory on PC hard disk (e.g. C:\). Call it, e.g. IOTACODE
2) copy all executable files from the diskette\EXE directory to the C:\IOTACODE
3) copy the file INPUT and all files with *.dat and *.zie extensions from diskette\EXAMPLE to C:\IOTACODE
4) run iota.exe and compare the output files with the files located in diskette\EXAMPLE directory

To check SL_LNS4A code execution run SL_LNS4A.exe and compare the files created with the corresponding files from diskette\EXAMPLE

To check data for the stopping power install the SRIM/TRIM code from www.srim.org, prepare the necessary data and compare them with the data with *.zie extension from the diskette\EXAMPLE directory.

To install GNU FORTRAN Compiler go in directory A:\GNU_FORTRAN_G77 and open file gnu.html by the Internet browser. Follow the instruction and install “A minimal set of G77”.

12. The text of the IOTA code

```
***************************************************************************
*
* IOTA code
* ---------
*
*
* Maximal number of the material component is 21. To increase this
* value change parameter(maxc= to N desirable value in all routines
* and maxc1= to N+1
*
* Maximal energy = 5 GeV
*
*
c
c GNU random number generator is used now. To use other generator
c change function RANDOM
c
*
* Last changes 09/01/2004
*
c
       PROGRAM IOTA
       parameter (maxe=50001)
       Common/EIN/EIN(MAXE),NH(MAXE),KEIN
       COMMON/RECSTORE/TKIN(50003),ZZ(50003),NKO(50003),INDIC(50003),
     #                NREC
       Common/MERCURY/CMD(5),Ecrit,MD
c
c=======================================================================
c
c
*
* read input data file
         CALL READ_INPUT
*
* prepare basic data for calculations
         CALL PRECALCULATION
*
* read cross-sections and dE/dx
         CALL READ_DATA
*
* make interpolation between data points
         CALL INCREASE_DATA_ARRAY
*
* calculate dE/dx by SPAR routine
         CALL DEDX_SPAR
*
*
c
c=========================================================================
c
* Cycle for energies
* -----------------
        Do IE=1,KEIN
        T0=EIN(IE)
        NHIST=NH(IE)
c
            If(T0.gt.5000.) then
            Print *,' T0 exceeds the allowable maximum. Change the code'
            Call Error('    ',5000)
            Endif
```

```
*
        Call INITIALIZE_AND_ZEROIZE(T0)
*
* Cycle for events
* ----------------
        Do 7 IH=1,Nhist
        Print 1000,IH,T0
 1000   Format('*************  history =',I10,'     E=',g12.5)
*
        Call DISPERS(1)
*
* Primary ion movement
*                    IDE corresponds to the energy of the primary
*                        particle which attenuates in material
        Z=0.0
        IK=1
        IDE=-1
        Call MODEL(IK,T0,Z,IDE)
        if(NREC.le.0) goto 6
*
* Secondary recoils movement
   2    Call STORE_OUT(MK,T1,Z,IDE)
*
                                    if(nrec.lt.0.and.md.eq.0) then
                                    print *,'strange error'
                                    stop
                                    endif

        if(NREC.le.0) goto 6
        IK=MK+1
        Call MODEL(IK,T1,Z,IDE)
        goto 2
*
*
*
   6    Call DISPERS(2)
   7    Continue
c
*
* Integration of nu(T)dS/dT along the trajectory
        Call  LindInt(T0)
        Print *,'*****  Finished !'
*
        Call Print(T0,Nhist)
c
        Enddo !   Do IE=1,KEIN
*
        Call end_of_calculation
        Stop
        End
*
*********************************************************************
*
        Subroutine ARRAYINT(KEY, E,CS,N,T,INTS0,RES)
*       -----------------------------------------
* REAL*4
*
*  To get RES value for arrays E(N),CS(N) at point T depending
*  -----------------------------------------------------
*  from INTS int scheme
*  ------------------
*
* KEY defines how to treat situation when T is not covered by E array
* ---
* KEY = 0   :  stop when T < E(1)  or T > E(N)
*     = 1   :  assign RES=0.0 if T < E(1) and RES=CS(N) if T > E(N)
*
* INT has the usual meaning as in ENDF/B format
*
```

```
* Attention (!)
* ---------
* Input arrays E(N) and CS(N) should be in increased order
*
*
       Dimension E(N), CS(N)
       if(KEY.lt.0.or.KEY.gt.1) then
       Print *,'                         E R R O R         '
       Print *,'Subroutine  ARRAYINT:  KEY=',KEY,' is not acceptable !'
       print *,'                            press any key...'
       pause
       stop
       endif
       INTS=INTS0
*
       If(T.lt.E(1)) goto 8000
       If(T.gt.E(N)) goto 9000
*
       Do i=1,N
       i1=i
       If( T.le.E(i) ) goto 1000
       Enddo
            goto 9500
1000   if(i1.eq.1) then  ! means that T=E(1)
       RES=CS(1)
       Return
       Endif
          X1=E(i1-1)
          X2=E(i1)
          Y1=CS(i1-1)
          Y2=CS(i1)
          WL=T
       CALL INTERMEDIATE(X1,X2,Y1,Y2,WL,INTS,RESINT)
       RES=RESINT
       Return
*
*
* T < E(1)
* --------
8000   if(key.eq.0) then
       Write(6,*)'                     E R R O R        '
       Write(6,8001) T,E(1),key
8001   Format(1x,'Subroutine  ARRAYINT:'/1x,
     # ' Requested energy T=',e12.5,' is less than minimal array '
     # ,'E(1) value=',e12.5
     #  //1x,'             CHANGE THE DATA or KEY number !!'/1x,
     # ' now  KEY is equal to', i5)
       print *,'                            press any key...'
       pause
       stop
       endif
c
       RES=0.0
       Print 8002,T,RES
8002   Format(1x,'ARRAYINT(N): for T =',e12.5,' result ',
     # 'is set to',e12.5,' (min)')
       Return
*
* T > E(N)
* --------
9000   if(key.eq.0) then
       Write(6,*)'                     E R R O R        '
       Write(6,9001) T,E(N),key
9001   Format(1x,'Subroutine  ARRAYINT:'/1x,
     # ' Requested energy T=',e12.5,' is more than MAXIMAL array '
     # ,'E(N) value=',e12.5
     #  //1x,'             CHANGE THE DATA or KEY number !!'/1x,
     # ' now  KEY is equal to', i5)
       print *,'                            press any key...'
       pause
       stop
       endif
c
       RES=CS(N)
       Print 9002,T,RES
9002   Format(1x,'ARRAYINT(X): for T =',e12.5,' result ',
     # 'is set to',e12.5,' (MAX)')
       Return
*
9500   Write(6,9501)
9501   Format(1x,'Subroutine  ARRAYINT:  Strange E R R O R')
       print *,'                           press any key...'
       pause
       stop
       End
*
*************************************************************************
*
       Function CF(IK,MK,TREC)
*      ----------------------
       parameter (maxc=21,maxc1=22)
       COMMON/NRT1/ANRT(MAXC1),BNRT(MAXC1),GNRT(MAXC1)
       Common/Z2A2/ZT(MAXC),AT(MAXC),FR(MAXC),ED(MAXC),NK
       EDkeV=ED(MK)*1000.
* NRT
       TT=TREC*1000.
*
       If(TT.lt.EDkeV) then
                       CF=0.0
                       Return
                       Endif
       If(TT.lt.2.*EDkeV) TT=2.*EDkeV
*
       CF=(0.8/(2.*EDkeV))*TT/
     # (1.+ANRT(IK)*TT+BNRT(IK)*(TT**0.75)+GNRT(IK)*(TT**0.16666666666))
        RETURN
        END
*
*************************************************************************
*
       FUNCTION CF1(ED,ATRN,BTRN,GTRN,TREC)
*      ------------------------------------
* As CF, but for special purpose
       TT=TREC*1000.
       CF1=(0.8/(2.*ED*1000.))*TT/
     # (1.+ATRN*TT+BTRN*(TT**0.75)+GTRN*(TT**0.16666666666))
        RETURN
        END
*
*************************************************************************
*
* Special function to incorporate MD result
*
       Function CF2(IK,MK,TREC)
*      -----------------------
       parameter (maxc=21,maxc1=22)
       COMMON/NRT1/ANRT(MAXC1),BNRT(MAXC1),GNRT(MAXC1)
       Common/Z2A2/ZT(MAXC),AT(MAXC),FR(MAXC),ED(MAXC),NK
       Common/MERCURY/CMD(5),Ecrit,MD
       EDkeV=ED(MK)*1000.
* mod NRT
       TT=TREC*1000.   ! MeV --> keV
*
       If(TT.lt.EDkeV) then
                       CF2=0.0
                       Return
                       Endif
*
```

```
       If(TT.lt.2.*EDkeV) TT=2.*EDkeV
*
*
       eMD= TT/
     # (1.+ANRT(IK)*TT+BNRT(IK)*(TT**0.75)+GNRT(IK)*(TT**0.16666666666))
*
* usual formula
       CF2=(0.8/(2.*EDkeV))*eMD
*
       Elow =CMD(4)
       Ehigh=CMD(5)
*
*
      if(eMD.lt.Elow)  eMD=Elow
      if(eMD.gt.Ehigh) eMD=Ehigh
      efficiency=CMD(1)*(  eMD**CMD(2)  ) + CMD(3)*eMD
*
*
* modified NRT
        CF2=efficiency*CF2
*
        RETURN
        END
*
************************************************************************
*
        Subroutine COS_DEFINE(IK,MK,T0,TPVA,COST)
*       ----------------------------------------
        parameter (maxc=21,maxc1=22)
*
        Common/Basic/Z1,A1,Z2,A2,AW,RO,RN0
        Common/Z2A2/ZT(MAXC),AT(MAXC),FR(MAXC),ED(MAXC),NK
        Common/PKA/ALPHA2(MAXC1,MAXC),E0(MAXC1,MAXC)
*
* T(M2)=[ 2M1*M2/(M1+M2)**2 ]* (  1-COS( TETA(CM) )) * T
*
* TG(TETA(LC))=SIN(TETA(CM))/( M1/M2 + COS(TETA(CM)) )
*
        T=T0
        TM2=TPVA
* cos(teta) in CM
        COSTCM=1.-TM2/(T*0.5*ALPHA2(IK,MK))
        if(COSTCM.gt.1.) COSTCM =0.99999999
        if(COSTCM.lt.-1.) COSTCM=-0.9999999
        SINTCM=Sqrt(1.-COSTCM**2)
                    A11=A1
        If(IK.ne.1) A11=AT(IK-1)
*
        TETA=ATAN(   SINTCM/( (A11/AT(MK))+COSTCM )   )
        COST=COS(TETA)
c       write(445,*)'COS(TCM),COST=',COSTCM,COST
*
        RETURN
        END
*
************************************************************************
*
        Subroutine CRITICAL_ENERGY
*       --------------------------
        parameter (maxc=21,maxc1=22)
*
* search root Tdam(E)-Ecrit=0, it is supposed, that d^2Tdam/dE^2 < 0
*
        Common/Z2A2/ZT(MAXC),AT(MAXC),FR(MAXC),ED(MAXC),NK
        COMMON/NRT1/ANRT(MAXC1),BNRT(MAXC1),GNRT(MAXC1)
        Common/MERCURY/CMD(5),Ecrit,MD
        Common/SATURN/TcritLAB(MAXC1)
*
* Tcrit correspods to  Abs( Tdam(E)-Ecrit ) < Err1
```

```
        Err1=1.E-4
*
        NK1=NK+1
        If(Ecrit.le.0.0) Call Error('Ecri',0)
*
        Do 70000 IK=1,NK1
*
         X0=Ecrit
         If(Tdam(IK,X0)-Ecrit)1,8000,90000
*
    1    h=0.1
    2    XU=X0*(1.+h)
             RK=(Tdam(IK,XU)-Tdam(IK,X0))/(XU-X0)
             X1=X0-(Tdam(IK,X0)-Ecrit)/RK
* Tcrit > 0.1 GeV ?
         If(X1.ge.1.e+5) then
                         Tcrit=X1
                         Print 91000,Tcrit
                         goto 9000
                         endif
         If(abs(Tdam(IK,X1)-Ecrit).lt.err1) then
         Tcrit=X1
         goto 9000
                                        endif
         If(Tdam(IK,X1)-Ecrit)1000, 8500, 2000
 1000    h=0.1
         X0=X1
         goto 2
 2000    h=h/2.
         goto 2
c
 8000    Tcrit=X0
         goto 9000
 8500    Tcrit=X1
         goto 9000
c
 9000    continue
*
* keV --> MeV
        TcritLAB(ik)=Tcrit*0.001
70000   continue
*
*
        Return
*
*
90000   Print *,'Tcrit search failed !  Tdam(Xinitial) must be < Ecrit'
        Print *,'                                 press any key...'
        pause
        stop
91000   Format(70('*')/'*         Energy Tcrit =',g12.5,' keV',
     #  ' is not realistic',T69,'*'/70('*'))
        End
*
************************************************************************
*
        Subroutine DATATAB
*       -----------------
c A, W (amu), Ro (g/cm3) , <I> (eV) for elements
c
c Data from SRIM code
c  <I> is mean ionization potential (eV) checked by data in figures
c  presented in http://www.srim.org/SRIM/SRIMPICS/IONIZ.htm,
c  except data noted below
c
c subroutine called from PRECALCULATION
c
      Common/DATATAB1/WTAB1(101),ROTAB1(101),RITAB1(101),IATAB1(101)
      Dimension IATAB(92), WTAB(92), ROTAB(92), RITAB(92)
```

```
*
*  H
      Data IATAB( 1),  WTAB( 1),  ROTAB( 1),  RITAB( 1)
     & /       1   ,    1.0079,   0.071486,    19.60/
* He
      Data IATAB( 2),  WTAB( 2),  ROTAB( 2),  RITAB( 2)
     & /       4   ,    4.0026,   0.125880,    39.00/
* Li
      Data IATAB( 3),  WTAB( 3),  ROTAB( 3),  RITAB( 3)
     & /       7   ,    6.9410,   0.534000,    42.20/
* Be
      Data IATAB( 4),  WTAB( 4),  ROTAB( 4),  RITAB( 4)
     & /       9   ,    9.0122,   1.848000,    63.70/
*  B
      Data IATAB( 5),  WTAB( 5),  ROTAB( 5),  RITAB( 5)
     & /      11   ,   10.8110,   2.350200,    76.00/
*  C
      Data IATAB( 6),  WTAB( 6),  ROTAB( 6),  RITAB( 6)
     & /      12   ,   12.0110,   2.253000,    79.10/
*  N
      Data IATAB( 7),  WTAB( 7),  ROTAB( 7),  RITAB( 7)
     & /      14   ,   14.0070,   1.026000,    84.20/
*  O
      Data IATAB( 8),  WTAB( 8),  ROTAB( 8),  RITAB( 8)
     & /      16   ,   15.9990,   1.426000,    96.00/
*  F
      Data IATAB( 9),  WTAB( 9),  ROTAB( 9),  RITAB( 9)
     & /      19   ,   18.9980,   1.111100,   107.00/
* Ne
      Data IATAB(10),  WTAB(10),  ROTAB(10),  RITAB(10)
     & /      20   ,   20.1797,   1.204000,   133.00/
* Na
      Data IATAB(11),  WTAB(11),  ROTAB(11),  RITAB(11)
     & /      23   ,   22.9898,   0.970000,   149.00/
* Mg
      Data IATAB(12),  WTAB(12),  ROTAB(12),  RITAB(12)
     & /      24   ,   24.3050,   1.736600,   156.00/
* Al
      Data IATAB(13),  WTAB(13),  ROTAB(13),  RITAB(13)
     & /      27   ,   26.9815,   2.702000,   170.00/
* Si
      Data IATAB(14),  WTAB(14),  ROTAB(14),  RITAB(14)
     & /      28   ,   28.0860,   2.321200,   171.00/
*  P
      Data IATAB(15),  WTAB(15),  ROTAB(15),  RITAB(15)
     & /      31   ,   30.9737,   1.821900,   173.00/
*  S
      Data IATAB(16),  WTAB(16),  ROTAB(16),  RITAB(16)
     & /      32   ,   32.0660,   2.068600,   184.00/
* Cl
      Data IATAB(17),  WTAB(17),  ROTAB(17),  RITAB(17)
     & /      35   ,   35.4530,   1.895600,   178.00/
* Ar
      Data IATAB(18),  WTAB(18),  ROTAB(18),  RITAB(18)
     & /      40   ,   39.9480,   1.650400,   190.00/
*  K
      Data IATAB(19),  WTAB(19),  ROTAB(19),  RITAB(19)
     & /      39   ,   39.0983,   0.863180,   190.00/
* Ca
      Data IATAB(20),  WTAB(20),  ROTAB(20),  RITAB(20)
     & /      40   ,   40.0800,   1.540000,   198.00/
* Sc
      Data IATAB(21),  WTAB(21),  ROTAB(21),  RITAB(21)
     & /      45   ,   44.9560,   2.989000,   225.00/
* Ti
      Data IATAB(22),  WTAB(22),  ROTAB(22),  RITAB(22)
     & /      48   ,   47.9000,   4.518900,   231.00/
*  V
      Data IATAB(23),  WTAB(23),  ROTAB(23),  RITAB(23)
     & /      51   ,   50.9420,   5.960000,   249.00/
* Cr
      Data IATAB(24),  WTAB(24),  ROTAB(24),  RITAB(24)
     & /      52   ,   51.9960,   7.200000,   265.00/
* Mn
      Data IATAB(25),  WTAB(25),  ROTAB(25),  RITAB(25)
     & /      55   ,   54.9380,   7.434100,   275.00/
* Fe
      Data IATAB(26),  WTAB(26),  ROTAB(26),  RITAB(26)
     & /      56   ,   55.8470,   7.865800,   286.00/
* Co
      Data IATAB(27),  WTAB(27),  ROTAB(27),  RITAB(27)
     & /      59   ,   58.9330,   8.900000,   306.00/
* Ni
      Data IATAB(28),  WTAB(28),  ROTAB(28),  RITAB(28)
     & /      58   ,   58.6900,   8.895500,   321.00/
* Cu
      Data IATAB(29),  WTAB(29),  ROTAB(29),  RITAB(29)
     & /      63   ,   63.5460,   8.920000,   332.00/
* Zn
      Data IATAB(30),  WTAB(30),  ROTAB(30),  RITAB(30)
     & /      64   ,   65.3900,   7.140000,   330.00/
* Ga
      Data IATAB(31),  WTAB(31),  ROTAB(31),  RITAB(31)
     & /      69   ,   69.7200,   5.904000,   334.00/
* Ge
      Data IATAB(32),  WTAB(32),  ROTAB(32),  RITAB(32)
     & /      74   ,   72.6100,   5.350000,   335.00/
* As
      Data IATAB(33),  WTAB(33),  ROTAB(33),  RITAB(33)
     & /      75   ,   74.9220,   5.727000,   353.00/
* Se
      Data IATAB(34),  WTAB(34),  ROTAB(34),  RITAB(34)
     & /      80   ,   78.9600,   4.810000,   351.00/
* Br
      Data IATAB(35),  WTAB(35),  ROTAB(35),  RITAB(35)
     & /      79   ,   79.9040,   3.199000,   317.00/
* Kr
      Data IATAB(36),  WTAB(36),  ROTAB(36),  RITAB(36)
     & /      84   ,   83.8000,   2.602100,   348.00/
* Rb
      Data IATAB(37),  WTAB(37),  ROTAB(37),  RITAB(37)
     & /      85   ,   85.4700,   1.532000,   364.00/
* Sr
      Data IATAB(38),  WTAB(38),  ROTAB(38),  RITAB(38)
     & /      88   ,   87.6200,   2.600000,   366.00/
*  Y
      Data IATAB(39),  WTAB(39),  ROTAB(39),  RITAB(39)
     & /      89   ,   88.9050,   4.469000,   375.00/
* Zr
      Data IATAB(40),  WTAB(40),  ROTAB(40),  RITAB(40)
     & /      90   ,   91.2200,   6.490000,   389.00/
* Nb
      Data IATAB(41),  WTAB(41),  ROTAB(41),  RITAB(41)
     & /      93   ,   92.9060,   8.570000,   417.00/
* Mo
      Data IATAB(42),  WTAB(42),  ROTAB(42),  RITAB(42)
     & /      98   ,   95.9400,  10.206000,   423.00/
* Tc
      Data IATAB(43),  WTAB(43),  ROTAB(43),  RITAB(43)
     & /      97   ,   97.0000,  11.500000,   428.00/
* Ru
      Data IATAB(44),  WTAB(44),  ROTAB(44),  RITAB(44)
     & /     102   ,  101.0700,  12.300000,   441.00/
* Rh
      Data IATAB(45),  WTAB(45),  ROTAB(45),  RITAB(45)
     & /     103   ,  102.9100,  12.399000,   468.00/
* Pd
      Data IATAB(46),  WTAB(46),  ROTAB(46),  RITAB(46)
     & /      106   ,  106.4000,  12.020000,   476.00/
* Ag
      Data IATAB(47),  WTAB(47),  ROTAB(47),  RITAB(47)
     & /      107   ,  107.8700,  10.473000,   488.00/
* Cd
      Data IATAB(48),  WTAB(48),  ROTAB(48),  RITAB(48)
     & /      114   ,  112.4000,   8.642000,   441.00/
* In
      Data IATAB(49),  WTAB(49),  ROTAB(49),  RITAB(49)
     & /      115   ,  114.8200,   7.300000,   481.00/
* Sn
      Data IATAB(50),  WTAB(50),  ROTAB(50),  RITAB(50)
     & /      120   ,  118.7100,   7.281600,   478.00/
* Sb
      Data IATAB(51),  WTAB(51),  ROTAB(51),  RITAB(51)
     & /      121   ,  121.7500,   6.684000,   472.00/
* Te
      Data IATAB(52),  WTAB(52),  ROTAB(52),  RITAB(52)
     & /      130   ,  127.6000,   6.250000,   485.00/
*  I
      Data IATAB(53),  WTAB(53),  ROTAB(53),  RITAB(53)
     & /      127   ,  126.9000,   4.937300,   455.00/
* Xe
      Data IATAB(54),  WTAB(54),  ROTAB(54),  RITAB(54)
     & /      132   ,  131.3000,   3.058900,   440.00/
* Cs
      Data IATAB(55),  WTAB(55),  ROTAB(55),  RITAB(55)
     & /      133   ,  132.9100,   1.878500,   488.00/
* Ba
      Data IATAB(56),  WTAB(56),  ROTAB(56),  RITAB(56)
     & /      138   ,  137.3270,   3.510000,   491.00/
* La
      Data IATAB(57),  WTAB(57),  ROTAB(57),  RITAB(57)
     & /      139   ,  138.9100,   6.173800,   472.00/
* Ce
      Data IATAB(58),  WTAB(58),  ROTAB(58),  RITAB(58)
     & /      140   ,  140.1200,   6.672400,   501.00/
* Pr
      Data IATAB(59),  WTAB(59),  ROTAB(59),  RITAB(59)
     & /      141   ,  140.9100,   6.773000,   513.00/
* Nd
      Data IATAB(60),  WTAB(60),  ROTAB(60),  RITAB(60)
     & /      142   ,  144.2400,   7.008000,   546.00/
* Pm
      Data IATAB(61),  WTAB(61),  ROTAB(61),  RITAB(61)
     & /      148   ,  148.0000,   6.475000,   560.00/
* Sm
      Data IATAB(62),  WTAB(62),  ROTAB(62),  RITAB(62)
     & /      152   ,  150.3600,   7.520000,   566.00/
* Eu
      Data IATAB(63),  WTAB(63),  ROTAB(63),  RITAB(63)
     & /      153   ,  151.9700,   5.244000,   580.00/
* Gd
      Data IATAB(64),  WTAB(64),  ROTAB(64),  RITAB(64)
     & /      158   ,  157.2500,   7.901000,   586.00/
* Tb
      Data IATAB(65),  WTAB(65),  ROTAB(65),  RITAB(65)
     & /      159   ,  158.9300,   8.230000,   614.00/
* Dy
      Data IATAB(66),  WTAB(66),  ROTAB(66),  RITAB(66)
     & /      164   ,  162.5000,   8.551000,   603.00/
* Ho
      Data IATAB(67),  WTAB(67),  ROTAB(67),  RITAB(67)
     & /      165   ,  164.9300,   8.795000,   641.00/
* Er
      Data IATAB(68),  WTAB(68),  ROTAB(68),  RITAB(68)
     & /      166   ,  167.2600,   9.066000,   661.00/
* Tm
      Data IATAB(69),  WTAB(69),  ROTAB(69),  RITAB(69)
     & /      169   ,  168.9300,   9.321000,   683.00/
* Yb
      Data IATAB(70),  WTAB(70),  ROTAB(70),  RITAB(70)
     & /      174   ,  173.0400,   6.960000,   684.00/
* Lu
      Data IATAB(71),  WTAB(71),  ROTAB(71),  RITAB(71)
     & /      175   ,  174.9700,   9.841000,   694.00/
* Hf
      Data IATAB(72),  WTAB(72),  ROTAB(72),  RITAB(72)
     & /      180   ,  178.4900,  13.310000,   675.00/
* Ta
      Data IATAB(73),  WTAB(73),  ROTAB(73),  RITAB(73)
     & /      181   ,  180.9500,  16.601000,   729.00/
*  W
      Data IATAB(74),  WTAB(74),  ROTAB(74),  RITAB(74)
     & /      184   ,  183.8500,  19.350000,   735.00/
* Re
      Data IATAB(75),  WTAB(75),  ROTAB(75),  RITAB(75)
     & /      187   ,  186.2000,  20.530001,   749.00/
* Os
      Data IATAB(76),  WTAB(76),  ROTAB(76),  RITAB(76)
     & /      192   ,  190.2000,  22.480000,   746.00/
* Ir
      Data IATAB(77),  WTAB(77),  ROTAB(77),  RITAB(77)
     & /      193   ,  192.2000,  22.421000,   757.00/
* Pt
      Data IATAB(78),  WTAB(78),  ROTAB(78),  RITAB(78)
     & /      195   ,  195.0800,  21.450001,   792.00/
* Au
      Data IATAB(79),  WTAB(79),  ROTAB(79),  RITAB(79)
     & /      197   ,  196.9700,  19.311001,   789.00/
* Hg
      Data IATAB(80),  WTAB(80),  ROTAB(80),  RITAB(80)
     & /      202   ,  200.5900,  13.546200,   800.00/
* Tl
      Data IATAB(81),  WTAB(81),  ROTAB(81),  RITAB(81)
     & /      205   ,  204.3800,  11.850000,   810.00/
* Pb
      Data IATAB(82),  WTAB(82),  ROTAB(82),  RITAB(82)
     & /      208   ,  207.1900,  11.343700,   787.00/
* Bi
      Data IATAB(83),  WTAB(83),  ROTAB(83),  RITAB(83)
     & /      209   ,  208.9800,   9.800000,   819.00/

* Po (no fig for <I>)
      Data IATAB(84),  WTAB(84),  ROTAB(84),  RITAB(84)
     & /      209   ,  210.0000,   9.251100,   830.00/
* At <I> is not well defined from SRIM data (see original files)
      Data IATAB(85),  WTAB(85),  ROTAB(85),  RITAB(85)
     & /      210   ,  210.0000,  10.000000,   756.95/
* Rn <I> is not well defined from SRIM data (see original files)
      Data IATAB(86),  WTAB(86),  ROTAB(86),  RITAB(86)
     & /      222   ,  222.0000,   9.910000,   736.54/
* Fr (no fig for <I>)
      Data IATAB(87),  WTAB(87),  ROTAB(87),  RITAB(87)
     & /      223   ,  223.0000,  10.000000,   827.00/
* Ra (no fig for <I>)
      Data IATAB(88),  WTAB(88),  ROTAB(88),  RITAB(88)
     & /      226   ,  226.0000,   5.022200,   826.00/
* Ac (no fig for <I>)
      Data IATAB(89),  WTAB(89),  ROTAB(89),  RITAB(89)
     & /      227   ,  227.0000,  10.000000,   841.00/
* Th (no fig for <I>)
      Data IATAB(90),  WTAB(90),  ROTAB(90),  RITAB(90)
     & /      232   ,  232.0000,  11.658000,   847.00/
* Pa (no fig for <I>)
      Data IATAB(91),  WTAB(91),  ROTAB(91),  RITAB(91)
     & /      231   ,  231.0000,  15.370000,   878.00/
*  U
```

```
      Data IATAB(92),  WTAB(92),  ROTAB(92),  RITAB(92)
     & /      238   ,  238.0300,  19.042999,   842.00/
c
c
      Do i=1,92
      IATAB1(i)=IATAB(i)
      WTAB1(i) =WTAB(i)
      ROTAB1(i)=ROTAB(i)
      RITAB1(i)=RITAB(i)
      enddo
          Do i=93,101
          IATAB1(i)=0
          WTAB1(i) =0.0
          ROTAB1(i)=0.0
          RITAB1(i)=0.0
          enddo
      Return
      End
*
************************************************************************
*
       Subroutine dEdXNUC_RECALC(Z1,Z2,A1,A2,E,sN)
*      ----------------------------------------
c Z1,A1: projectile, Z2,A2: target
c
c
       Real*8 Z,EL,RL,rksi,eps,t,sNd
c
c E (MeV)
c
       Z=(Z1**0.666666d0+Z2**0.6666666d0)**1.5d0
c
       EL=3.07358d-05*Z1*Z2*(Z**0.33333333d0)*( (A1/A2)+1.d0 )
c (no coeff, n0)
       RL=(Z**0.6666666d0) *(  (A1+A2)**2  )/(A1*A2)
c
       rksi=1.309d0
c
       eps=E/EL
       t=((2.d0*rksi)**0.6666666d0)*(eps**0.8888888d0)
c
       sNd=(EL/RL)*(1.d0/eps)*
     # (  -dsqrt(t)/dsqrt(1.+t)   +dlog(dsqrt(t)+dsqrt(1.d0+t))  )
c
       sN=real(sNd)
       Return
       End
*
************************************************************************
*
       Subroutine DEDX_SPAR
*      --------------------
* Calculation of dE/dx by SPAR (Armstrong, Chandler) routine
*
* This subroutine should be executed after
*     subr INCREASE_DATA_ARRAY, where NMAX is calculated
*
* units for stopping power [MeV/cm], range [cm]
*
       parameter (maxc=21,maxc1=22)
       Character OUTSPAR*12, C5*5
       Common/Basic/Z1,A1,Z2,A2,AW,RO,RN0
       Common/Z2A2/ZT(MAXC),AT(MAXC),FR(MAXC),ED(MAXC),NK
*
       Common/SKY1/dEdx(MAXC1,10002),dEdxN1(MAXC1,10002),
     # Rp1(MAXC1,10002)
*
       Common/STARS1/ dSdX1(MAXC1,MAXC,10002),
     # dEdXNuc1(MAXC1,MAXC,10002),
     # dEdXNuc0Ed1(MAXC1,MAXC,10002), dEdXNucEdTm1(MAXC1,MAXC,10002),
     # CST1(MAXC1,MAXC,10002), NMAX
       Common/ISTOP/ISTOP
       Common/VENUS/RI(MAXC)
       Dimension TmpE(10002),dxdE(10002)
*
       if(istop.ne.1) return
*
*
* Condensed matter supposed
      If(RO.le.0.01)  goto 91000
*
* Mean <I> for compound (only for printing)
           RIL=0.d0
           Do i=1,NK
           RIL= RIL+FR(i)*ZT(i)*alog( RI(i) )
           Enddo
           RIL=2.718281828**(RIL/Z2)
*
      jj=2 ! condensed element
          if(NK.ne.1) then
* check if all components are isotopes of the same element
          IZT0=ifix( ZT(1)+0.001)
          do i=1,NK
          IZT=ifix( ZT(i)+0.001)
          if(IZT.ne.IZT0) then
                          jj=4   ! condensed compound
                          goto 1
                          endif
          IZT0=IZT
          enddo
                      endif
    1     continue
*
*
      NK1=NK+1
*
*
*
      Do  IK=1,NK1
*     ------------
* IK - id of the moving ion,   IK=1: primary particle
*
*
* Projectile
      if(IK.eq.1) then
                     ZPR=Z1
                     APR=A1
                  else
                     ZPR=ZT(IK-1)
                     APR=AT(IK-1)
                  endif
*
        IZA=1000*ifix(ZPR+0.001)+ifix(APR+0.001)
        Write(C5,'(i5)')IZA
        outspar=C5//'.sto'
        open(21,file=outspar)

        Write(21,90001)ZPR,APR,IK,Z2,A2,JJ,RO,RN0,RIL*1.e+6
        Write(21,90002)
        Write(21,90003)
*
*
        Do I=1,999999999
*
* J=(lg(E)+6)*1000+1  means that energy can changes from 10**-6 MeV
* If 10002 array dimension will be increased change also 1000 below
* and in other subroutines (i.e. MODEL)
        RLG=Float(I-1)/1000. -6.
        E=10.**RLG   ! Energy (MeV)
! NMAX is already defined in subr INCREASE_DATA_ARRAY and check < 5 GeV is done
       If(I.gt.NMAX) goto 7000
! Attention. NMAX is the same for all material components, it is
* supposed that SL_NLS code which is executed before IOTA code
* was stopped at the same energy for all components
*
         Call SPAR0(ZPR, APR, E, dEdxTOTS, dEdxNS, JJ)
*
* total stopping power
          dEdx(IK,i)  =dEdxTOTS
* nuclear stopping
          dEdxN1(IK,i)=dEdxNS
*
* calculate total path from 1/ dE/dx integration (lin-lin)
            TmpE(i)=E
                                dxdE(i)=0.0
            if(dEdxTOTS.ne.0.0) dxdE(i)=1./dEdxTOTS
       if(i.eq.1) then
                  Rp1(IK,i)   =0.0
                  else
       Rp1(IK,i)=Rp1(IK,i-1)+(TmpE(i)-TmpE(i-1))*
     #           (dxdE(i)+dxdE(i-1))/2.
                 endif
*
*
        Enddo    !  Do I1=1,999999999
 7000   continue
*
* write results
* units for printing MeV/mg/cm2, cm
c                   prc=1./(1000.*RO)
                   prc=1.
        Do i=1,NMAX
        Write(21,90004) TmpE(i), prc*(dEdx(IK,i)-dEdxN1(IK,i)) ,
     #                            prc*dEdxN1(IK,i), Rp1(IK,i)
        Enddo
        close(21)
*
*
        Enddo    !  Do IK=1,NK1
*       ----------------------
*
*
               iiii=1
c        if(iiii.eq.1) stop
        Return
c
c
90001   Format(' Stopping power from SPAR (MCNPX)'/
     #  ' projectile ',f6.1,f8.2,'       IK =',i3,
     #  /' medium     ',f6.1,f8.2,'    condensed   JJ =',i2,
     #  /' density =',f8.3,' g/cm3     n0 =',1pe12.5,' cm-3'
     #  /' mean ioniz pot = ',0pf8.2,' eV')
90002   Format(/' units  E [MeV],  dE/dx [MeV/mg/cm2],  R [cm] ')
90003   Format(/'     E        dE/dx(electr)  dE/dx(nucl)',
     #  '       R (total) '/ 61('-'))
90004   Format(1pe12.5,2e14.5,e17.5)
c
91000   Print 91001
91001   Format(/' Gaseous target is not allowed ! '
     #  /20x,'Change subr DEDXSPAR  !!')
        print *,'                        press any key...'
        pause
        stop
c
        End
*
************************************************************************
```

```
*
         Subroutine  DEDX_ZIEGLER_READ(IK, IR, RO0, Ex, dEdx, Rp, NdE,
     #   dEdxN)
*        ------------------------------------------------------------
c
c Ex,  dEdx  - energy and total stopping power   (Ziegler/SRIM)
c      dEdxN - nuclear stopping power            (Ziegler/SRIM)
c
         parameter (maxc=21)
         Character C12*12,C33*33,C7*7,CID3*3,CID2*2, C17*17
         Character C6*6, C80*80
         Common/Z2A2/ZT(MAXC),AT(MAXC),FR(MAXC),ED(MAXC),NK
         Common/Basic/Z1,A1,Z2,A2,AW,ROOO,RN0
         Dimension Rp(1501), Ex(1501), dEdx(1501)
         Dimension dEdxN(1501)
         Dimension ExTMP(1501),dEdxETMP(1501), dEdxE(1501)
         Dimension ExTMP2(1501),dEdxETMP2(1501)
         If(IK.eq.1) then
                     Zini=Z1
                     Aini=A1
                     else
                     Zini=ZT(IK-1)
                     Aini=AT(IK-1)
                     endif
                     IZT=Ifix(Zini+0.001)
                     IAT=Ifix(Aini+0.001)
*
* find "Ion ="
*
         Do i=1,111111111
         read(IR,'(a6)',end=90001)C6
         If(C6.eq.' Ion =')             then
         backspace ir
         read(ir,'(a80)')C80
*
* Get Z,A from SRIM output
         Call ZieION(C80,Zzie,Azie)
*
*
                                        goto 11
                                        endif ! If(C6.eq.' Ion =')
         Enddo
*
*
* find "Target density"
*
 11      Do i=1,111111111
         read(IR,'(a17)',end=90005)C17
         If(C17.eq.' Target Density =') then  ! 1)
         backspace ir
         read(ir,'(t18,e12.0)')RO
            if(RO.ne.RO0) then                ! 2)
            print 1,IZT,IAT,RO0,RO
  1         format(1x,'  W A R N I N G !'/1x,
     #      'for ',i3,i4,' RO(input)=',g12.5,' BUT  Ziegler(RO)=',
     #      g12.5/1x,'The last one will be used to recover dEdx')
            print *,'                   press any key to continue'
            pause
                         endif                 ! 2)
         goto 2
                                        endif  ! 1)
         Enddo
*
* find line "Stopping units"
*
  2      do i=1,111111111
         read(IR,'(a17)',end=90006)C17
         If(C17.eq.' Stopping Units =') then
             backspace ir
            read(IR,'(a33)',end=90006)C33
            if(C33.ne.' Stopping Units =  MeV / (mg/cm2)') then
            Print *,' Units of stopping power must be MeV/mg/cm2 !'
            print *,'                          press any key...'
            pause
            stop
                                                     endif
         goto 3
                                     endif
         enddo
*
* skip 4 lines
  3      do i=1,4
         read(IR,*,end=90007)
         enddo
*
         read(IR,'(a12)')C12
         if(C12.ne.'----------- ') Call error('----',12)
*
* Begin reading
*
* transformation of SRIM data
*
         i=1
           Ex(1)=0.0
           dEdx(1)=0.0
           Rp(1)=0.0
           dEdxN(1)=0.0
c
c
         Do iii=1,1111111       ! ---c y c l e begins----
         read(IR,'(a7)')C7
         if(C7.eq.'-------') goto 2000
         backspace IR
         read(IR,'(e7.0,1x, a3,   e12.0,   e11.0,    e8.0,1x, a2)')
     #            EN,       CID3, dEdx_el, dEdx_nuc, RRR,       CID2
         if(CID3.ne.'eV '.and.CID3.ne.'keV'.and.CID3.ne.'MeV'.
     #   and.CID3.ne.'GeV') call ERROR(' ev?',11111)
*
* energy ---> MeV
                            ENERGY_MULT=1.0
         if(CID3.eq.'eV ')  ENERGY_MULT=1.E-06
         if(CID3.eq.'keV')  ENERGY_MULT=1.E-03
         if(CID3.eq.'GeV')  ENERGY_MULT=1.E+03
*
         if(CID2.ne.'A '.and.CID2.ne.'um'.and.CID2.ne.'mm'
     #   .and.CID2.ne.'cm'.and.CID2.ne.'m ') call ERROR(' um?',11111)
*
* all units ---> cm
                           RANGE_MULT=1.
         if(CID2.eq.'A ')  RANGE_MULT=1.E-08     ! 1 A=10**-10 m
         if(CID2.eq.'um')  RANGE_MULT=1.E-04     ! 1 um=micron= 10**-6 m
         if(CID2.eq.'mm')  RANGE_MULT=0.1        ! 1 mm=0.001 m
         if(CID2.eq.'m ')  RANGE_MULT=100.       ! 1 m =100 cm
* 1)
* MeV/(mg/cm2) ----> MeV/(g/cm2)
         dEdx_sum= 1000.*( dEdx_el+dEdx_nuc)
         dEdx_nuc2= 1000.*dEdx_nuc
* 2)
* MeV/(g/cm2) ----> MeV/cm
         dEdx_sum= dEdx_sum*RO
         dEdx_nuc2= dEdx_nuc2*RO
*
         Energy=EN* ENERGY_MULT
         Range= RRR* RANGE_MULT
*
* fill arrays
         i=i+1
                  if(i.gt.1501)  Call Error('DEDX',1501)
         Ex(i)=Energy
         dEdx(i)=dEdx_sum
         dEdxN(i)= dEdx_nuc2
         Rp(i)=Range
         Enddo                  ! ---c y c l e finished----
         call error('_Zie',11111)
*
* end of reading
2000     NdE=i
*
*
* Recalculate stopping if Z=Z', but A.ne.A'
*
         IZzie=Ifix(Zzie+0.001)
         If(IZzie.ne.IZT) then
         Print 3000,Zini,Aini,Zzie,Azie
3000     Format(' SRIM data failure !!!'/1x,' Current incident ion is ',
     #   f5.1,f8.2/20x,'BUT'/' SRIM data are prepared for ',f5.1,f8.2)
         print *,'                           press any key...'
         pause
         stop
                          endif
* no recalculation if A-A' < 1
         If( abs(Azie-Aini).lt.1.0) Return
*
         If(NK.ne.1) then
         Print 3100
3100     Format(70('*')/'*',25x,'W A R N I N G !',t70,'*'/'*',
     #   4x,' for compounds stopping power will not be recalculated',
     #   t70,'*'/70('*')//1x,
     #   'press any key to continue  calculations ...')
         pause
         Return
                     endif
*
* Recalculation
*
* i) electronic stopping
*
         Do i=1,NdE
         ExTMP(i)=Ex(i)*Aini/Azie
         dEdxETMP(i)=dEdx(i)-dEdxN(i)
         Enddo
c
         if(Aini-Azie) 4000,4000,5000
c
c A' < A ==> lack of high energy data, get them by interp two last points
c
4000        Do i=1,NdE
            m=i
            if(ExTMP(NdE).le.Ex(i)) goto 4010
            Enddo
         call Error('4010',NdE)
4010         X1=ExTMP(NdE-1)
             X2=ExTMP(NdE)
             Y1=dEdxETMP(NdE-1)
             Y2=dEdxETMP(NdE)
             INTS=2
         j=NdE
         Do i=m,NdE
         j=j+1
                  if(j.gt.1501)  Call Error('j :1',1501)
         ExTMP(j)=Ex(i)
         WL=ExTMP(j)
         Call INTERMEDIATE(X1,X2,Y1,Y2,WL,INTS,RESINT)
         dEdxETMP(j)=RESINT
         Enddo
         NdETMP=j
         goto 7000
```

```
c
c A' > A ==> lack of low energy data, get them by interp two first points
c
5000        Do i=2,NdE
            m=i-1
            if(ExTMP(2).le.Ex(i)) goto 5010  ! "2" is first non-zero value
            Enddo
         call Error('5010',NdE)
5010         X1=ExTMP(2)
             X2=ExTMP(3)
             Y1=dEdxETMP(2)
             Y2=dEdxETMP(3)
             INTS=2
         j=1
         ExTMP2(j)=0.0
         dEdxETMP2(j)=0.0
c
         Do i=2,m
         j=j+1
         ExTMP2(j)=Ex(i)
         WL=ExTMP2(j)
         Call INTERMEDIATE(X1,X2,Y1,Y2,WL,INTS,RESINT)
         dEdxETMP2(j)=RESINT
         Enddo
c
           Do i=2,NdE
           j=j+1
                  if(j.gt.1501)  Call Error('j :2',1501)
           ExTMP2(j)=ExTMP(i)
           dEdxETMP2(j)=dEdxETMP(i)
           Enddo
         NdETMP=j
           Do i=1,NdETMP
           ExTMP(i)=ExTMP2(i)
           dEdxETMP(i)=dEdxETMP2(i)
           Enddo
c
c recover old energy grid
c
7000     Do i=1,NdE
         E=Ex(i)
         Call ARRAYINT(0,ExTMP,dEdxETMP, NdETMP, E, 2, RES)
         dEdxE(i)=RES
         Enddo
*
* ii) nuclear stopping
*
       Do i=2,NdE
       E=Ex(i)
       Call dEdXnuc_recalc(Zzie,Z2,Azie,A2,E,sNzie)
       Call dEdXnuc_recalc(Zini,Z2,Aini,A2,E,sNini)
       If(sNzie.eq.0.0) Call error('snzi',0)
c
       dEdxN(i)=dEdxN(i)*sNini/sNzie
       Enddo
*
* total stopping
                           open(22,file='renorm.sto') ! test printing
       Write(22,7500)Zini,Aini,Zzie,Azie,Z2,A2
7500   Format(' Incident ion  ', f5.1,f8.2/
     # ' SRIM data for ',f5.1,f8.2/25x,' Target ',f5.1,f8.2//
     #' Energy [MeV],   Stopping power [MeV/cm]'//
     #'     E         dE/dx(elec)     dE/dx(nucl) '/44('-'))
c
       Do i=1,NdE
       dEdx(i)=dEdxN(i)+dEdxE(i)
       Write(22,'(1pe11.4,e14.3,e16.3)')Ex(i),dEdxE(i),dEdxN(i)
       Enddo
c
```

```
       Return
c
c
90001    Print *,'------- End of SRIM output file ------'
         Print *,'  Phrase "Ion =" is not found'
         print *,'                         press any key...'
         pause
         stop
90005    Print *,'-------  End of SRIM output file ------'
         Print *,'  Phrase "Target density" is not found'
         print *,'                         press any key...'
         pause
         stop
90006    Print *,'-------  End of SRIM output file ------'
         Print *,'  Phrase "Stopping units" is not found'
         print *,'                         press any key...'
         pause
         stop

90007    Print *,'-------  End of SRIM output file ------'
         Print *,'               UNEXPECTED !'
         print *,'                         press any key...'
         pause
         stop
         End
*
************************************************************************
*
       Subroutine DISPERS(KEY)
*      ----------------------
       Common/SUNDI/DIVel1(405),DIFRENKEL(405),DIVel5
       Common/SUN/Vel1L(405),Tmean1,Vel2Tm(405),Vel3Pl(405),
     #           R1,R1p,Tplay1,Vel4(405),Nel(405)
       Common/MEMORY/NFRENKEL(405)
       Common/SUN2/ZV5,Vel5,NN(405)
*
       Common/DIMEMO/DV1(405),DFREN(405),DV5
*
       If(Key.eq.2) goto 2
* store data before the history
       Do i=1,405
       DV1(i)=Vel1L(i)
       DFREN(i)=float(NFRENKEL(i))
       Enddo
       DV5=Vel5
       Return
* sum
  2     continue
        SUMV1=0.0
        SUMF=0.0
        Do i=1,405
        SUMV1=SUMV1+Vel1L(i)-DV1(i)   ! = result for the current history
        SUMF= SUMF +float(NFRENKEL(i))-DFREN(i)
          DIVel1(i)    =DIVel1(i)   +SUMV1**2
          DIFRENKEL(i) =DIFRENKEL(i)+SUMF**2
        Enddo
*
        DIVel5=DIVel5+ (Vel5-DV5)**2
        Return
        End
*
************************************************************************
*
        Subroutine END_OF_CALCULATION
*       ----------------------------
        Write(33,1)
  1     Format(33(1h*),'e*n*d',35(1h*))
        Return
        end
```

```
*
*************************************************************************
*
         Subroutine Error(phrase,I)
*        -------------------------
         Character phrase*4
         write(6,1)phrase,I
1        Format(1x,'       ???   ',a4/1x,
     *   'ERROR !!!          number=',i7)
         print *,'                     press <Enter>...'
         pause
         stop
         end
*
*************************************************************************
*
         FUNCTION F(X)
*        -------------
* Scattering f(x)= f(t^1/2) function
*
         Common/PARAM01/IDSDT
         Common/JUPITER/ALP1,ALP2,BET1,BET2,BET3,BET4
*
         If(IDSDT.eq.0)  then
*
         F=ALP1*(X**BET1)*(   ( 1.+( ALP2*(X**BET2) )**BET3)**BET4  )
         Return
                         else
*
* Burenkov et al function
*
         F=FBUR(X)
         Return
                         endif
         End
*
*************************************************************************
*
         FUNCTION FBUR(X)
*        ---------------
* Burenkov et al function
         If(X.le.0.04) then
           RJ=1.
           A1=0.088
           A2=1.
           A3=0.2
           goto 1
           endif
         If(0.04.lt.X  .and.  X.le.1.0) then
           RJ=1.
           A1=0.23
           A2=1.3
           A3=0.4
           goto 1
           endif
         If(1.0.lt.X  .and.  X.le.40.) then
           RJ=0.0
           A1=0.4563
           A2=0.0
           A3=0.3
           goto 1
           endif
         If(X.gt.40.) then
           RJ=0.0
           A1=0.5
           A2=0.0
           A3=0.0
           endif
1        FBUR=A1*(X**(RJ/2.))*(X+A2)/(  (X+A3)**2 )
```

```
         Return
         End
*
************************************************************************
*
         Function FINDP(S80,C10)
*        -------------------------------
* Find phrase C10*XX (here XX=1) in line S80*80
* N = position of the last symbol of C10 in S80
* N = 0 if the phrase not found
*
         Character S80*80, C10*1
         NS=80
         NC=1
                           N=0
         Do 2000 I=1,NS
               i1=I
               i2=I+NC-1
               if(i2.gt.NS) goto 2000
         if(s80(I1:I2).eq.C10) goto 3000
2000     continue
         findp=float(N)
         return
*
3000     continue
         N=i2
         findp=float(N)
         return
         end
*
******************************************************************
*
         Subroutine FT12CHOICE(idsdt)
*        --------------------------
* Main formula
* f(x)=Lambda* x**(1-2m) [1+ {2*Lambda*x**(2*(1-m)}**q ]**(-1/q)
*
* Parameters are taken mainly from   W.Eckstein, "Computer Simulation
* of Ion-Solid Interactions"  Springer, Berlin, 1991, p.59, Table 4.8
*
*
* alp1 = L,  bet1 = (1-2*m),  alp2 = 2*L
* bet2 = 2*(1-m), bet3= q,    bet4 = -1/q
*
         Common/JUPITER/ALP1,ALP2,BET1,BET2,BET3,BET4
         Common/AUX1/rlambda,rm,q
*
         if(idsdt.eq.1) return   ! --> Burenkov et al function
         if(idsdt.eq.0) goto 10
         goto(888,888,888,888,888,888,888,888,888,10,
     #        11,12,13,14,15), idsdt
         goto 888
*
* Original Winterbon et al parameters
10       rlambda=1.309
         rm=1./3.
         q=2./3.
         goto 777
*
* Thomas-Fermi-Sommerfeld
11       rlambda=1.7
         rm=0.311
         q=0.588
         goto 777
*
* Bohr
12       rlambda=2.37
         rm=0.103
         q=0.570
         goto 777
*
* Lenz-Jensen
13       rlambda=2.92
         rm=0.191
         q=0.512
         goto 777
*
* Moliere
14       rlambda=3.07
         rm=0.216
         q=0.530
         goto 777
*
* U.Littmark, J.F.Ziegler, "Handbook of Range Distributions"
* Kr-C potential
15       rlambda=3.35
         rm=0.2328
         q=0.4445
*
*
777      alp1=rlambda
         bet1=1.-2.*rm
         alp2=2.*rlambda
         bet2=2.*(1.-rm)
         bet3=q
         bet4=-1./q
!
         idsdt=0
         Return
*
  888    Print 999,idsdt
  999    Format(' ERROR:    Parameters of f(t^1/2) are not known !'/
     #   '           idsdt=',i5,' is not correct.')
         Print *,'                    press any key...'
         pause
         stop
         End
*
************************************************************************
*
        Subroutine GET_FILE_NAMES(IU,NK,NF)
*       ---------------------------------
*
* To get NK-number file names (max 12 charactres) from input lines
* written in arbitrary order.
*
*    Any number of lines can be used to write NK file names
* Example (NK=7) (no "c" in real input):
c---------------e x a m p l e-----------------
c abc.dat    cde.dat
c   qweert.dat q.dat   w.w   www.dat  iuiu.inp
c----------------------------------------------
*    Any line beginning from * is the comment and will be omitted.
*    Reading stops when all NK names will be obtained.
*    Input line lenght is max 80 characters.
*
* IU     - number of input device
* NK     - number of names of the files to be read
* NF(NK) - name of files
*
        Character C80*80, NF*12, NFTMP*12
        Dimension NF(NK)
c
 400    Read(IU,'(a80)') C80
        if(C80(1:1).eq.'*') goto 400
          MK=0
 1      Do J=1,80
        i=J
        If(C80(j:j).ne.' ') goto 2
        Enddo
* only blank line left
        If(MK.gt.NK)  goto 90000
        If(MK.eq.NK)  return
* MK < NK
 500    Read(IU,'(a80)') C80
        if(C80(1:1).eq.'*') goto 500
        goto 1
*
 2      MK=MK+1
        If(MK.gt.NK)  goto 90000
        NFTMP='            '
           L=0
           Do M=i,80
           If(C80(m:m).eq.' ') goto 3
           L=L+1
           If(L.gt.12) goto 91000
           NFTMP(L:L)=C80(M:M)
           C80(M:M)=' '
           Enddo
 3      NF(MK)=NFTMP
        If(MK.eq.NK) Return
        goto 1
*
90000   Print *,'***** Subr.GET_FILE_NAMES'
        Print *,' Number of files MK exceeds NK(input) !'
        print *,'                         press any key...'
        pause
        stop
91000   Print *,'***** Subr.GET_FILE_NAMES'
        Print *,' One of the file name exceeds 12 characters !'
        Print *,'               CHANGE INPUT   '
        print *,'                         press any key...'
        pause
        stop
*
        END
*
************************************************************************
*
      Subroutine GRAPH(Y,F,N3,IALOG)
*     ------------------------------
* IALOG .ne.0 : Log scale for Y
      COMMON/MAIIPR/IPR1
      DIMENSION Y(N3),F(N3),F1(11250)
      INTEGER A1(40),B1,C1,D1,D2,D3,D4,D5,D6
      DATA B1,C1,D1,D2,D5,D6/1H*,1H ,1H!,1H-,1HI,1H./
       IF(N3.GT.11250)then
       Print*,' BAD operation in the code GRAPH'
       STOP
       endif
      N1=1
      W=F(1)
      DO 1 L=1,N3
      IF(ABS(W).GT.ABS(F(L)))GOTO 1
      W=F(L)
    1 CONTINUE
      IF(W.NE.0.)GOTO 70
      WRITE(IPR1,41)
   41 FORMAT(1X,' GRAPH:  maximal value identically equal to zero')
      RETURN
   70 IF(IALOG.NE.0)GOTO 199
      WRITE(IPR1,60)
   60 FORMAT(/3X,'   X           Y        ',4X,4('.........!'))
      T=40./ABS(W)
      DO 300 I=1,N3
  300 F1(I)=F(I)
      GOTO 7
  199 WMIN=1.E+10
      DO 200 I=1,N3
      IF(F(I).NE.0..AND.F(I).LT.WMIN)WMIN=F(I)
  200 CONTINUE
      IMAX=-8
  210 IF(W.LE.10.**IMAX)GOTO 220
      IMAX=IMAX+1
      GOTO 210
  220 IMIN=IMAX
  230 IF(WMIN.GE.10.**IMIN)GOTO 240
      IMIN=IMIN-1
      GOTO 230
  240 R10=10.**IMIN
      IF(IMIN.EQ.IMAX)GOTO 250
      WW1=FLOAT(IMIN)
      WW=FLOAT(IMAX-IMIN)
      R20=10.**IMAX
      WRITE(IPR1,62)R10,R20
   62 FORMAT(1X,5X,'semilogarithmic net        '/3X,23X,E8.2,32X,E8.2)
      GOTO 260
  250 WW1=ALOG10(WMIN)
      WW=ALOG10(WMAX)-ALOG10(WMIN)
  260 DO 270 I=1,N3
      FF=F(I)
      IF(FF.EQ.0.)FF=R10
  270 F1(I)=ALOG10(FF)-WW1
      WRITE(IPR1,60)
      T=40./WW
    7 CONTINUE
      DO 3 I=1,N3
      X=F1(I)
      N=INT(X*T)
      IF(I.EQ.N3)GOTO 105
      IF(I.EQ.N1*10)GOTO 107
      D4=C1
      GOTO 108
  105 D4=D6
      GOTO 108
  107 D4=D2
      N1=N1+1
  108 N2=1
      DO 4 J1=1,40
      IF(J1.EQ.20)GOTO 110
      IF(J1.EQ.1.OR.J1.EQ.40)GOTO 24
      D3=D4
      GOTO 25
   24 D3=D5
   25 CONTINUE
      GOTO 111
  110 D3=D1
      N2=N2+1
  111 CONTINUE
      IF(J1.EQ.N)GOTO 101
      A1(J1)=D3
      GOTO 102
  101 A1(J1)=B1
  102 CONTINUE
    4 CONTINUE
      WRITE(IPR1,61)Y(I),F(I),A1
   61 FORMAT(1PE12.5,1X,E12.5,5X,40A1)
    3 CONTINUE
      RETURN
      END
*
************************************************************************
*
       Subroutine INCREASE_DATA_ARRAY
*      ------------------------------
       parameter (maxc=21,maxc1=22)
       Common/Basic/Z1,A1,Z2,A2,AW,RO,RN0
       Common/Z2A2/ZT(MAXC),AT(MAXC),FR(MAXC),ED(MAXC),NK
*
       Common/dEdxMain01/Ex01(MAXC1,1501), dEdx01(MAXC1,1501),
     # Rp01(MAXC1,1501), dEdxN01(MAXC1,1501),NdE01(MAXC1)
*
       Common/Lindhard01/Energy01(MAXC1,MAXC,1701),
     # dSdX01(MAXC1,MAXC,1701), dEdX_Nuc01(MAXC1,MAXC,1701),
     # dEdX_Nuc_0Ed01(MAXC1,MAXC,1701),
     # dEdX_Nuc_Ed_Tmax01(MAXC1,MAXC,1701),
     # CST01(MAXC1,MAXC,1701), NSL01(MAXC1,MAXC)
*
* corresponds to /dEdxMain01/
       Common/SKY1/dEdx(MAXC1,10002),dEdxN1(MAXC1,10002),
     # Rp1(MAXC1,10002)
* corresponds to /Lindhard01/
       Common/STARS1/ dSdX1(MAXC1,MAXC,10002),
     # dEdXNuc1(MAXC1,MAXC,10002),
     # dEdXNuc0Ed1(MAXC1,MAXC,10002), dEdXNucEdTm1(MAXC1,MAXC,10002),
     # CST1(MAXC1,MAXC,10002), NMAX
*
       Common/istop/istop
       Dimension TMP1(1701),TMP2(1701),TMP3(1701),TMP4(1701),
     #           TMP5(1701),TMP6(1701)
*
* =1 test printing, 0 no
       itestxs=0
*
       If(itestxs.ne.0) open(21,file='test.xs')
       If(itestxs.ne.0) Write(21,1000)
1000   Format('   I       E        dSdX      dEdX_Nuc',
     # '    dEdX_Nuc     dEdX_Nuc      CST'/
     # '                               (tot)      [0-Ed]',
     # '       [Ed-Tmax]    (barn)'/77('-'))
*
*
      NK1=NK+1
*
*
* General arrays preparation
*****************************
*
*
                               if(istop.ne.-1) goto 3001
* Ziegler data
* ============
*
      Do 2000 IK=1,NK1
*     -------------
* IK - id of the moving ion
*
        NdE01tmp=NdE01(IK)
        Do II=1, NdE01tmp
        TMP1(II)=Ex01(IK,II)
        TMP2(II)=dEdx01(IK,II)
        TMP3(II)=dEdxN01(IK,II)
        TMP4(II)=Rp01(IK,II)
        Enddo
*
        Emaxtmp=Amin1(5000.,TMP1( NdE01tmp) )
        Do I=1,999999999
*
* J=(lg(E)+6)*1000+1  means that energy can changes from 10**-6 MeV
* If 10002 array dimension will be increased change also 1000 below
* and in other subroutines (i.e. MODEL, DEDX_SPAR)
        RLG=Float(I-1)/1000. -6.
        E=10.**RLG
* E > 5000 MeV is not considered
        If(E.gt.Emaxtmp) goto 2000
```

```
! Attention. NMAX is the same for all material components, it is
* supposed that SL_LNS code which is executed before IOTA code
* was stopped at the same energy for all components
        NMAX=I
        If(NMAX.gt.10002) Call Error('NMAX',10002)
        Call ARRAYINT(0,TMP1,TMP2, NdE01tmp,E,2, RES1)
        Call ARRAYINT(0,TMP1,TMP3, NdE01tmp,E,2, RES2)
        Call ARRAYINT(0,TMP1,TMP4, NdE01tmp,E,2, RES3)
          dEdx(IK,NMAX)  =RES1
          dEdxN1(IK,NMAX)=RES2
          Rp1(IK,NMAX)   =RES3
        Enddo    !  Do I1=1,999999999
 2000   Continue !  Do IK=1,NK1
*
*
*
* Cross-sections obtained by Lindhard approach
* ============================================
*
 3001                         continue
*
      Do 3 IK=1,NK1
*     -------------
*
* IK - id of the moving ion
* MK - id of the material component
*
      Do 2 MK=1,NK
*     ------------
*
      If(itestxs.ne.0) Write(21,11) IK,MK
11    Format('Interaction (ion id)  ',i2,' +',i3)
*
        NSL01tmp=NSL01(IK,MK)
        Do II=1, NSL01tmp
        TMP1(II)=Energy01(IK,MK,II)
        TMP2(II)=dSdX01(IK,MK,II)
        TMP3(II)=dEdX_Nuc01(IK,MK,II)
        TMP4(II)=dEdX_Nuc_0Ed01(IK,MK,II)
        TMP5(II)=dEdX_Nuc_Ed_Tmax01(IK,MK,II)
        TMP6(II)=CST01(IK,MK,II)
        Enddo
*
      Emaxtmp=Amin1(5000.,TMP1( NSL01tmp) )
      Do I=1,999999999
*
* J=(lg(E)+6)*1000+1  means that energy can changes from 10**-6 MeV
* If 10002 array dimension will be increased change also 1000 below
* and in other subroutines ( MODEL, DEDX_SPAR, ...)
        RLG=Float(I-1)/1000. -6.
        E=10.**RLG
* E > 5000 MeV is not considered
        If(E.gt.Emaxtmp) goto 2
        NMAX=I
        If(NMAX.gt.10002) Call Error('NMAX',10002)
*
*
        Call ARRAYINT(0,TMP1,TMP2, NSL01tmp,E,2, RES1)
        Call ARRAYINT(0,TMP1,TMP3, NSL01tmp,E,2, RES2)
        Call ARRAYINT(0,TMP1,TMP4, NSL01tmp,E,2, RES3)
        Call ARRAYINT(0,TMP1,TMP5, NSL01tmp,E,2, RES4)
        Call ARRAYINT(0,TMP1,TMP6, NSL01tmp,E,2, RES5)
       If(itestxs.ne.0) Write(21,1)I,E,RES1,RES2,RES3,RES4,RES5
1      Format(i5,g12.5,5g12.5)
          dSdX1(IK,MK,NMAX)        =RES1
          dEdXNuc1(IK,MK,NMAX)     =RES2
          dEdXNuc0Ed1(IK,MK,NMAX)  =RES3
          dEdXNucEdTm1(IK,MK,NMAX) =RES4
* barn ---> cm2
```

```
          CST1(IK,MK,NMAX)         =RES5*1.e-24
       EndDo
*
*
   2   Continue
   3   Continue
*
        If(itestxs.ne.0) close(21)
        Return
        End
*
************************************************************************
*
        Subroutine INDICATOR(IK,T,IDE)
*       ----------------------------
        Common/EKIN1/EKIN(405),HEK,MEK ! MEK can be less than 405
*
        If(IK.ne.1) goto 2000
        Do I=1,405
        IDE=I
        If(T.le.EKIN(i)) Return
        Enddo
*
        Print *,'SUBR. INDICATOR   T SEEMS STRANGE =',T
        Call ERROR('IDE ',00000)
* to check IDE for non primary particle
2000    If(IDE.gt.0.and.IDE.le.405)   Return
* IDE <= 0 or > 405
*
        Print *,'SUBR. INDICATOR:   E R R O R !   IDE =',IDE
        Call ERROR('IDE ',00000)
        End
*
************************************************************************
*
       Subroutine Initialize_and_Zeroize(T00)
*      ------------------------------------
       parameter (maxc1=22)
       Common/Basic/Z1,A1,Z2,A2,AW,RO,RN0
       Common/SKY1/dEdx(MAXC1,10002),dEdxN1(MAXC1,10002),
     # Rp1(MAXC1,10002)
*
       Common/urand1/iyg
       Common/SUN/Vel1L(405),Tmean1,Vel2Tm(405),Vel3Pl(405),
     #           R1,R1p,Tplay1,Vel4(405),Nel(405)
       Common/Interr/Interr1
       Common/MOON/XX(105),DD(105),DDF(105),TK(105),HST
       COMMON/RECSTORE/TKIN(50003),ZZ(50003),NKO(50003),INDIC(50003),
     #                NREC
       COMMON/MEMORY/NFRENKEL(405)
       Common/CUT1/rmaxpl,itot,icut
       Common/EKIN1/EKIN(405),HEK,MEK ! MEK can be less than 405
       Common/SUN2/ZV5,Vel5,NN(405)
       Common/SUNDI/DIVel1(405),DIFRENKEL(405),DIVel5
               iyg=1
       T0=T00
*
*
*                                             Ziegler Range
       JE=Ifix((alog10(T0)+6.)*1000.)+1
                  If(JE.le.0.or.JE.gt.10002) Call Error(' JE ',11111)
*
* Ziegler Range for projectile
       RZ=Rp1(1,JE)
*
       RFULL=(1.+0.333333*A2/A1)*RZ     !  For effective A2
*
       HST=RFULL/20.
       XXX=-HST
```

```
*
       Do i=1,105
       XXX=XXX+HST
       XX(i)=XXX
       DD(i)=0.0
       DDF(i)=0.0
       TK(i)=0.0
       enddo
*
*
         MEK=200       ! should be less than 405
         HEK=T0/FLOAT(MEK)
         EEE=-HEK+0.5*HEK
*
         Do i=1,405
         EEE=EEE+HEK
         EKIN(i)=EEE
         enddo
*
* indent for Z where the primary energy lost = 5 %
         ZV5= 0.05*T0/dEdx(1,JE)
         Vel5=0.0
         DIVel5=0.0
*
* Zeroize
* -------
*
        Do i=1,405
        Nel(i)   =0
        NN(i)    =0
        Vel1L(i) =0.
        DIVel1(i)=0.
        Vel2Tm(i)=0.
        Vel3Pl(i)=0.
        Vel4(i)  =0.
        Enddo
*
        Tmean1=0.
        R1=0.
        R1p=0.
        Tplay1=0.0
*
        Interr1=0
        Icut=0
        Itot=0
*
        Do i=1,50003
        Tkin(i)=0.0
        ZZ(i)=0.0
        NKO(i)=0
        enddo
        NREC=0
*
        Do i=1,405
        NFRENKEL(i)=0
        DIFRENKEL(i)=0.0
        Enddo
C
        Return
        End
*
*************************************************************************
*
          Subroutine INTERMEDIATE(X1,X2,Y1,Y2,WL,INTS,RESINT)
*         ---------------------------------------------------
          If(INTS.eq.1) goto 1
          If(INTS.eq.2) goto 2
          If(INTS.eq.3) goto 3
          If(INTS.eq.4) goto 4
```

```
           If(INTS.eq.5) goto 5
           Print *,
     #     'This interpolation scheme ',INTS,' is not supported'
           Print *,'                        press any key...'
           pause
           Stop
* Histogr
1          RESINT =  Y1
           Return
* Lin-lin
2          Call REGR1(X1,X2,Y1,Y2,AAA,BBB)
           RESINT =  AAA * WL + BBB
           Return
* Lin(lg)-lin
3          aX1=alog10(x1)
           aX2=alog10(x2)
           aY1=y1
           aY2=y2
           aWL=alog10(wl)
           Call REGR1(aX1,aX2,aY1,aY2,AAA,BBB)
           RESINT =  AAA * aWL + BBB
           Return
* Lin-lin(lg)
4          aX1=x1
           aX2=x2
           aY1=alog10(y1)
           aY2=alog10(y2)
           aWL=wl
           Call REGR1(aX1,aX2,aY1,aY2,AAA,BBB)
           RESINT = 10.**( AAA * aWL + BBB )
           Return
* Lin(lg)-lin(lg)
5          aX1=alog10(x1)
           aX2=alog10(x2)
           aY1=alog10(y1)
           aY2=alog10(y2)
           aWL=alog10(wl)
           Call REGR1(aX1,aX2,aY1,aY2,AAA,BBB)
           RESINT = 10.**(   AAA * aWL + BBB  )
           Return
           End
C
C*********************************************************************
C
          Subroutine REGR1(x1,x2,y1,y2,a,b)
          Rr=x2-x1
          A=(y2-y1)/rr
          B=(y1*x2-x1*y2)/rr
          Return
          End
*
**********************************************************************
*
       Subroutine LINDINT(T00)
*      ----------------------
*
       parameter (maxc=21,maxc1=22)
       Common/SKY1/dEdx(MAXC1,10002),dEdxN1(MAXC1,10002),
     # Rp1(MAXC1,10002)
*
       Common/STARS1/ dSdX1(MAXC1,MAXC,10002),
     # dEdXNuc1(MAXC1,MAXC,10002),  dEdXNuc0Ed1(MAXC1,MAXC,10002),
     # dEdXNucEdTm1(MAXC1,MAXC,10002), CST1(MAXC1,MAXC,10002), NMAX
       Common/Edmin/Edmin ! It is not minimal displacement energy
       Common/BASIC/Z1,A1,Z2,A2,AW,RO,RN0
       Common/Z2A2/ZT(MAXC),AT(MAXC),FR(MAXC),ED(MAXC),NK
       Common/SUN/Vel1L(405),Tmean1,Vel2Tm(405),Vel3Pl(405),
     #           R1,R1p,Tplay1,Vel4(405),Nel(405)
*
```

```
       T=T00
       X=0.0
*
       IK=1  ! primary particle
*
*
*                                            Ziegler Range
       JE=Ifix((alog10(T)+6.)*1000.)+1
                   If(JE.le.0.or.JE.gt.10002) Call Error(' JE ',11111)
*
* Ziegler Range for projectile
       RZ=Rp1(1,JE)
*
       RFULL=(1.+0.333333*A2/A1)*RZ     !  For effective A2
*
        DO 1000 ILIL=1,11111111
* Get "energy indicator number" for current T
*   IDE corresponds only to the energy of primary particle (IK=1)
       Call INDICATOR(IK,T,IDE)
* general array index
       JE=Ifix((alog10(T)+6.)*1000.)+1
                   If(JE.le.0.or.JE.gt.10002) Call Error(' JE ',11111)
*
c Define STEP for x(STEP) and dT (DT)
c (energy step is 0.01 from current energy of particles)
         STOPPING=dEdx(IK,JE)
         DT=0.01*T
         STEP=DT/STOPPING
*
         dSdx=0.0
         Do MK=1,NK
         dSdx=dSdx+dSdx1(IK,MK,JE)*FR(MK)
         Enddo
*
         Vel4(IDE)=Vel4(IDE)+dSdx*STEP
cc         Write(551,*)'T,STEP,VEL4,DT=',T,STEP,VEL4,DT
         T=T-DT
*
* stop option
         If(T.lt.EDmin) goto 2000
         X=X+STEP
***      if(X.gt.RFULL)  goto 2000
*
*
1000     Continue
                                              call error('str ',1000)
2000     Continue
*
         Return
         End
*
************************************************************************
*
        Subroutine MDPARAM(IZ0,NOTE)
*       --------------------------------------------
        Character NOTE*50
        Common/MERCURY/CMD(5),Ecrit,MD
*
* Here are default parameters for efficiency obtained from MD
* calculations
*
* General formula for efficiency :  eff=C(1)*eMD**C(2) + C(3)*eMD
*
*  introduce
*            a) C(1)-C(3)
*            b) C(4):   eMD(keV) the lowest energy where effic=const
*            c) C(5):   eMD(keV) the highest energy where effic=const
*            d) Ecrit (keV):   eMD < Ecrit: MD;  eMD > Ecrit: BCA
*
```

```
* Note. eMD is not the incident ion energy, it is equal
*       approximately to the damage energy in the NRT formula
*
* Identification only for Z
*
        If(IZ0.eq.26) goto 26
        If(IZ0.eq.29) goto 29
        If(IZ0.eq.74) goto 74
        goto 99999
c
c Iron
c ----
c
  26    NOTE='from Stoller '
        CMD(1) = 0.5608
        CMD(2) =-0.3029
        CMD(3) = 3.227e-03
        CMD(4) = 0.0
        CMD(5) = 40.0
        Ecrit  = 40.0
        return
c
c Copper
c ------
c
  29    NOTE='from Caturla et al '
        CMD(1) = 0.7066
        CMD(2) =-0.437
        CMD(3) = 2.28E-03
        CMD(4) = 0.451
        CMD(5) = 20.00
        Ecrit  = 20.00
        return
c
c Tungsten
c --------
c
  74    NOTE='from Caturla et al '
        CMD(1) = 1.0184
        CMD(2) =-0.667
        CMD(3) = 5.06E-03
        CMD(4) = 1.0
        CMD(5) = 31.02
        Ecrit  = 31.02
        return
*
*
*
99999   Print 1,IZ0
        Write(33,1)IZ0
    1   Format(/' Default parameters for calculation of efficiency',
     #  ' of defect production '/'           according to MD',
     #  ' are ABSENT for Z=',i3,'  !!!'/20x,'Check Subr. MDPARAM !'/)
        Print *,'Chiao !'
        Print *,'                       press any key...'
        pause
        stop
        end
*
************************************************************************
*
       Subroutine MODEL(IK,T00,Z00,IDE)
*      -------------------------------
*
* IK  = 1 primary ion, 2,3,... =material component
* Example:
* dEdx(1,10002) is the full stopping power for the INITIAL particle +
*               composite material
* dEdx(2,10002) is the full stopping power for the FIRST component
*               of the material with the composite material  etc.
* dSdX1(1,1,10002) is the stopping power "considering defect production"
*               for the interaction of the INITIAL particle with
*               the energy transfer to FIRST material component (this
*               PKA is moving in the media with the effective Z,A (see
*               separate code SL_LNSXX, XX=ver))
* dSdX1(1,2,10002) is for the INITIAL particle and energy
*               transfer to the SECOND material component
* dSdX1(2,1,10002) is for the FIRST material component and energy
*               transfer to the same FIRST material component
* dSdX1(2,2,10002) is for the FIRST material component and energy
*               transfer to the SECOND material component
* etc.
* NK            is the number of material components (max 7)
*
*
      parameter (maxc=21,maxc1=22)
      Common/BASIC/Z1,A1,Z2,A2,AW,RO,RN0
      Common/Z2A2/ZT(MAXC),AT(MAXC),FR(MAXC),ED(MAXC),NK
      Common/Edmin/Edmin ! Edmin is not minimal displacement energy
      Common/STARS1/ dSdX1(MAXC1,MAXC,10002),
    # dEdXNuc1(MAXC1,MAXC,10002), dEdXNuc0Ed1(MAXC1,MAXC,10002),
    # dEdXNucEdTm1(MAXC1,MAXC,10002), CST1(MAXC1,MAXC,10002), NMAX
      Common/SKY1/dEdx(MAXC1,10002),dEdxN1(MAXC1,10002),
    # Rp1(MAXC1,10002)
      Common/Interr/Interr1
      Common/SUN/Vel1L(405),Tmean1,Vel2Tm(405),Vel3Pl(405),
    #           R1,R1p,Tplay1,Vel4(405),Nel(405)
      Common/MOON/XX(105),DD(105),DDF(105),TK(105),HST
      Common/MEMORY/NFRENKEL(405)
      Common/SUN2/ZV5,Vel5,NN(405)
      Common/SATURN/TcritLAB(MAXC1)
      Dimension CSS(MAXC)
*
      T=T00
      X=0.0  ! along the trajectory
      Z=0.0  ! along the z-axis
      ZD=Z00
      Tcrit=TcritLAB(ik)
* general array index
      JE=Ifix((alog10(T)+6.)*1000.)+1
                  If(JE.le.0.or.JE.gt.10002) Call Error(' JE ',11111)
*
*
*
        TplH1=0.0   ! Tplay for single history
*
  1     Continue      ! General loop
* Get "energy indicator number" for current T
*   IDE corresponds only to the energy of primary particle
*   and is not changed for any other generations
      Call INDICATOR(IK,T,IDE)
*
      If(IK.eq.1) NN(ide)=NN(ide)+1
*
*
* Estimation "elastic proton path"
* general array index
      JE=Ifix((alog10(T)+6.)*1000.)+1
                  If(JE.le.0.or.JE.gt.10002) Call Error(' JE ',11111)
* define total cross-section
      CSS(1)=FR(1)*CST1(IK,1,JE)
      If(NK.eq.1) goto 2
      DO MK=2,NK
      CSS(MK)=CSS(MK-1)+FR(MK)*CST1(IK,MK,JE)
      Enddo
 2    CSelT=CSS(NK)
                  If(CSelT.le.0.0) goto 2000
*
```

```
       dLel=1./(CSelT*RN0)  ! xsect in cm2
*
*
* STEP definition :
*
* Alternative  counting   DT0=dEdx(IK,JE)*STEP
! DT1 = only electronic component
       DT1=(   dEdx(IK,JE)-dEdxN1(IK,JE)  )
! DT2 = nucl stopp power for Tk < Ed
             dEdXNuc0Ed=0.0
             Do MK=1,NK
             dEdXNuc0Ed=dEdXNuc0Ed+dEdXNuc0Ed1(IK,MK,JE)*FR(MK)
             Enddo
       DT2=dEdXNuc0Ed
*
       DT3=DT1+DT2
* Free path/10.
       ST1=dLel/10.
                                                ST2=1.e+20
* Change of T = 0.05 due to el.loss and nucl.loss < Ed
       If(DT3.gt.0.0) ST2=0.05*T/DT3
       STEP=AMIN1(ST1,ST2)
*
*
* Play if ELASTIC interaction occurs (for the transferring of the
* energy > Ed)

c
        PREL=STEP/dLel
        PR0=RANDOM(0)
        IEL=0
        COST=1.0
*
        if(PR0.le.PREL)               then
*"Elastic" interaction occurs !
        IEL=1
*
* Define (play) the type of the component of the composite material
        PR00=RANDOM(0)
            If(NK.eq.1) then
            LK=1
            goto 3
            endif
        DO MK=1,NK
        LK=MK
        If(PR00 .le. CSS(MK)/CSelt) goto 3
        ENDDO
        print *,' It is not corr play:',CSS
  3     Continue
*
             Tmean=dEdXNucEdTm1(IK,LK,JE)/(CSelT*RN0)
        If(IK.eq.1 ) then
             Nel(IDE)=Nel(IDE)+1
             DISPLL=dSdx1(IK,LK,JE)/(CSelT*RN0)
             Vel1L(IDE)=Vel1L(IDE)+DISPLL
             Tmean1=Tmean+Tmean1
             Vel2Tm(IDE)=Vel2Tm(IDE)+CF(LK+1,LK,Tmean)   ! LK
                      endif
* play T of PKA basing on dS/dT above Ed
        TPVA=  TPKA(IK,LK,T,Tmean)
*
        if(IK.eq.1)  then
* define cos of the scattering
        CALL COS_DEFINE(IK,LK,T,TPVA,COST)
           Tplay1=Tplay1+TPVA
           Vel3Pl(IDE)=Vel3Pl(IDE)+CF(LK+1,LK,TPVA)     ! LK
                      endif
*
        If(T.le.Tcrit) goto 7000    !   MD
```

```
*
        T=T-TPVA
        TplH1=TplH1+TPVA   ! Tplay for single history
*
        If(TPVA.gt.Tcrit) then
        CALL STORE_IN(LK,TPVA,ZD,IDE)
* number of Frenkel pairs
        NFRENKEL(IDE)=NFRENKEL(IDE)+1
                          else  ! do not remember this particle
* MD
                          Frcut=0.0
        If(TPVA.gt.0.0)   FRcut=CF2(LK+1,LK,TPVA) +0.5
        NFRENKEL(IDE)=NFRENKEL(IDE)+Ifix(FRcut)
                          endif
*
        If(T.le.Tcrit) goto 7000    !   MD
*
*
        If(ZD.le.ZV5) Vel5=Vel5+1.
*
        Do IR=1,105
        If(ZD.le.XX(IR)) then
        DDF(IR)=DDF(IR)+ 1.  ! Direct summing
        goto 499
        endif
        enddo
*
*
499     If(IK.ne.1) goto 510
        Do IR=1,105
        If(Z.le.XX(IR)) then
        DD(IR)=DD(IR)+ DISPLL  ! Lindhard component is summing
        goto 500
        endif
        enddo
                                    endif !  if(PR0.le.PREL)
 500    continue
*End ---elastic
*
        Do IR=1,105
        If(Z.le.XX(IR)) then
        TK(IR)=TK(IR)+ T*STEP
        goto 510
        endif
        enddo
*
510     continue
*
        If(T.le.Tcrit) goto 7000    !   MD
*
        T=T-DT3*STEP
*
        If(T.le.Tcrit) goto 7000    !   MD
*
* usual end of history
         If(T.lt.EDmin) goto 2000
         X=X+STEP
                     DZ=STEP
         If(IEL.ne.0) DZ=STEP*COST
         Z=Z+DZ
         ZD=ZD+DZ
         Goto 1
*
2000     If(IK.eq.1) then
         R1=R1+X
         R1p=R1p+Z
                    endif
         Return
*
```

```
* Interrupt history taking MD results for the rest energy
7000    continue
*no alloy
        LK=1
                       Frcut=0.0
        If(T.gt.0.0)   FRcut=CF2(IK,LK,T) +0.5
        NFRENKEL(IDE)= NFRENKEL(IDE)+Ifix(FRcut)
        goto 2000
        End
*
************************************************************************
*
          Subroutine NRT_COEFF(Z1,Z2,A1,A2,ANRT,BNRT,GNRT)
*         -----------------------------------------------
          PI=3.1415926
          A0=0.52918E-08    ! Bohr radius, cm
          EL=4.8E-10        ! electron charge
C Classical NRT, Nucl.Eng.Des., 1975 :
C         RK0=0.1337*(Z1**(1./6.))*SQRT(Z1/A1)
C Robinson
          RK0=0.0793*(Z1**0.6666666)*SQRT(Z2)*((A1+A2)**1.5)/         (
     #    (A1**1.5)*SQRT(A2)*((Z1**0.666666666+Z2**0.66666666)**0.75) )
C
          AA=A0*(  (9.*Pi*Pi/128.)**0.333333333  )/
     #          SQRT(Z1**0.6666666+Z2**0.6666666)
          E0=Z1*Z2*(EL**2)*(A1+A2)/(AA*A2)
            E0=E0/1.6E-06    !  erg ---> MeV
            E0=1000.*E0      !      ---> keV
            EE=1./E0
C
         ANRT=RK0*EE
         BNRT=0.40244*RK0*(EE**0.75)
         GNRT=3.4008*RK0*(EE**0.1666666666)
         RETURN
         END
*
************************************************************************
*
        Subroutine PRECALCULATION
*       ------------------------
        parameter (maxc=21,maxc1=22,maxe=50001)
        Character INPNAM_XS*12, INPNAM_DEDX*12, OUTNAM*12
        Character CTASK_NAME*80, C50*50
        Common/Basic/Z1,A1,Z2,A2,AW,RO,RN0
        Common/Z2A2/ZT(MAXC),AT(MAXC),FR(MAXC),ED(MAXC),NK
        Common/PKA/ALPHA2(MAXC1,MAXC),E0(MAXC1,MAXC)
        COMMON/NRT1/ANRT(MAXC1),BNRT(MAXC1),GNRT(MAXC1)
        Common/Edmin/Edmin
        Common/PARAM01/IDSDT
        Common/FILENAMES/INPNAM_XS(MAXC1,MAXC),INPNAM_DEDX(MAXC1),
     #  OUTNAM
        Common/MERCURY/CMD(5),Ecrit,MD
        Common/DATATAB1/WTAB(101),ROTAB(101),RITAB(101),IATAB(101)
        Common/ISTOP/ISTOP
        Common/VENUS/RI(MAXC)
        Common/EARTH/IDEFMD,CTASK_NAME,C50
        Common/AUX1/rlambdaft12,rmft12,qft12
        Common/SATURN/TcritLAB(MAXC1)
        Common/EIN/EIN(MAXE),NH(MAXE),KEIN
        Dimension TmpEn(MAXE)
*
                            AVO=6.022E+23     ! Avogadro
        IZ1=ifix(Z1+0.001)
*
* Get masses, density and <I>
        Call DATATAB
*
* Get projectile A, if absent
        If(A1.le.0.0) then
```

```
                        A1=WTAB(IZ1)
                        endif
                        If(A1.le.0.0) Call Error(' A1 ',000)
                        IA1=ifix(A1+0.001)
*
* Get A for material components, if absent
        do i=1,nk
        If(AT(i).le.0.0) then
                        IZT=ifix(ZT(i)+0.001)
                        AT(i)=WTAB(IZT)
                        endif
                        If(AT(i).le.0.0) Call Error(' AT ',i)
        enddo
*
* Density
        if(RO.le.0.0) then
                        IZT=ifix(ZT(NK)+0.001)  ! check for NK=1 in sub read_inp
                        RO=ROTAB(IZT)
                        endif
                        If(RO.le.0.0) Call Error(' RO ',NK)
*
* Mean ionization potential for SPAR calculation, if chosen
        if(istop.eq.1) then
        do i=1,nk
        IZT=ifix(ZT(i)+0.001)
        RI(i)=RITAB(IZT)*1.E-6  ! eV --> MeV (SPAR)
        If(RI(i).le.0.0) Call Error(' RI ',i)
        enddo
                         endif
*
*
* Normalize fractions on unity
           sum=0.0
           do i=1,nk
           sum=sum+FR(i)
           enddo
           Do i=1,nk
           FR(I)=FR(I)/SUM
           Enddo
*
* Get effective Z and A
           Z2=0.0
           A2=0.0
        Do i=1,nk
        Z2=Z2+FR(I)*ZT(I)
        A2=A2+FR(I)*AT(I)
        Enddo
           AW=A2
*
* Atoms (molecules)
        RN0=AVO*RO/AW     ! at/cm3
        DA=(1./RN0)**0.33333333  ! ave dist betw at (not used)
*
* Min ED for the composite material
* ! Edmin is not minimal displacement energy
        Edmin=1.E+10
        do i=1,nk
        If(ED(i).lt.Edmin) Edmin=ED(i)
        enddo
*
* omit input energies < Edmin
        ke=0
        Do i=1,KEIN
        If(EIN(i).ge.Edmin) then
        ke=ke+1
        TmpEn(ke)=EIN(i)
                              endif
        Enddo
        Do i=1,ke
        EIN(i)=TmpEn(i)
        Enddo
        KEIN=KE
*
        NK1=NK+1
*
* NRT coefficients for effective material Z,A
        Do IK=1,NK1
         If(IK.eq.1) then
                     Z111=Z1
                     A111=A1
                     endif
         If(IK.ne.1) then
                     Z111=ZT(IK-1)
                     A111=AT(IK-1)
                     endif
        Call NRT_COEFF(Z111,Z2,A111,A2,ATRN,BTRN,GTRN)
        ANRT(IK)=ATRN
        BNRT(IK)=BTRN
        GNRT(IK)=GTRN
        Enddo
*
* For joint BCA-MD calculations define "critical" energy for each ion
       If(MD.eq.1) Call CRITICAL_ENERGY
*
*
* Data for Lindhard F(x) function
* ------------------------------
          PI=3.1415926
          EL=4.8E-10        ! electron charge
          A0=0.52918E-08    ! cm
*
* IK =1, 2...  1 = projectile, other material component
* MK =1, 2...    = material component
*
*
         DO IK=1,NK1
         If(IK.eq.1) then
                     Z111=Z1
                     A111=A1
                     endif
         If(IK.ne.1) then
                     Z111=ZT(IK-1)
                     A111=AT(IK-1)
                     endif
         DO MK=1,NK
         Z222=ZT(MK)
         A222=AT(MK)
c
* LNS:
          AA=A0*(  (9.*Pi*Pi/128.)**0.333333333  )
     #         /SQRT(Z111**0.6666666+Z222**0.6666666)
* Firsov
          If(IDSDT.ne.0)
     #    AA=A0*(  (9.*Pi*Pi/128.)**0.333333333  )
     #         /(   ( SQRT(Z111)+SQRT(Z222) )**0.666666666   )
*
*
          E0(IK,MK)=Z111*Z222*(EL**2)*(A111+A222)/(AA*A222)
            E0(IK,MK)=E0(IK,MK)/1.6E-06    !  erg ---> MeV
          ALPHA2(IK,MK)=4.*A111*A222/((A111+A222)**2)
          PIA2=Pi*(AA**2)
          ENDDO
          ENDDO
*
*********************************************************************
* open main output file and write basic information
*
        open(33,file=OUTNAM)
        Write(33,*)' I O T A   C O D E'
        Write(33,'(a80)')CTASK_NAME
        Write(33,*)' '
*
        Write(33,1000)IZ1,A1
 1000   Format(1x,'Primary particle (Z,A)',i3,f8.2)
        Write(33,1001)
 1001   Format(' Material      at %    Ed (eV)')
        Do IK=1,NK
        IZ=Ifix(ZT(IK)+0.0001)
        FR0=FR(IK)*100.
        Write(33,1002)IZ,AT(ik),FR0,ED(IK)*1.e+06
 1002   Format(i3,f8.2,f9.2,f9.1)
        Enddo
        If(IDSDT.eq.0) Write(33,1003)rlambdaft12,rmft12,qft12
 1003   Format(' F(t1/2) function: LNS   L=',f6.3,
     #  '  m=',f6.3,'  q=',f6.3)
        If(IDSDT.ne.0) Write(33,1004)
 1004   Format(' F(t1/2) function: Burenkov et al')
*
             If(MD.eq.1)      then
             Write(33,701)
701          Format(/t27,'Joint BCA-MD simulation')
             If(IDEFMD.eq.1) Write(33,7011)C50
7011         Format(' default parameters   ',a50)
*
        if(ifix(ZT(1)+0.001).eq.IZ1.and.ifix(AT(1)+0.001).eq.IA1) then
        Write(33,702)TcritLAB(1)*1000.
702     Format(' MD below Tcrit =',f6.2,' keV')
                                                           else
        if(TcritLAB(1)*1000..lt.1.e+3) then
        Write(33,7021)TcritLAB(1)*1000.,TcritLAB(2)*1000.
                                    else
        Write(33,7022)TcritLAB(1)*1000.,TcritLAB(2)*1000.
                                    endif
7021    Format(' MD below Tcrit =',f6.2,' keV (projectile); =',f6.2,
     #  ' keV (ion of target)')
7022    Format(' MD below Tcrit =',g12.5,' keV (projectile); =',f6.2,
     #  ' keV (ion of target)')
        If(TcritLAB(1).ge.100.) Write(33,7023)TcritLAB(1)
7023    Format(70('*')/'*         Energy Tcrit =',g12.5,' keV',
     #  ' is not realistic',T70,'*'/70('*'))
                                                           endif
*
             Write(33,703)(CMD(i),i=1,3)
703          Format(' efficiency(MD)=',f7.4,'*( eMD**(',f8.4,') ) +',
     #       1pe10.3,'*eMD')
             Write(33,704)CMD(4),CMD(5)
704          Format(' constant effciency below',f7.3,' keV and above',
     #       f7.2,' keV'/)
                          endif
        If(istop.eq.-1) Write(33,*)' Stopping power: Ziegler, SRIM'
        If(istop.eq. 1) Write(33,*)' Stopping power: Armstrong, SPAR'
*
*
          Write(33,1005)RO,RN0
 1005     Format(' Density =',f8.4,' g/cm3    n0=',1pe12.5,' 1/cm3')
          Write(33,1006)Z2,A2
 1006     Format(' Z(eff) =',f6.2,'    A(eff)=',f7.2)
          Write(33,1007)
 1007     Format(' Note. Nuclear nonelastic interactions ',
     #    'are not considered here')
*
          Return
          End
*
************************************************************************
*
       Subroutine PRINT(T0,Nhist)
```

```
*       --------------------------
        parameter (maxc=21,maxc1=22)
        Character INPNAM_XS*12, INPNAM_DEDX*12, OUTNAM*12
        Common/FILENAMES/INPNAM_XS(MAXC1,MAXC),INPNAM_DEDX(MAXC1),
      # OUTNAM
*
        Common/BASIC/Z1,A1,Z2,A2,AW,RO,RN0
        Common/Z2A2/ZT(MAXC),AT(MAXC),FR(MAXC),ED(MAXC),NK
        Common/SKY1/dEdx(MAXC1,10002),dEdxN1(MAXC1,10002),
      # Rp1(MAXC1,10002)
        Common/SUN/Vel1L(405),Tmean1,Vel2Tm(405),Vel3Pl(405),
      #            R1,R1p,Tplay1,Vel4(405),Nel(405)
        Common/STARS1/ dSdX1(MAXC1,MAXC,10002),
      # dEdXNuc1(MAXC1,MAXC,10002), dEdXNuc0Ed1(MAXC1,MAXC,10002),
      # dEdXNucEdTm1(MAXC1,MAXC,10002), CST1(MAXC1,MAXC,10002), NMAX
        Common/Interr/Interr1
        Common/MOON/XX(105),DD(105),DDF(105),TK(105),HST
        COMMON/MAIIPR/IPR1
        COMMON/MEMORY/NFRENKEL(405)
        Common/CUT1/rmaxpl,itot,icut
        Common/OPTIONS1/igraph,maxplay
        Common/EKIN1/EKIN(405),HEK,MEK ! MEK can be less than 405
        Common/SUN2/ZV5,Vel5,NN(405)
        Common/SUNDI/DIVel1(405),DIFRENKEL(405),DIVel5
        Common/MERCURY/CMD(5),Ecrit,MD
        Common/ISTOP/ISTOP
        Dimension EKIN2(405)
*
* Output file is opened in Subr. Read_input
*
        JE=Ifix((alog10(T0)+6.)*1000.)+1
                    If(JE.le.0.or.JE.gt.10002) Call Error(' JE ',11111)
* SRIM or SPAR range for projectile
        RZ=Rp1(1,JE)
        if(istop.eq.-1) RFULL=(1.+0.333333*A2/A1)*RZ     !  for effective A2
        if(istop.eq. 1) RFULL=RZ
*
             RNel=0.0
             Do i=1,405
             RNel=RNel+Float(Nel(i))/Float(Nhist)
             Enddo
*
         Do i=1,405
         Vel1L(i) =Vel1L(i) /Float(Nhist)
         Vel2Tm(i)=Vel2Tm(i)/Float(Nhist)
         Vel3Pl(i)=Vel3Pl(i)/Float(Nhist)
         Enddo
*
         Vel5=Vel5/Float(Nhist)
* Sum
               Vel1LS =0.0
               Vel2TmS=0.0
               Vel3PlS=0.0
               Vel4S  =0.0
         Do i=1,405
         Vel1LS =Vel1LS +Vel1L(i)
         Vel2TmS=Vel2TmS+Vel2Tm(i)
         Vel3PlS=Vel3PlS+Vel3Pl(i)
         Vel4S  =Vel4S  +Vel4(i)
         Enddo
*
* delta = Root[ M(ksi**2) -M**2 ] /M
         DIS1= DIVel1(405)/Float(Nhist) -Vel1LS**2
*
         DIS1= DIS1/Float(Nhist)
         DEL11=SQRT(  DIS1  )
                           ERR1=100.
         If(Vel1LS.ne.0.0) ERR1=100.*DEL11/Vel1LS
*
```

```
        Tmean1=Tmean1/Float(Nhist)
        Tplay1=Tplay1/Float(Nhist)

        R1=R1/Float(Nhist)
        R1p=R1p/Float(Nhist)
*
        If(MD.eq.1) then
        Write(33,701)
701     Format(35(1H*),'BCA+MD',37(1H*))
                    else
        Write(33,1)
1       Format(78(1H*))

                    endif
        Write(33,2)T0, Nhist
2       Format(1x,'Primary energy ',1pg12.5,' MeV'/1x,
     #  ' Number of events ',i7) !1x,60('-'))
*
        CFNRT=0.0
        Do MK=1,NK
        CFNRT=CFNRT+CF(1,MK,T0)*FR(MK)
        Enddo
*
        Write(33,*)'                                   ',
     #  '                           error %      /NRT '
        Write(33,30)CFNRT,CFNRT/CFNRT
30      Format(1x,
     #  'Simple NRT-formula prediction               =',g12.5
     #  ,t69,f6.3,'  (1)')
*
        If(MD.eq.0) Write(33,3)Vel1LS,ERR1, Vel1LS/CFNRT
3       Format(1x,
     #'Number of vacancies (Lindhard, free path )  =',f8.2,
     # '  (',f6.2,')'
     #  ,t69,f6.3,'  (2)')
*
        Write(33,31)Vel4S,Vel4S/CFNRT
31      Format(1x,
     #'Number of vacancies (Lindhard, integration) =',f8.2
     #  ,t69,f6.3,'  (3)')
*
        FRENK1=0.0
        Do i=1,405
        FRENK1=FRENK1+float(NFRENKEL(i))/float(Nhist)
        Enddo
*
* delta
        DISF=  DIFRENKEL(405)/Float(Nhist) - FRENK1**2
        DISF=  DISF/Float(Nhist)
        DELF=SQRT(  DISF )
                          ERRF=100.
        If(FRENK1.ne.0.0) ERRF=100.*DELF/FRENK1
        Write(33,32)
32      Format(76('-'))
        Write(33,9)FRENK1,ERRF,FRENK1/CFNRT
9       Format(1x,
     #  'Number of Frenkel pairs (direct counting)   =',f8.2
     #  ,'  (',f6.2,')'
     #  ,t69,f6.3,'  (4)')
*
* unit=555
        If(MD.ne.0) Write(55,555)
     #  T0, FRENK1/CFNRT, (ERRF/100.)*FRENK1/CFNRT, FRENK1,
     #                    (ERRF/100.)*FRENK1,       CFNRT
555     Format(1pg12.5,  0pf8.3, f8.3, f14.2, f9.2, f14.2)
*
        Write(33,32)
*
* "Modified" counting = direct count *0.8 ( "as in NRT" )
```

```
        If(MD.eq.0) Write(33,10)0.8*FRENK1, ERRF, 0.8*FRENK1/CFNRT
10      Format(1x,
     #  'Modified counting                          =',f8.2
     #  ,'  (',f6.2,')'
     #  ,t69,f6.3/)
*
        CFTRN=0.0
*
* NRT formula for pure material weighted with given fractions
        If(NK.eq.1) goto 1000
*
        CFTRN=0.0
        Do MK=1,NK
             Z222=ZT(MK)
             A222=AT(MK)
             ED2=ED(MK)
        Call NRT_COEFF(Z1,Z222,A1,A222,ATRN,BTRN,GTRN)
        CFTRN=CFTRN+FR(MK)*CF1(ED2,ATRN,BTRN,GTRN,T0)
        Enddo
        Write(33,11)CFTRN
11      Format(1x,
     #  'NRT-formula for pure material components weighted =',f8.2
     #  ,T69,'      ','  (5)')
 1000   continue
*
        If(MD.eq.0) Write(33,4)Vel2TmS,Vel2TmS/CFNRT
4       Format(1x,

     #  'Number of vacancies from Tmean calc              =',f8.2
     #  ,t69,f6.3)
*
        If(MD.eq.0) Write(33,5)Vel3PLS,Vel3PLS/CFNRT
5       Format(1x,
     #  'Number of vacancies from Tplay calc              =',f8.2
     #  ,t69,f6.3)
*

* Get displacement cross-sections
* Lindhard, code SL_LNS :
        CSDL=0.0
        Do MK=1,NK
        CSDL=CSDL+FR(MK)*dSdx1(1,MK,JE)/RN0
        Enddo
        CSDL=CSDL*1.e+24
        CSDDIR=1.e+24*Vel5/(RN0*ZV5)
*
* delta
        DISCS= DIVEL5/Float(Nhist) - Vel5**2
        DISCS= DISCS/Float(Nhist)
        DELCS=SQRT(  DISCS )
                                          ERRCS=100.
        If(CSDDIR.ne.0.0.and.Vel5.ne.0.0) ERRCS=100.*DELCS/Vel5
*
        If(NK.eq.1) Write(33,51)CSDL
  51    Format(/
     #  ' Displacement cross-section (Lindhard, integr)   (b) =',
     #  1pg12.5,
     #  T69,'      ','  (6)')
        If(NK.ne.1) then
        Write(33,52)CSDL
  52    Format(/
     #  ' Displacement cross-section (Lindhard, weighted) (b) =',
     #  1pg12.5,
     #  T69,'      ','  (6)')
        Write(33,53) (dSdx1(1,MK,JE)/(1.e-24*RN0),MK=1,NK)
  53    Format(' Components: ',1pg12.5,4g12.5)
                     endif
        If(MD.eq.0) Write(33,54)CSDDIR,ERRCS
  54    Format(
     #  ' Displacement cross-section (direct counting)    (b) =',
     #  1pg12.5,  ' (',  0pf6.2,   ') (7)' )
*
        If(MD.eq.0) Write(33,55)0.8*CSDDIR,ERRCS
  55    Format(
     #  '                            (modified count )    (b) =',
     #  1pg12.5,  ' (',  0pf6.2,   ')    '/)
*
* unit=44
        If(MD.eq.0) then
        Write(44,111)T0,CFNRT,Vel1LS,Vel4S,FRENK1,CFTRN,CSDL,CSDDIR
 111    Format(1pg12.5,   0pf10.2,  4f10.2,1pg12.5,g12.5)
                    else
        Write(44,111)T0,CFNRT, 0.0  ,Vel4S,FRENK1,CFTRN,CSDL, 0.0
                    endif
*
*
*
        If(MD.eq.0) Write(33,6)Tmean1,Tplay1
6       Format(1x,'Tmean =',1pe12.5,' MeV     Tplay=',g12.5)
*
        If(MD.eq.0) then
        if(istop.eq.-1) Write(33,7)RZ,RFULL,R1
   7    Format(
     #' Path (Ziegler,evl)=',1pe10.3,' cm,  total=',e10.3,' cm'/
     #' Path (total,calc )                    =',e10.3,' cm')
        if(istop.eq.1) Write(33,8)RFULL,R1
   8    Format(' Total path (SPAR)=',1pe10.3,' cm',
     #'   path (total,calc)=',e10.3,' cm')
                    else
        if(istop.eq.-1) Write(33,17)RZ
  17    Format(' Path (Ziegler,evl)=',1pe10.3,' cm')
        if(istop.eq.1) Write(33,18)RFULL
  18    Format(' Total path (SPAR)=',1pe10.3,' cm')
                    endif
*
*
        Write(33,81)Interr1
81      Format(1x,' Histories interrupted (subr model)=', i12)
                      tpc=0.0
        if(itot.ne.0) tpc=100.*float(icut)/float(itot)
*
        If(maxplay.gt.1) then
        Write(33,82)tpc,maxplay
  82    Format(1x,' T play was cut (TPKA rout) =',f16.3,
     #  '  %    Maxplay=',i9)
                         endif
        If(maxplay.lt.0) then
        Write(33,821)tpc
 821    Format(1x,' T play was cut (TPKA rout) =',f16.3,
     #  '  %    no limitation on Maxplay')
                         endif
        If(tpc.gt.5.0.and.maxplay.gt.1) write(33,83)tpc
  83    Format(26('*'),' W A R N I N G ',30('*')/'*',
     #  f15.3,' is too much,  increase Maxplay in input',
     #  ' file         *'/26('*'),' W A R N I N G ',30('*'))
*
* GRAPHs preparation
*
        Sum=0.0
        Sumf=0.0
        Sumt=0.0
        do i=1,105
*
        DD(i)=DD(i)/float(Nhist)
        DDF(i)=DDF(i)/float(Nhist)
c recalculation to the fluens 1/cm2
        DD(i)=DD(i)/(HST*RN0)
        DDF(i)=DDF(i)/(HST*RN0)
```

```
*
        TK(i)=TK(i)/(HST*float(Nhist))
        sum=sum+DD(i)
        sumf=sumf+DDF(i)
        sumt=sumt+TK(i)
        enddo
*
* check the maximal i-number, where DD .ne. 0
        do i=1,105
        j=105-i+1
        if(DD(j).ne.0.0) goto 100
        enddo
        j=105
 100    if(j.le.105-2) j=j+2
        jddmax=j
*
c
c For BCA-MD no damage profile
        If(MD.eq.1.and.igraph.ne.0) then
                                    Write(33,101)
 101    Format(/' (no damage profiles for joint BCA-MD calculations)')
                                    igraph=0
                                    endif
        If(igraph.eq.0) goto 2000
*
        IPR1=33
        DD(1)=0.0
        Write(33,12)
  12    Format(/
     #  ' Profile of damage from Lindhard dS/dx (free path )')
*
        Write(33,13)
  13    Format(
     # ' X=(cm)  Y=(dpa), result is normalized on primary ',
     # 'ion fluence 1/cm**2')
*
        Call GRAPH(XX,DD,jddmax,0)
*
        Write(33,14)Sum,Sum*(HST*RN0)
  14    Format(
     # ' Sum(Y)=',1pe13.5,' dpa','    (unnormalized)= ',g12.5/)
*
        Write(33,15)
  15    Format(/
     #  ' Profile of damage from direct counting            ')
        Write(33,13)
*
        Call GRAPH(XX,DDF,jddmax,0)
        Write(33,14)Sumf,Sumf*(HST*RN0)
*
*
2000    continue
*
* Get cascade function for different primary energies
*
* energy corresponds to the middle of interval
        Do i=2,405
        EKIN2(i)=0.5*( EKIN(i)+ EKIN(i-1) )
        Enddo
        EKIN2(1)=EKIN(1)/2. ! (see subr.Initialize_)
*
        MEK1=MEK+1
        If(MD.eq.0) then
        Write(33,16)
  16    Format(' Number of vacancies at different',
     #' primary energies from current calculation'/
     #'                                   ',
     #'    [explanation (2),(3),(4) see above]'/75('-')/
     #'    Energy     dS/dx(free path)  ',
     #' dS/dx(integr)     Direct counting'/
     #'     (MeV)          (2)       ',
     #  '        (3)               (4)')
              Vel1LS2 =0.0
              Vel4S2  =0.0
              FRENK2  =0.0
        Do i=1,MEK1
              Vel1LS2 =Vel1LS2 +Vel1L(i)
              Vel4S2  =Vel4S2  +Vel4(i)
              FRENK2  =FRENK2  +float(NFRENKEL(i))/float(Nhist)
        Write(33,19) EKIN2(i),Vel1LS2, Vel4S2,FRENK2
  19    Format(1pg12.4,4x,     0pg12.5, 2(6x,g12.5))
        Enddo
                    endif ! If(MD.eq.0)
*
        Write(33,*)' '
        Write(33,*)' '
        Return
        End
*
************************************************************************
*
       Function RANDOM(NO ARGUMENT)
*      -------------------------------
*
* random number generator
*
**** GNU Fortran pseudo random generator:
****
****
      RANDOM=RAND(0)
**
**
** WATCOM Fortram pseudo random number generator
** generator is initialized in Subroutine Initialize_and_Zeroize
** See   Common/urand1/iyg;    iyg=1
**
**      Common/urand1/iyg
**      RANDOM=URAND(iyg)
**
       Return
       End
*
************************************************************************
*
        Subroutine READ_DATA
*       --------------------
        parameter (maxc=21,maxc1=22)
        Character INPNAM_XS*12, INPNAM_DEDX*12, OUTNAM*12
        Common/FILENAMES/INPNAM_XS(MAXC1,MAXC),INPNAM_DEDX(MAXC1),
     #  OUTNAM
        Common/Basic/Z1,A1,Z2,A2,AW,RO,RN0

        Common/Z2A2/ZT(MAXC),AT(MAXC),FR(MAXC),ED(MAXC),NK
*
        Common/dEdxMain01/Ex01(MAXC1,1501), dEdx01(MAXC1,1501),
     #  Rp01(MAXC1,1501), dEdxN01(MAXC1,1501),NdE01(MAXC1)
*
        Common/Lindhard01/Energy01(MAXC1,MAXC,1701),
     #  dSdX01(MAXC1,MAXC,1701),
     #  dEdX_Nuc01(MAXC1,MAXC,1701), dEdX_Nuc_0Ed01(MAXC1,MAXC,1701),
     #  dEdX_Nuc_Ed_Tmax01(MAXC1,MAXC,1701),
     #  CST01(MAXC1,MAXC,1701), NSL01(MAXC1,MAXC)
*
        Common/istop/istop
*
        Dimension Ex01tmp(1501), dEdx01tmp(1501),Rp01tmp(1501),
     #  dEdxN01tmp(1501) ! NdE01tmp
c
```

```
      NK1=NK+1
      if(istop.ne.-1) goto 2
*
* istop=-1
*
* Reading dE/dx from SRIM/Ziegler OUTPUT
* -------------------------------------
      Do IK=1,NK1
*
                                                       IR=21
      open(IR,file=INPNAM_DEDX(IK))
      read(ir,*,end=90003)
      backspace ir
*
      Call dEdX_ZIEGLER_READ(IK, IR, RO, Ex01tmp, dEdx01tmp, Rp01tmp,
     #                       NdE01tmp,   dEdxN01tmp)
* Now units for range and stopping power are Rp(cm), Ex(MeV), dEdx(MeV/cm)
       NdE01(IK)=NdE01tmp
* fill arrays
       do J=1,NdE01tmp
       Ex01(IK,J)   =Ex01tmp(J)
       dEdx01(IK,J) =dEdx01tmp(J)
       Rp01(IK,J)   =Rp01tmp(J)
       dEdxN01(IK,J)=dEdxN01tmp(J)
       enddo
*
       close(IR)
       Enddo ! Do IK=1,NK1
  2    continue
*
* Reading data prepared by SL_LNSxx code
* ------------------------------------
      Do IK=1,NK1
*
      if(IK.eq.1) then
                  IZP=ifix(Z1+0.001)
                  IAP=ifix(A1+0.001)
                  else
                  IZP=ifix(ZT(IK-1)+0.001)
                  IAP=ifix(AT(IK-1)+0.001)
                  endif
*
* IK - id of the moving ion
* MK - id of the material component
*
      Do MK=1,NK
      IZC=ifix(ZT(MK)+0.001)
      IAC=ifix(AT(MK)+0.001)
                                                        IR=21
      open(IR,file=INPNAM_XS(IK,MK))
        Read(IR,*,end=90001)Z10,A10,Z20,A20
        IZ10=ifix(Z10+0.001)
        IA10=ifix(A10+0.001)
        IZ20=ifix(Z20+0.001)
        IA20=ifix(A20+0.001)
        If(IZ10.ne.IZP .or. IA10.ne.IAP .or.
     #     IZ20.ne.IZC .or. IA20.ne.IAC) goto 90002
        do iii=1,MAXC
        read(IR,*)
        enddo
*
        NSL01tmp=1
        Energy01(IK,MK,NSL01tmp)=0.0
        dSdX01(IK,MK,NSL01tmp)  =0.0
        dEdX_Nuc01(IK,MK,NSL01tmp)=0.0
        dEdX_Nuc_0Ed01(IK,MK,NSL01tmp)=0.0
        dEdX_Nuc_Ed_Tmax01(IK,MK,NSL01tmp)=0.0
        CST01(IK,MK,NSL01tmp)=0.0
*
```

```
        Do iii=1,111111111
* 1    2     3               4                5                  6
* T0,  DSDX, DEDX_nuc_total, DEDX_nuc_0_Ed,  DEDX_nuc_Ed_Tmax,  CS_Ed_Tmax
           Read(IR,*,end=4000) XXX1, XXX2, XXX3, XXX4, XXX5, XXX6
           NSL01tmp=NSL01tmp+1
           If(NSL01tmp.gt.1701) Call Error('Dime',1701)
*
           Energy01(IK,MK,NSL01tmp)=XXX1
           dSdX01(IK,MK,NSL01tmp)  =XXX2
           dEdX_Nuc01(IK,MK,NSL01tmp)=XXX3
           dEdX_Nuc_0Ed01(IK,MK,NSL01tmp)=XXX4
           dEdX_Nuc_Ed_Tmax01(IK,MK,NSL01tmp)=XXX5
           CST01(IK,MK,NSL01tmp)=XXX6
*
        Enddo ! Do iii=1,111111111
        Call Error('????',IR)
*
4000    continue
*
        NSL01(IK,MK)=NSL01tmp
*
        close(IR)
*
        Enddo !   Do MK=1,NK
        Enddo !   Do IK=1,NK1
*
check if data are prepared by SL_LNS code up to the same energy
        Do IK=1,NK1
        Do MK=1,NK
        If(NSL01(IK,MK).ne.NSL01(1,1)) goto 90004
        enddo
        enddo
*
        Return
*
90001   Print *,' File with the cross-sections from SL_LNS ',
     #  ' code is ABSENT'
        print *,' for ',IZP,IAP,' +',IZC,IAC,' interactions !'
        print *,'                       press any key...'
        pause
        stop
90002   Print 90012,INPNAM_XS(IK,MK),IZP,IAP,IZC,IAC,Z10,A10,Z20,A20
90012   Format(1x,' Input file with the cross-sections ',a12/1x,
     #  ' for ',i3,i5,' +',i3,i5,' interactions '/1x,
     #  '     contains the information '/1x,
     #  ' for ',2f6.1,' +',2f6.1,' interactions - E R R O R')
        print *,'                       press any key...'
        pause
        stop
90003   Print 90013,INPNAM_DEDX(IK)
90013   Format(//30x,'E  R  R  O  R'
     #  //' There are NO input file from SRIM (Ziegler)',
     #  ' code '//1x,'                        NAME =  ',a12//)
        print *,'                       press any key...'
        pause
        stop
90004   Print 90014
90014   Format(//30x,'E  R  R  O  R'
     #  //' Data prepared by SL_LNS code for different components ',
     #  ' do not coincide !'//1x,'  Delete corresponding files and ',
     #  ' run SL_LNS again.'//)
        print *,'                                     press any key...'
        pause
        stop
        End
*
************************************************************************
*
        Subroutine READ_INPUT
```

```
*       --------------------
* + write general information in output file
c
        parameter (maxc=21,maxc1=22,maxe=50001)
        Character CTASK_NAME*80
        Character INPNAM_XS*12, INPNAM_DEDX*12, OUTNAM*12, OUTNAM2*12
        Character OUTNAM3*12
        Character NF(MAXC)*12,  C50*50, C80*80
        Common/FILENAMES/INPNAM_XS(MAXC1,MAXC),INPNAM_DEDX(MAXC1),
     #  OUTNAM
        Common/Basic/Z1,A1,Z2,A2,AW,RO,RN0
        Common/Z2A2/ZT(MAXC),AT(MAXC),FR(MAXC),ED(MAXC),NK
        Common/EIN/EIN(MAXE),NH(MAXE),KEIN
        Common/OPTIONS1/igraph,maxplay
        Common/CUT1/rmaxpl,itot,icut
        Common/PARAM01/IDSDT
        Common/MERCURY/CMD(5),Ecrit,MD
        Common/ISTOP/ISTOP
        Common/EARTH/IDEFMD,CTASK_NAME,C50
        Common/SATURN/TcritLAB(MAXC1)
        Print *,' Wait a bit...'
c
        open(1,file='input')
c
c
c find line with problem name
        call star(1)
        read(1,'(a80)')CTASK_NAME
c
c find line with graph printing option and maxplay
        call star(1)
c
c dS/dT
c IDSDT=0, 10(=0), 11-15 LNS expression;
c      =1                Burenkov et al
c Tlim maximal energy for SL_LNS code
        read(1,*)igraph,maxplay,Tlim,IDSDT
        if(maxplay.eq.0) maxplay=250000  ! default option
           If(Tlim.gt.5200.)then
           Print *,'           E R R O R  in INPUT file       '
           Print *,'Maximal energy should be below 5.2 GeV  !'
           Print *,'                       press any key...'
           pause
           stop
           endif
c
        If(Tlim.le.0.0) Tlim=5000.
c

c Define f(t^1/2) function parametrs
        Call FT12CHOICE(IDSDT)
c
c
        If(IDSDT.ne.0.and.IDSDT.ne.1) then
        Print *,'           E R R O R  in INPUT file       '
        Print *,'  IDSDT parameter is incorrect'
        Print *,'                          press any key...'
        pause
        stop
        endif
c
        rmaxpl=float(maxplay)
        if(maxplay.lt.0) rmaxpl=1.e+9
c
c try to find line with BCA-MD instruction
        call star(1)
        read(1,'(a80)')C80
c "BCA-MD" is identified by "B" or "M" letter
        if(findp(C80,'B').ne.0. .or. findp(C80,'M').ne.0. .or.
     #     findp(C80,'b').ne.0. .or. findp(C80,'m').ne.0. ) then
c
        MD=1

c find information needed for BCA-MD joint calculations
        call star(1)
c
c general formula for efficiency :  eff=C(1)*eMD**C(2) + C(3)*eMD
c if T(keV) < CMD(4) eff=const   (see function CF2)
c          and
c if T(keV) > CMD(5) eff=const
        read(1,*,err=9919) (CMD(i),i=1,5),Ecrit
                                                         IDEFMD=0
        if(CMD(1).eq.0.0.and.CMD(2).eq.0.0.and.CMD(3).eq.0.0.and.
     #  CMD(4).eq.0.0.and.CMD(5).eq.0.0.and.Ecrit.eq.0.0) IDEFMD=1
c
                          if(IDEFMD.eq.0) then   ! no default MD parameters
        if(CMD(4).lt.1. .or. Ecrit.lt.1.0)      Call Error(' keV',1)
        if(CMD(4).gt.1000. .or. Ecrit.gt.1000.) Call Error(' keV',1000)
        if(CMD(1).eq.0.0.and.CMD(2).eq.0.0.and.CMD(3).eq.0.0.and.
     #  CMD(4).ne.0.0.or.CMD(5).ne.0.0.or.Ecrit.ne.0.0)
     #                                          Call Error('MDde',0)
c
                                          endif
                                                          else
        MD=0
            Do itcritlab=1,maxc1
            TcritLAB(itcritlab)=-1.E+6
            Enddo
        backspace 1
                                                          endif
c
c find line with Z1, A1 of projectile
        call star(1)
        read(1,*,err=9929)Z1,A1
c
c find line with RO (density of the material), number of material components
c and option to calculate dE/dx
        call star(1)
        read(1,*)RO, NK
                 backspace 1
                 istop=0
                 read(1,'(a80)')C80
c
c Ziegler = SRIM (identified by "Z", "I")
        if(findp(C80,'Z').ne.0.0.or.findp(C80,'z').ne.0.0
     #  .or.findp(C80,'I').ne.0.0.or.findp(C80,'i').ne.0.0) istop=-1
c
c Armstrong=SPAR (identified by "A")
        if(findp(C80,'A').ne.0.0.or.findp(C80,'a').ne.0.0)  istop=+1
c
        If(istop.eq.0) then
        Print *,' '
        Print *,' Model to obtain dE/dx is not identified !!'
        print *,'                               press any key...'
        pause
        stop
        endif
c
        If(NK.gt.MAXC) then
        Print *,'Number of components > ',MAXC,' !'
        print *,'                      press any key...'
        pause
        stop
        endif
c
        If(RO.le.0. .and. NK.ne.1) then
        Print *,' '
        Print *,
     #  '      NO  default density for compounds !!!'
        print *,'                           press any key...'
        pause
        stop
        endif
c
        If(MD.ne.0 .and. NK.ne.1) then
        Print *,
     #  'Joint BCA-MD calculations are not tested for compounds yet !'
        Print *,
     #  '  (!!!)  See comments *no alloy in Subr MODEL and READ_INP'
        print *,'                        press any key...'
        pause
        stop
        endif
c
c
c find file names with the specific cross-sections from SL_LNSxx code
c (run before) for each component of the media and dE/dx from SRIM
c for projectile + media
        call star(1)
        Call GET_FILE_NAMES(1,NK,NF)
        Do MK=1,NK
        INPNAM_XS(1,MK)=NF(MK)
        Enddo
*
        if(istop.eq.-1) then
        call star(1)
        Call GET_FILE_NAMES(1,1,NF)
        INPNAM_DEDX(1)=NF(1)
                        endif
c
c find lines with the description of material componets
        Do IK=1,NK
        call star(1)
        read(1,*,err=90000)ZZ,AA, FR0, ED0
                             If(ED0.le.0.0) ED0=40.0  ! default = 40 eV
        IZ=ifix(ZZ+0.001)
*
* default parameters for MD efficiency
*no alloy
        If(MD.eq.1 .and. IDEFMD.eq.1) Call MDPARAM(IZ,C50)
*
        ZT(IK)=ZZ
        AT(IK)=AA
        FR(IK)=FR0
        ED(IK)=ED0 * 1.e-06    ! eV --> MeV
c
        call star(1)
        Call GET_FILE_NAMES(1,NK,NF)
        Do MK=1,NK
        INPNAM_XS(IK+1,MK)=NF(MK)
        Enddo
*
        if(istop.eq.-1) then
        call star(1)
        Call GET_FILE_NAMES(1,1,NF)
        INPNAM_DEDX(IK+1)=NF(1)
                        endif
*
        Enddo ! Do IK=1,NK
*
* find line with output file name
        call star(1)
        if(MD.eq.0) then
        Call GET_FILE_NAMES(1,2,NF)
        OUTNAM=NF(1)
        OUTNAM2=NF(2)
                    else
```

```
* for BCA-MD get 3 file names
        Call GET_FILE_NAMES(1,3,NF)
        OUTNAM=NF(1)
        OUTNAM2=NF(2)
        OUTNAM3=NF(3)
                   endif
*
* find lines with energies
        Do ir=1,11111111
        call star(1)
        read(1,*,err=2,end=2)EE,NHH
            If(ir.gt.MAXE) then
            print *,' T O O   M U C H   I N P U T   E N E R G I E S  !'
            print *,' Change parameter MAXE in the code !'
            print *,'                                  press any key...'
            pause
            stop
            endif
        EIN(ir)=ABS(EE)
             If(TLIM.le.EIN(ir)) then
             Print *,'          E R R O R  in INPUT file      '
             Print *,'Input energy  ',EIN(ir),'  > TLIM ! '
             Print *,'                      press any key...'
             pause
             stop
             endif
*
        NH(ir)=NHH
        KEIN=ir
        if(EE.lt.0.0) goto 2
        Enddo
 2      Close(1)
*
****************************************************************************
* open part of output files (units 44,55)
* (main output file is open in subr precalculation)
*
        open(44,file=OUTNAM2)
        If(MD.eq.0) Write(44,2000)OUTNAM
 2000   Format(' See explanation of (1), (2),...',
     #  ' in main output file:  "',a12,' "'/)
        If(MD.ne.0) Write(44,2001)OUTNAM
 2001   Format(' See explanation of (1), (3),...',
     #  ' in main output file:  "',a12,' "'/)
        Write(44,2002)
 2002   Format(
     #  ' E(MeV)         (1)       (2)',

     #  '       (3)       (4)       (5)       (6)       (7)'/86('-'))
*
        If(MD.eq.0) Return
        open(55,file=OUTNAM3)
        Write(55,2003)
 2003   Format(
     #  'Eff[iciency]= N(direct counting)/N(NRT)'//
     #  '    T0         Eff +/- dEff  ',
     #  '    N(direct) +/- dN(FR)     N(NRT)'/65('-'))
*
        Return
 9919   Call Error(' MD ',9919)
 9929   Call Error('Z,A ',9929)
90000   print 90001
90001   Format(/'  E R R O R  in input file INPUT !!!'
     #  /'  Possible reason:  dE/dx model chosen is inconsistent ',
     #  /'                    with the file names introduced in INPUT'
     #  /'  Note:   SPAR option - names should be omitted'
     #  /'          SRIM option - names should be given')
        print *,'                                press any key...'
        pause
        stop
        End
*
************************************************************************
*
       Subroutine SPAR0(ZPR, APR, E, dEdxTOTS, dEdxNS ,JJ)
*      ---------------------------------------------------
       parameter (maxc=21,maxc1=22)
       real *8 spar, e_proj, proj_m, z_proj, avz, at_den, bari_ln,
     # fme_(maxc), a_(maxc), del_pr, sparN
*
       Common/Basic/Z1,A1,Z2,A2,AW,RO,RN0
       Common/Z2A2/ZT(MAXC),AT(MAXC),FR(MAXC),ED(MAXC),NK
       Common/VENUS/RI(MAXC)
       Dimension iza_(maxc)
*
*
       e_proj= E
       proj_m= APR*931.5d0  !    mass in MeV
       z_proj= ZPR
*
       avz=0.d0
       bari_ln=0.d0
           Do i=1,NK
           avz=avz+FR(i)*ZT(i)
           bari_ln= bari_ln+FR(i)*ZT(i)*alog( RI(i) )
           fme_(i)= FR(i)
           a_(i)  = AT(i)
                IZT=ifix(ZT(i)+0.001)
                IAT=ifix(AT(i)+0.001)
                iza_(i)=1000*IZT+IAT
           enddo
       bari_ln=bari_ln/avz
*
       at_den=RN0/1.d+24  ! 1/b-cm
       nel1=NK
c
c
       stopping=spar(e_proj,proj_m,z_proj,avz,at_den,bari_ln,fme_,a_,
     # iza_,nel1,jj,del_pr,sparN)
c
       dEdxTOTS=real(stopping)
       dEdxNS=real(sparN)
c
       Return
       end
*
*
************************************************************************
*
      function spar(e_proj,proj_m,z_proj,avz,at_den,bari_ln,fme_,a_,
     # iza_,nel1,jj,del_pr,sparN)
*
!       REAL dedx CODING
!       dedx for heavy ion, protons,pions,muons
!       units: MeV/cm
!
!         e_proj = projectile kinetic energy (MeV)
!         proj_m = projectile mass (MeV)
!         z_proj = projectile charge number
!         avz = <Z> for the medium
!             = SUM fme_(i)*z(i)
!         at_den = atomic density (1/barn-cm)
!         bari_ln = <ln I> with I in MeV
!                 = SUM fme_(i)*z(i)*ln I(i) / <Z>
!         fme_(i)  = array of atom fracions i=1,...,nel1
!         a_(i) = array of mass numbers i=1,...,nel1
!                  (more properly, atomic weights)
!         iza_(i) = array of 1000*Z + A numbers i=1,...,nel1
!         nel1 = number of isotopes in medium
!         jj:
!            jj=1 gaseous element
!            jj=2 condensed element
!            jj=3 gaseous mixture
!            jj=4 condensed mixture
* added
!         sparN = nuclear stopping power
!
!
      implicit double precision (a-h,o-z)
      parameter (huge_float = 1.0d+37, tiny_float = 1.0d-37)
      parameter (dp0=0.d0, dp1=1.d0, dp2=2.d0, dp3=3.d0, dp4=4.d0,
     & dp5=5.d0, dp6=6.d0, dp10=10.d0, dp12=12.d0, dp24=24.d0,
     & dp30=30.d0, dp41=41.d0, dp120=120.d0, dp130=130.d0,
     & dp200=200.d0, dp650=650.d0, dp720=720.d0,
     & dp1p30=1.d30)
      parameter (dpth=dp1/dp3, dph=0.5d0, dp2th=dp2/dp3,
     & dppi=3.1415926535898d0)
      parameter (dp1m2=1.d-02, dp1m4=1.d-04, dp1m5=1.d-05,
     & dp1m6=1.d-06, dp1m10=1.d-10, dp1m20=1.d-20)
      parameter (dp4m3=4.d-03, dp2m2=2.d-02, dp9m1=0.9d0, dp11m1=1.1d0,
     & dp7095m6=7.095d-03,
     & dp6906m2=69.06d0)
      save
      parameter (prot_mass=938.2723128d0)
      parameter (prot_m=prot_mass)
      dimension fme_(nel1), a_(nel1)
      dimension iza_(nel1)
      data elow /1.31d0/, ehigh /5.24d0/
      e_prot=e_proj*prot_m/proj_m
      del_pr=dp0
**
      stpN1=0.d0
      stpN2=0.d0
**
      if (e_prot.gt.elow) stp1=z_proj**2*dedx1(e_prot,proj_m,avz,at_den
     & ,bari_ln,fme_(1),iza_(1),nel1,jj,del_pr)
      if (e_prot.lt.ehigh) stp2=dedx0(e_proj,proj_m,z_proj,avz,at_den
     & ,fme_(1),a_(1),iza_(1),nel1, stpN2)
      fx=(e_prot-elow)/(ehigh-elow)
      fx=min(dp1,max(dp0,fx))
c
      spar=fx*stp1+(dp1-fx)*stp2
**
      sparN=fx*stpN1+(dp1-fx)*stpN2
      return
      end
*
************************************************************************
*
      function dedx0(e_proj,proj_m,z_proj,avz,at_den,fme_,a_,iza_,nel1,
     &               stpN2)
!
!       dedx for heavy ion, protons,pions,muons
!
!       units: MeV/cm
!
!       low energy models from SPAR.
!       equivalent proton energy < 8 MeV.
!
!         e_proj = projectile kinetic energy (MeV)
!         proj_m = projectile mass (MeV)
!         z_proj = projectile charge number
!         avz = <Z> for the medium
!             = SUM fme_(i)*z(i)
!         at_den = atomic density (1/barn-cm)
!         fme_  = array of atom fracions i=1,...,nel1
!         a_(i) = array of mass numbers i=1,...,nel1
!                  (more properly, atomic weights)
!         iza_(i) = array of 1000*Z + A numbers i=1,...,nel1
!         nel1 = number of isotopes in medium
!
      implicit double precision (a-h,o-z)
      parameter (huge_float = 1.0d+37, tiny_float = 1.0d-37)
      parameter (dp0=0.d0, dp1=1.d0, dp2=2.d0, dp3=3.d0, dp4=4.d0,
     & dp5=5.d0, dp6=6.d0, dp10=10.d0, dp12=12.d0, dp24=24.d0,
     & dp30=30.d0, dp41=41.d0, dp120=120.d0, dp130=130.d0,
     & dp200=200.d0, dp650=650.d0, dp720=720.d0,
     & dp1p30=1.d30)
      parameter (dpth=dp1/dp3, dph=0.5d0, dp2th=dp2/dp3,
     & dppi=3.1415926535898d0)
      parameter (dp1m2=1.d-02, dp1m4=1.d-04, dp1m5=1.d-05,
     & dp1m6=1.d-06, dp1m10=1.d-10, dp1m20=1.d-20)
      parameter (dp4m3=4.d-03, dp2m2=2.d-02, dp9m1=0.9d0, dp11m1=1.1d0,
     & dp7095m6=7.095d-03,
     & dp6906m2=69.06d0)
      save
      parameter (emass=0.5109990615d0)
      parameter (av_num=0.6022136736d0)
      parameter (el_rad=0.28179409238d0)
      parameter (stp_con=dp4*dppi*el_rad**2*emass)
      parameter (prot_mass=938.2723128d0)
      parameter (prot_m=prot_mass)
      parameter (om30=1.0d-30)
      dimension a_(nel1), fme_(nel1)
      dimension iza_(nel1)
      se=dp0
      snuc=dp0
      if (e_proj.eq.dp0) go to 20
      rel_m=proj_m/prot_m
      if (e_proj.gt.8.d0*rel_m) stop ' energy too large in DEDX0 '
      fac1=z_proj**dpth
      bcut1=.07d0*fac1**2
      bcut2=.0046d0*fac1
      gammasq=(e_proj/proj_m+dp1)**2
      betasq=(gammasq-dp1)/gammasq
      beta=sqrt(betasq)
      beta0=beta
      fac0=dp1
      if (beta0.lt.bcut2) then
!-----
! BETA0 < BCUT2
        beta=bcut2
        betasq=beta**2
        gammasq=dp1/(dp1-betasq)
        ei=(sqrt(gammasq)-dp1)*proj_m
        fac0=sqrt(e_proj/ei)
      endif
      a7=stp_con*avz*at_den*f_linhard(betasq,avz)/betasq
      zsq=z_proj*z_proj
      se=zsq*fac0*a7
      if (beta0.ge.bcut1) go to 20
!-----
! Compute electronic stopping power
      temp=(dp1-exp(-125.d0*beta/fac1**2))**2
      se=se*temp
!-----
! Compute nuclear stopping power
      do 10 j=1,nel1
      z = iza_(j)/1000
      q1=sqrt(fac1**2+z**dp2th)
      q2=a_(j)+rel_m
      q3=z_proj*z
      fj=3.255d+04*a_(j)/(q2*q3*q1)
      gj=1.96d-4*a_(j)*q2*q1/(q3*rel_m)
      ej=fj*e_proj
      if (ej.gt.1.d+03) go to 10
      if (ej.gt.dp4) then
        dedp=0.5455d0*log(ej)/(ej*(dp1-0.9988d0*ej**(-1.5391d0)))
      else
        dedp=4.46426d0*sqrt(ej)*exp(-2.542d0*ej**0.277d0)
      endif
      sj=(a_(j)*at_den*fme_(j)/av_num)*dedp/gj
      snuc=snuc+sj
   10 continue
!-----
   20 continue
      dedx0=max(se+snuc,om30)
**
      stpN2=snuc
      return
      end
*
*-----------------------------------------------------------------------
      the unchanged routines of SPAR are not shown
*-----------------------------------------------------------------------
*
*
************************************************************************
*

       Subroutine star(k)
*      -----------------
       character c1*1
1      read(1,'(a1)',end=1000)c1
       if(c1.eq.'*'.or.c1.eq.'c'.or.c1.eq.'C'.or.c1.eq.'!') goto 1
       backspace 1
       return
1000   if(k.eq.0) return
       Print *,' U N E X P E C T E D   E N D   O F   I N P U T '
       print *,'                         press any key...'
       pause
       stop
       end
*
************************************************************************
*
      Subroutine STORE_IN(MK,T,Z,IDE)
*     ------------------------------
       COMMON/RECSTORE/TKIN(50003),ZZ(50003),NKO(50003),INDIC(50003),
     #                 NREC
*
      NREC=NREC+1
               If(NREC.gt.50003) Then
               Print *,'Dimension of TKIN,ZZ.. is not sufficient !'
               Print *,'Check COMMON/RECSTORE/  in the code      !'
               call error('TKIN',50003)
               endif
      TKIN(NREC)=T
      ZZ(NREC)=Z
      NKO(NREC) =MK
      INDIC(NREC)=IDE
      RETURN
      END
*
************************************************************************
*
      Subroutine STORE_OUT(MK,T,Z,IDE)
*     -------------------------------
       COMMON/RECSTORE/TKIN(50003),ZZ(50003),NKO(50003),INDIC(50003),
     #                 NREC
*
      NREC=NREC-1
      If(NREC.le.0) Return
      T =TKIN(NREC+1)
      Z =ZZ(NREC+1)
      MK=NKO(NREC+1)
      IDE=INDIC(NREC+1)
      RETURN
      END
*
*************************************************************************
*
       Function TDAM(IK,TkeV)
*      -----------------------
       parameter (maxc1=22)
       COMMON/NRT1/ANRT(MAXC1),BNRT(MAXC1),GNRT(MAXC1)
       TT=TkeV
*
       Tdam= TT/
     # (1.+ANRT(IK)*TT+BNRT(IK)*(TT**0.75)+GNRT(IK)*(TT**0.16666666666))
       RETURN
       END
*
*************************************************************************
*
        Function TPKA(IK,MK,T0,Tmean)
*       ----------------------------
        parameter (maxc=21,maxc1=22)
        Common/Z2A2/ZT(MAXC),AT(MAXC),FR(MAXC),ED(MAXC),NK
        Common/PKA/ALPHA2(MAXC1,MAXC),E0(MAXC1,MAXC)
        Common/CUT1/rmaxpl,itot,icut
        itot=itot+1
*
        Emin=ED(MK)
        Tmax=ALPHA2(IK,MK)*T0
                             If(Tmax.le.Emin) then
                             TPKA=Emin+0.0001
                             return
                             endif
        EPS= T0/E0(IK,MK)
        COEFF=ALPHA2(IK,MK)*E0(IK,MK)/EPS
*
        Xmin= SQRT( Emin/COEFF )
c x(T)  X= SQRT( T0/COEFF )
*
* dx= (1/2) * (1/coeff)**0.5 / T**0.5
        RM=F(Xmin)/(Emin**1.5)      ! 3/2:  1 from x**2 and 0.5 from dx
*
        rn=0.
*
1000    TT=Emin+RANDOM(0)*(Tmax-Emin)
        XX= SQRT( TT/COEFF )
        rn=rn+1.
             if(rn.gt.rmaxpl) then
             icut=icut+1
             TPKA=Tmean !
123      Format(1x,60('-')/1x,' IK,MK,T0,Tmean,Tmax=',2i4,3g12.5)
             return
             endif
1900    continue
        If( RM*RANDOM(0)-F(XX)/(TT**1.5) )2000,2000,1000
*
2000    TPKA=TT
        If(TT.gt.T0)then
        Print *,' T0=',T0,'  Tplay=',TT,' ALPHA2=',ALPHA2
        endif
*
         Return
         End


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

```
      Subroutine ZIEION(C80,Z,A)
      Character C80*80
c
c Ex:   C80=' Ion = Tungsten [74] , Mass = 183.951 amu'
      open(29,status='scratch')
      Do i=1,80
      i1=i
      If(C80(i:i).eq.'[') goto 1000
      enddo
      Print *,' Subr. ZIEION   Symbol "[" not found !'
      print *,'                           press any key...'
      pause
      stop
 1000 Do i=i1,80
      i2=i
      If(C80(i:i).eq.']') goto 1001
      enddo
      Print *,' Subr. ZIEION   Symbol "]" not found !'
      print *,'                           press any key...'
      pause
      stop
 1001 if((i2-i1).eq.2) then
      k=i1+1
      write(29,'(a1)')C80(k:k)
                    endif
      if((i2-i1).eq.3) then
      k1=i1+1
      k2=i1+2
      write(29,'(a2)')C80(k1:k2)
                    endif
      if((i2-i1).ne.2.and.(i2-i1).ne.3) then
      Print *,' Subr. ZIEION    Error:   i2-i1=',i2-i1
      print *,'                        press any key...'
      pause
      stop
                    endif
      Do i=i2,80
      j1=i
      If(C80(i:i).eq.'=') goto 2000
      enddo
      Print *,' Subr. ZIEION    Second symbol "=" not found !'
      print *,'                                 press any key...'
      pause
      stop
 2000 Do i=j1,80
      j2=i
      If(C80(i:i).eq.'a') goto 2001
      enddo
      Print *,' Subr. ZIEION   "amu" not found !'
      print *,'                       press any key...'
      pause
      stop
 2001 k1=j1+1
      k2=j2-1
      write(29,'(a12)')C80(k1:k2)
      rewind 29
      read(29,*)Z
      read(29,*)A
      close(29)
      Return
      end
```