

### **Source routines**

An introduction to a new approach to source routines

Beginner online training, Spring 2021

## Why user routines?

- FLUKA offers plenty of built-in tools to define primary beams and estimate quantities
- Sometime these are not enough
- There is the need to write some dedicated code: a "User Routine"
- URs are beyond the scope of this course because of intrinsic difficulties
- Nevertheless, we have started an effort to make URs more user-friendly
- We want to introduce here the first effort in this direction: a modernized version of the source routine
- Why the source routine first? Built-in options allow to sample from a limited number of distribution and not from histograms. This is an effort to overcome this limitation



### The "old" source routine

- Scary for beginners, limited documentation
- Use of **IMPLICIT** and **FORTRAN77** naming convention (see later)

1	*	64	*	push many but this way we reserve a
2	*== source*			stack for the secondaries to be gene
3	*	66		Npflka is the stack counter: of cour
4	SUBROUTINE SOURCE ( NOMORE )	67	*	must be =0
5		68		NPFLKA = NPFLKA + 1
6	INCLUDE 'dblprc.inc'	69	*	Wt is the weight of the particle
7	INCLUDE 'dimpar.inc'	70		WTFLK (NPFLKA) = ONEONE
8	INCLUDE 'iounit.inc'	71		WEIPRI = WEIPRI + WTFLK (NPFLKA)
9	*	72	*	Particle type (1=proton). Ijbea
10	**	73	*	card
11	* *	74		+
12	* Copyright (C) 2003-2019: CERN & INFN *	75	*	(Radioactive) isotope:
13	* All Rights Reserved. *	76		IF ( IJBEAM .EQ2 .AND. LRDBEA
14	* *	77		IARES = IPROA
15	<ul> <li>New source for FLUKA9x-FLUKA20xy:</li> </ul>	78		IZRES = IPROZ
16	* *	79		IISRES = IPROM
17	* Created on 07 January 1990 by Alfredo Ferrari & Paola Sala *	80		CALL STISBM ( IARES, IZRES, II
18	* Infn - Milan *	81		IJHION = IPROM * 100000 + MOD
19	* *	82		IJHION = IJHION * 100 + KXH
20	* This is just an example of a possible user written source routine. *	83		IONID = IJHION
21	* note that the beam card still has some meaning - in the scoring the *	84		CALL DCDION ( IONID )
22	<pre>* maximum momentum used in deciding the binning is taken from the *</pre>	85		CALL SETION ( IONID )
23	* beam momentum. Other beam card parameters are obsolete. *	86		LFRPHN (NPFLKA) = .FALSE.
24	*	87	*	The second se
25	* Output variables: *	88	*	+
26	* output variables. *	89	*	Heavy ion:
27	* Nomore = if > 0 the run will be terminated *	90		ELSE IF ( IJBEAM .EQ2 ) THEN
28	* Nomore = 11 > 0 the run with be terminated *	91		IJHION = IPROM * 100000 + MOD
29	ŶŶ	92		IJHION = IJHION * 100 + KXH
30	*	93		IONID = IJHION
31	INCLUDE 'beamcm.inc'	94		CALL DCDION ( IONID )
32	INCLUDE 'fheavy.inc'	95		CALL SETION ( IONID )
33	INCLUDE 'flkstk.inc'	96		ILOFLK (NPFLKA) = IJHION
34	INCLUDE 'ioiocm.inc'	97	+	Flag this is prompt radiation
35		98		LRADDC (NPFLKA) = .FALSE.
35	INCLUDE 'ltclcm.inc'	99	+	Group number for "low" energy neu
37	INCLUDE 'paprop.inc'	100		IGROUP (NPFLKA) = 0
38	INCLUDE 'sourcm.inc'	101	+	Parent radioactive isotope:
39	INCLUDE 'sumcou.inc'	102		IRDAZM (NPFLKA) = 0
		103	*	Particle age (s)
40	LOGICAL LFIRST, LISNUT	104	^	AGESTK (NPFLKA) = +ZERZER
41	*	105		Kinetic energy of the particle (G
42	SAVE LFIRST	105	*	TKEFLK (NPFLKA) = SQRT ( PBEAM
43	DATA LFIRST / .TRUE. /	100		8 - AM (IONID)
44	* Statement function:			Particle momentum
45	LISNUT (IJ) = INDEX ( PRNAME (IJ), 'NEUTRI' ) .GT. 0	108	*	PMOFLK (NPFLKA) = PBEAM
46	**			
47	* *	110		PMOFLK (NPFLKA) = SQRT ( TKEFL
48	* BASIC VERSION *	111	*	8 + TWOTW
49	* *	112		LFRPHN (NPFLKA) = .FALSE.
50	**	113		
51	NOMORE = 0	114	*	
52	* +*	115	*	
53	<ul> <li>* First call initializations:</li> </ul>	116		ELSE
54	IF ( LFIRST ) THEN	117		IONID = IJBEAM
55	* *** The following 3 cards are mandatory ***	118		ILOFLK (NPFLKA) = IJBEAM
56	TKESUM = ZERZER	119	*	
57	LFIRST = .FALSE.	120		LRADDC (NPFLKA) = $.FALSE$ .
58	LUSSRC = .TRUE.	121	*	
59	<pre>* *** User initialization ***</pre>	122		IGROUP (NPFLKA) = 0
60	END IF	123	*	
61	*	124		IRDAZM (NPFLKA) = 0
62	**	125	*	

63 \* Push one source particle to the stack. Note that you could as well 64 \* push many but this way we reserve a maximum amount of space in the enerated ourse any time source is called it eam is the type set by the BEAM ..... ) THEN ISRES ) OD ( IPROZ, 100 ) \* 1000 + IPROA HEAV DD ( IPROZ, 100 ) \* 1000 + IPROA KHEAV eutrons, set to 0 anyway GeV) AM\*\*2 + AM (IONID)\*\*2 ) FLK (NPFLKA) \* ( TKEFLK (NPFLKA) WO \* AM (IONID) ) ) eutrons, set to 0 anyway

126		AGESTK (NPFLKA) = +ZERZER	
127	*	Kinetic energy of the particle (GeV)	
128		TKEFLK (NPFLKA) = SQRT ( PBEAM**2 + AM (IONID)**2 )	
129		δ – AM (IONID)	
130	*	Particle momentum	
131		PMOFLK (NPFLKA) = PBEAM	
132	*	PMOFLK (NPFLKA) = SQRT ( TKEFLK (NPFLKA) * ( TKEFLK (NPFLKA)	
133	*	<pre>&amp; + TWOTWO * AM (IONID) ))</pre>	
134	*	+*	
135	*	Check if it is a neutrino, if so force the interaction	
136	*	(unless the relevant flag has been disabled)	
137		IF ( LISNUT (IJBEAM) .AND. LNUFIN ) THEN	
138		LFRPHN (NPFLKA) = .TRUE.	
139	*		
140	*	**	
141	*	Not a neutrino	
142		ELSE	
143		LFRPHN (NPFLKA) = .FALSE.	
144		END IF	
145	*		
146	*	**	
147		END IF	
148	*		
149	*	*	
150	*	From this point	
151	*	Particle generation (1 for primaries)	
152		LOFLK (NPFLKA) = 1	
153	*	User dependent flag:	
154		LOUSE (NPFLKA) = 0	
155	*	No channeling:	
156		KCHFLK (NPFLKA) = 0	
157		ECRFLK (NPFLKA) = ZERZER	
158	*	Extra infos:	
159		INFSTK (NPFLKA) = 0	
160		LNFSTK (NPFLKA) = 0	
161		ANFSTK (NPFLKA) = ZERZER	
162	*	Parent variables:	
163		IPRSTK (NPFLKA) = 0	
164		EKPSTK (NPFLKA) = ZERZER	
165	*	User dependent spare variables:	
166		DO 100 ISPR = 1, MKBMX1	
167		SPAREK (ISPR, NPFLKA) = ZERZER	
168	1	DØ CONTINUE	
169	*	User dependent spare flags:	
170		DO 200 ISPR = 1, MKBMX2	
171		ISPARK (ISPR,NPFLKA) = 0	
172	2	00 CONTINUE	
173	*	Save the track number of the stack particle:	
174		ISPARK (MKBMX2,NPFLKA) = NPFLKA	
175		NPARMA = NPARMA + 1	
176		NUMPAR (NPFLKA) = NPARMA	
177		NEVENT (NPFLKA) = 0	
178		DFNEAR (NPFLKA) = +ZERZER	
179	*	to this point: don't change anything	
180		AKNSHR (NPFLKA) = -TWOTWO	
181	*	Cosines (tx,ty,tz)	
182		TXFLK (NPFLKA) = UBEAM	
183		TYFLK (NPFLKA) = VBEAM	
184		TZFLK (NPFLKA) = WBEAM	
185	*	TZFLK (NPFLKA) = SQRT ( ONEONE - TXFLK (NPFLKA)**2	
186	*	δ - TYFLK (NPFLKA)**2 )	
187	*		
188		TXPOL (NPFLKA) = -TWOTWO	

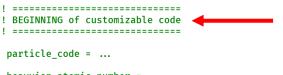
	TYPOL (NPFLKA) = +ZERZER TZPOL (NPFLKA) = +ZERZER
	Particle coordinates
~	XFLK (NPFLKA) = XBEAM
	YFLK (NPFLKA) = YBEAM
	ZFLK (NPFLKA) = ZBEAM
*	Calculate the total kinetic energy of the primaries: don't change
*	(Radioactive) isotope:
*	IF ( IJBEAM .EQ2 .AND. LRDBEA ) THEN
*	
	ELSE IF ( ILOFLK (NPFLKA) .EQ2 .OR.
	5 ILOFLK (NPFLKA) .GT. 100000 ) THEN
	TKESUM = TKESUM + TKEFLK (NPFLKA) * WTFLK (NPFLKA)
*	
*	
.2	ELSE IF ( ILOFLK (NPFLKA) .NE. 0 ) THEN
	TKESUM = TKESUM + ( TKEFLK (NPFLKA) + AMDISC (ILOFLK(NPFLKA)) )
	<pre>&amp; * WTFLK (NPFLKA)</pre>
*	
*	
*	
	ELSE TKESUM = TKESUM + TKEFLK (NPFLKA) * WTFLK (NPFLKA)
	END IF
*	
*	
	RADDLY (NPFLKA) = ZERZER
*	Here we ask for the region number of the hitting point.
*	NREG (NPFLKA) =
*	The following line makes the starting region search much more
*	robust if particles are starting very close to a boundary:
	CALL GEOCRS ( TXFLK (NPFLKA), TYFLK (NPFLKA), TZFLK (NPFLKA) ) CALL GEOREG ( XFLK (NPFLKA), YFLK (NPFLKA), ZFLK (NPFLKA),
	8 NRGFLK(NPFLKA), IDISC )
*	
	CALL GEOHSM ( NHSPNT (NPFLKA), 1, -11, MLATTC )
	NLATTC (NPFLKA) = MLATTC
	CMPATH (NPFLKA) = ZERZER
	CALL SOEVSV
	RETURN
*:	
*1	End of subroutine Source
*	End of subroutine Source END



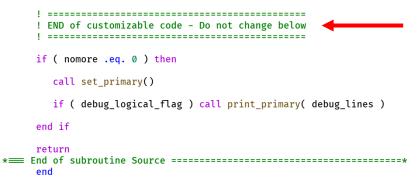
## The "new" source routine

- Distributed since FLUKA4-1.0 release
- Simplified appearance
- Long & meaningful names for variables and routines
- Use of implicit none (see later)
- Abundant comments and examples
- Advanced sampling routines
- Variables for user's usage clearly indicated
- Lines not to be edited are "hidden" in routines. in the source library.inc library file
- Old source routines can still be used

**Removed from** snapshot



- heavyion atomic number = ...
- heavyion mass number = ... \*
- heavyion\_isomer = ... \*
- \* radioactive\_isotope = .true.
- momentum\_energy = ... \*
- energy\_logical\_flag = .true. \*
- particle\_weight = ...
- divergence  $x = \dots$
- divergence  $y = \dots$
- gaussian divergence logical flag = .true. \*
- coordinate x = ...
- coordinate\_y = ...
- coordinate z = ...
- direction cosx = ...
- direction\_cosy = ...
- direction cosz = ...
- direction flag = ... \*
- \* polarization cosx = ...
- polarization cosy = ... \*
- polarization cosz = ...
- particle age = ...
- kshort component = ...
- delayed\_radioactive\_decay = ...





### The "new" source routine

- Without removing comments, examples and advanced features (notice the ratio of code and comment lines)
- Note: the snapshot is not meant to be read Detailed view will follow

a #	117 *== Source	233 ! Default: 234 ! Momentum calculated from values set on the BEAM card (if present), 200 GeV/c otherwise	369 350 I 5.2. Sampling functions and subroutines	465 * delayed_radioactive_decay = 466
Copyright (C) 2020: CERN	119 subroutine SOURCE ( nomore )	235 236 • momentum_energy =	351	467 468   8. Sampling from phase space file
<ul> <li>All Rights Reserved.</li> </ul>	127 use source_library 122 use source_variables	237 and a contract of contract	353 354   5.2.1. Flat distribution	449
<ul> <li>Source routine or FLUKA 4:</li> </ul>	123	239 13.2. Energy flag		470 ! Allows to read particle information from a phase space file and sets the primary accordingly 471
* * Created on 24 September 2028 by David Horvath & Roberto Versaci *	124 implicit none 125	240 241   Select between nomentum and energy	356 1 Replace (a) with "x", "y", or "z". 357	472 473 ! 8.1. Input variables
* ELI Beamlines *	126 logical lfirst	242 I If the energy flag is: 243 I .false. : The momentum energy variable contains the momentum of the particle	<pre>&gt;&gt;&gt; coordinate_[a] = sample_flat_distribution( [min], [max] )</pre>	474
Modified on 17 November 2028 by David Horvath & Roberto Versaci	128 data lfirst / .true. /		359 360	475   - [filoname] 476   - [energy unit]:
ELI Beamlines	129 130 integer nomere	245   Default: 246	361   5.2.2. Gaussian distribution	477 Possible [emergy unit]s: "TeV", "GeV", "MeV", "keV" "eV" 478 "TeV/c", "GeV/c", "MeV/c", "keV/c", "eV/c"
<ul> <li>This is a simplified user written source routine utilizing a</li> <li>separate source routine library.</li> </ul>	111	247 248 * energy_logical_flag = .true.	363 i Replace [a] with "x", "y", or "z".	478 "TeV/c", "GeV/c", "MeV/c", "NeV/c", "NeV/c" 479 I - [length_unit]: 480 Possible [length_unit]s: "Nu", "cn", "cn"
·	132 logical debug_logical_flag 133 data debug_logical_flag / .false. /	249 energy_togical_stag = .true.	365 + coordinate_[a] - sample_gaussian_distribution( [mean], [fmhm] )	480 i Possible [length_unit]s: "ka", "m", "cm", "nm"
<ul> <li>It is intended as an alternative new-user-friendly version of the *</li> <li>source, f routine, Existing FLUKA &amp; source routines remain</li> </ul>	135 integer debug lines	250 251   3.3. Particle weight	366 367	481 ! - [sequential_logical_flag]: 482 ! Possible values;
<ul> <li>compatible.</li> </ul>	136 data debug_lines / 100 /	252 253 I Sets the initial weight of the primary	368 1 5.2.3. Annular distribution	483 .false. : Particles from the phase space file selected randomly 484 .true. : Particles from the phase space file read sequentially, the simulation stops if all particles has been read
. Note that the beam card still has some meaning - in the scoring the +	138 double precision xdammy	254 Default:	Applies an annular distribution to any two coordinates	485
<ul> <li>maximum momentum used in deciding the binning is taken from the</li> <li>beam nomentum. Other beam card parameters are obsolete.</li> </ul>	139 140 type(phase_space) phase_space_entry	255 1.800	371 i Input variables: 372 i - rmin [cm]	487   8.2. File format
• Output variables:	141 142   Function declarations	257 * particle_weight =	373 1 - rmax [cm] 374 1 - Two coordinates of the center of the annular distribution (coordinate_(a/b)) [cm]	488 1 The phase space file has to contain the following columns in this order:
	151	259	375 1 Replace [a] and [b] with "x", "y", or "z".	450 461 - Particle code [integer]
nomore = if > 0 the run will be terminated	144 double precision sample_flat_momentum_energy 145 double precision sample_gaussian_momentum_energy 146 double precision sample_maxwell_boltzmann_energy	200   3.4. Sampling functions and subroutines	376 I Output variables: 377 - Nodified coordinates of the sampled location (input values have been overwritten)	492
•	146 double precision sample_maxwell_boltzmann_energy 147 double precision sample_histogram_momentum_energy	202	<pre>378 * call sample_annular_distribution( [rmin], [rmax], coordinate_[a], coordinate_[b] )</pre>	493 ! - Particle momentum / energy [double precision]
Quick start guide:		264 1 3.4.1. Flat distribution	375 * Catt Smpte_mnuter_styline finitel, finitel, containete_sal, containete_sal,	495   - Starting X coordinate [double precision]
	160 double precision sample_gaussian_distribution 150 double precision sample_flat_distribution	200	381 382   6. Beam direction	496   - Starting Y coordinate [double precision] 497   - Starting Z coordinate [double precision]
This over source routine template aims to modernize the legacy routine by implementing modern fortran conventions and to provide built in	151 152 double precision FLMDM	<pre>267 * momentum_energy = sample_flat_momentum_energy( [min], [max] ) 144</pre>	383 !	498 499 ! - Starting X direction cosine [double precision]
sampling functions.	153	249	385 1 6.1. Divection cosines	<ul> <li>540   - Starting X direction cosine [double precision]</li> <li>541   - Starting Z direction cosine [double precision]</li> </ul>
The users only need to change / add code between the BIGINWING and END	155 BEGINNING of user declared variables	270 1 3.4.2. Gaussian distribution	382	Sei   - Starting Z direction cosine [double precision]
<ul> <li>marks, one section for declaration of user variables, and one for</li> <li>assigning values to the beam parameters.</li> </ul>	15	272 273 • momentum_energy = sample_gaussian_momentum_energy( [mean], [fwhm] )	388 I Sets the direction cosines of the beam with respect to the X,Y and Z-axis 100 I Defemtion	543   - Particle weight [double precision]
	158	224 meeting(extly - subjectionseconcertal( (even), (even) )	390 1 Set on the BEAPODS card if present, otherwise (0.000, 0.000, 1.000)	504 585
<ul> <li>By default there is no user variable defined, and all code lines for parameter assignment are commented out. These comments start with the symbol: "*. To enable one, the '* needs to be deleted.</li> </ul>	159	275 276 ! 3.4.3. Maxwell-Boltzmann distribution	391 392 • direction coss =	586 8.3. Output variables
symbol: '*'. To enable one, the '*' needs to be deleted.	161 1 END of user declared variables	277 278 Temperature is given in GeV, energy flag must be .true.	393 * direction_cosy = 394 * direction_cosz =	380 - phase_space_entry: Variable containing information of a single particle from the phase space file
<ul> <li>(Note: In Fortran each code line should start in column 7 or further in.)</li> </ul>	163	279	355	509 I - nomore: Flag to indicate that all particles has been read 518
Every beam parameter has a default value based on the FLUKA input file.	164 nomore - 0 185	<pre>200 * momentum_energy - sample_maxwell_boltzmann_energy( [temperature] ) 201</pre>	395 397   6.2. Direction flag	511
<ul> <li>A parameter assignment should only be used if the default value has to be modified.</li> </ul>	166 if ( lfirst ) then 167 call initialization()	282 283 1 3.4.4. Sampling from histogram	398 1 Sets how the direction cosines are treated	512 8.4. Subroutine call
	100 lfirstfalse.	285	400 ! Possible values: 401 . 8 : All 3 direction conines are taken into account (values will be normalized)	<pre>514 515 * call read_phase_space_file( [filename], [energy_unit], [length_unit], phase_space_entry, [sequential_logical_flag], nomore )</pre>
There are three ways to assign a value to a parameter:	109 end 11 170	285 Possible [unit]s: 'TeV', 'GeV', 'NeV', 'NeV' 'eV' 286 'TeV/c', 'GeV/c', 'NeV/c', 'NeV/c', 'eV/c'	402 1 : Only direction cosines with respect to the X- and Y-axis are taken into account, the third cosine (2) is calculated	515 Cart read_mase_space_rice (ricename), lenergy_unit), [congin_unit], phase_space_entry, [sequentiat_rig], neere ) 516
* I. Direct assignment: A parameter is equal to a value. For example:	171 BEGINNING of customizable code	287   Histogram file has to have 3 columns: 288   - Emin (of the bin)	403 1 with a positive sign 404 1 2 : Only direction cosines with respect to the X- and Y-axis are taken into account, the third cosine (Z) is calculated	517 518   8.5. Reading information from 'phase_space_entry' variable
<pre>momontum_energy = 0.150</pre>	173	200 - Enax (of the bin) 200 - dK/dt (bin height; KOTE: doesn't need to be normalized)	465 1 with a negative sign 406 1 Default:	519 520 The information stored in the 'phase_space_entry' has to be copied to the appropriate variables
<ul> <li>If the parameter defined as a double precision, then the assigned</li> </ul>	174		400 I DHT4ULT 407 I 0	81
<ul> <li>value should be represented as double precision as well so as to not loose numerical precision. To do this a 'D' exponential mark</li> </ul>	176 1. Accessing variables from the SOURCE card	292 * momentum_energy = sample_histogram_momentum_energy( [filename], [unit] ) 201	408 409 x direction_flag	522 * particle_code = phase_space_entrySpc 523 * momentum_energy = phase_space_entrySm_e
<ul> <li>must be used.</li> </ul>	178 Values set on the SOURCE card can be accessed with the following variables:	294 295   3.4.5. Exponential distribution	410	576
<ul> <li>2. Using a sampling function: A parameter is assigned to a value, which is calculated by a separate function. For example:</li> </ul>	179 Numerical values (double precision): 180 NUASOU(1), MMATSOU(2),, WMASOU(18)		412 1 6.3. Sampling functions and subroutines	<pre>\$25 * energy_logical_flag = .true. \$26</pre>
	181 SDUM text (8 character): 182 SDUSDU	297 Input variables: 288 - e min [Gev]	413	527 * coordinate_x - phase_space_entry%x 528 * coordinate_y = phase_space_entry%y
<pre>coordinate_x - sample_flat_distribution( [min], [max] )</pre>	183	200 - e.max (GeV) 300 - intensity ratio = (int e.max / int e.min)	415 1 6.3.1. Isotropic distribution	529 * Coordinate_r = phase_space_entrys 529 * Coordinate_r = phase_space_entrys
The parameters between the '[' and ']' brackets need to be replaced	185 ! 2. Primary particle	301 Output variables:	417 ! Output variables:	530 * direction_cosx = phase_space_entry%u
with numbers, or user variables containing the desired values.	105	382 - momentum_energy 383 - particle weight	418 I - direction_cosx 439 I - direction_cosy	532 * direction_cosy = phase_space_entry%v 533 * direction_cosz = phase_space_entry%v
<ol> <li>Using a sampling subroutine: They are similar to function, but they are not returning values directly; instead they modify the variables</li> </ol>	188 189   2.1. Particle code	384 385 * call sample_exponential_energy_weight( (e_min), (e_max), [intensity_ratio], momentum_energy, particle_weight )	420 ! - direction_cosz	534
<ul> <li>in their argument list. For example:</li> </ul>	190	200 circ improvements/energy/method (a/mail (a/mail (a/mail (a/mail) (a/mail))) and an	<pre>422 * call sample_isotropic_direction( direction_cosy, direction_cosy , direction_cosy )</pre>	515 • particle_weight = phase_space_entry%wei 516
<pre>call sample_annular_distribution( [rmin], (rmax], coordinate_x, coordinate_y )</pre>	191 ! FLUKA particle code of the primary 192 ! See section 5.1 of the FLUKA manual for the list of particle code	387 388 ! 4. Beam angular divergence	423	\$37
The example above has two input parameters between brackets, and two	193   Default: 194   Particle code of the primary defined on the BEAM card if present, otherwise 1 (proton)	309 1	425 1 7. Other chargeable parameters	538 9. Debugging 539
	195	311	127 I for most of the uses none of these parameters should be changed from the defaults	540
provided, as for functions. The output parameter names usually don't need to be changed, but there are cases, where a subset of possible	195 * particle_code = 197	312 4.1. Divergence value	429	541 542   9.1. Debug logical flag
output parameters has to be selected.	198 199   2.2. Heavy ion	314 I Sets the beam divergence in the X-2 (divergence_x) and Y-2 (divergence_y) planes [rad] 315 I Divergences are applied before beam direction	438 1 7.1. Polarization cosines	543 Enables or disables the printout of the beam parameters for debugging
For further details see the FLUMA manual.	200 If the HEAVION particle type (particle code = -2) has been selected on the BEAM card.	315 Defaults: 317 Set on the BEAN card if present (converted to radians), 0.0 otherwise	The three inputs indicate the direction cosines of the particle polarization	Set         1. fails.         1 bing output disable(d) for output disable(d)           547         1. rfails.         1 bing output disable(d)
	202 I the ion can be specified with the following parameters:	318	433   DOFMALTS  434   (-2.006, 0.000, 0.000)	547 . true. : Debug output anabled
module source variables	203 Defaults: 204 Specified on HI-PROPE card (if present), otherwise Z+6, A+12, I+0 (12C)	319 • divergence_x = 329 • divergence_x =	435 + polarization cotx +	548 549 * debug_logical_flag = .true.
imlicit note	285 286 • beaution atomic number :	321	437 * polarization_cosy = 438 * polarization_cosy =	550
	207 • beaution mass number =	323 4.2. Divergence type	439 polarization_cosz = 439	551 552 9.2. Debug Lines
<pre>integer, save :: particle_code integer, save :: heavyion_stomic_number, heavyion_mass_number, heavyion_isomer</pre>	208 * heavylon_isomer =	324 325 Selects between flat and Gaussian divergence.	440 441   7.2. Particle age	553 554 ! Sets the maximum number of lines printed in the debug output
logical, save :: radioactive_isotope	210 211   2.3. Radioactive isotope		442 1 443 1 Sets the starting age of the primary particle in seconds	555 Default:
double precision, save :: momentum_energy, particle_weight		<ol> <li>false. I The divergence is flat - Divergence values are taken as full opening angle</li> <li>true. : The divergence is Gaussian - Divergence values are taken as FWHM of the distribution</li> </ol>		556   100 557
logical, save :: energy_logical_flag	213 ! Selects if the specified ion is to be used as a radioactive source 214 ! Default:	329 ! Default: 330 Set in the BIAN card (if present), .false, otherwise	445 ! 0.000 446	SS8 * debug_lines - 100
<pre>double precision, save :: divergence_x, divergence_y logical, save :: gaussian_divergence_logical_flag</pre>	215 true. : if ISOTOPE particle type is selected on the BEAN card, false. : otherwise	331 332 * gaussian_divergence_logical_flag = .true.	647 * particle_age =	548
	217   Note:	333	449	551   IND of customizable code - Do not change below
double precision, save :: coordinate_x, coordinate_y, coordinate_z	218 Requires RADDECAY (semi-analogue) and DCYSCORE cards Direction sampling is always isotropic	334 335   5. Beam starting position	450 1 7.3. Kshort component 451	543 Contractory of the providence of the provide
integer, save :: direction_flag double precision, save :: direction cosx, direction cosx, direction cosx	220 ! Momentum / energy settings are disregarded	336	1 Sets The Kshort component of the KB/K8bar	564 565 if (nemore.eg. 0) then
	<pre>221 222 * radioactive_isotope = .true.</pre>	338	453   Default: 454   -2.40	540 567 call set primary()
double precision, save :: polarization_cosx, polarization_cosy, polarization_cosz	223	339 ! 5.1. Coordinates	455 456 * kshort_component =	568
double precision, save # particle_age double precision, save # kshort_component	225 1 3. Particle momentum / energy and weight	341 Sets the starting coordinates (x,y,z) of the beam [cm]		<pre>559 if ( debug_logical_flag ) call print_printry( debug_lines ) 578</pre>
double precision, save :: Ashort_component double precision, save :: delayed_radioactive_decay	220 227	342 I Defaults: 343 I Coordinates set on the BEAMPOS card if present, (0.000, 0.000, 0.000) otherwise	408 409 I 7.4. Delayed radioactive decay	571 end if
end module source variables	228 229   3.1. Homentum & Energy	344 345 • coordinate_x •	460 1	372 573 return
	218	346 * coordinate v =	442   Default: 462   0 efault: 463   0 .000	574 x= End of subroutine Source
include 'source_library.inc'	231 Set the momentum [GeV/c] or the kinetic energy [GeV] of the primary particle	347 * coordinate_z =	ARE 1 8-400	272 BIN



## **History of Fortran**

• Fortran born in the early 1950s, and the first compiler was released in 1957

#### Standards:

- Fortran 66 The first standard
- Fortran 77 Extension on Fortran 66
- Fortran 90 Dynamic memory allocation / introduction of the *Free* format
- Fortran 95 High performance Fortran specification
- Fortran 2003 Object oriented programming
- Fortran 2008 / 2018 Extensions of Fortran 2003

FLUKA is still mostly (if not fully) compatible Fortran 77 This doesn't mean that we can't use newer things in our user routines



# (Unexpected) Features and limitations of Fortran (77)

### Source file format

- Fixed
- Free
- Naming convention
- Subprograms
  - Functions
  - Subroutines
- Variable declaration
  - Implicit
  - Explicit



### **Source file format**

- Fortran 77 uses the *Fixed* file format (extensions: .f or .for):
  - Maximum 78 characters in one line
  - First 6 are reserved for special function:
    - If the first character is 'c' or '\*', then the line is a comment
    - If the 6<sup>th</sup> position is not empty, then the line is treated as a continuation of the previous one (Often the '&' character is used)
  - With the gfortran compiler it is possible to increase the maximum line length
    - In FLUKA 4 it is extended to 132 characters
- Fortran 90 introduced the *Free* format (extensions: .f90, [.f95, etc.]):
  - Code can start at the 1<sup>st</sup> position
- Note: It is not possible to mix both in the same source file.
   Gfortran compiler expects the "correct" format based on the file extension.



## Naming convention

- Fortran 77 variable and (subprogram) names:
  - Limited to 6 alphanumerical characters
  - Have to start with a letter
  - Case insensitive
- Starting with Fortran 90 the variable names
  - Can be up to 31 character long
  - Can contain letters, numbers and underscore ('\_')
  - Have to start with a letter
  - Case insensitive
- Note: Try to use descriptive names, to make code readable

Feature exploited in the new source routine



# Subprograms

### • Two types:

- Function
  - Has a return value
  - Used in assignment: **variable = function(input\_variable\_1**, ...)
- Subroutine
  - Doesn't have a return value
  - Accessible with the CALL statement: call subroutine (input\_variable\_1, ...)

### Passing variables

- In Fortran you pass the variable, not the value of the variable (Like passing a pointer in C)
- This means the subprograms may irreversibly modify the value of the input variables
  - Desired behavior if you want to return multiple variables
  - Can lead to side effects



## **Variable declaration**

- Fortran by default uses *implicit declaration*, which means the type of the variable (integer, real, etc.) is determined by a preset rule.
- The default rule is:
  - If the variable starts with the letter I, J, K, L, M, or N it is an integer
  - Otherwise, it is a real (single precision float)
- In FLUKA however:
  - Variables with the 1<sup>st</sup> letter I, J, K, L, M, and N are still integers
  - But the others are double precision (floats)
- It is possible (and necessary) to overwrite this with *explicit declaration*, where you manually specify the type of the variable, like:
   double precision my\_intensity
   logical my flag



## **Variable declaration**

#### • Biggest issue is that typos remain hidden:

If you have a typo in a variable name, the compiler won't raise an error It is a different, but valid variable without a value Using it in calculations will lead to unexpected results

#### • Other issue is the unexpected type conversion:

For example: Information is lost if you want to assign a double precision number to INTEGER

#### Solution in the "new" source routine: implicit none

This statement disables the implicit declaration, and every variable has to me manually declared <u>Exception</u>: FLUKAs built in variables don't need to be declared in the source routine (they will remain implicitly declared)

#### • Convention in the "new" source routine:

- Variables with uppercase names: FLUKA variables
- Variables with lowercase names: explicitly declared variables



### **Numbers and Constants in User routines**

#### • To keep the high accuracy of the calculation

- Every variable containing a floating-point number should have the type *double precision*
- The assigned numbers should also be double precision:

For example: **radius** = **2.0D0**, Or in a function: **variable** = **function(1.0D0)** The 'D' character indicated, that this is number should be treated as double precision. If it is 'E' or missing, then the number will be single precision

- To simplify writing numbers FLUKA already defined many numbers as variables:
  - ONEONE = 1.0D0
  - **TWOTWO** = 2.0D0
  - HLFHLF = 0.5D0
  - PIPIPI =  $\pi$  = 3.141592...
  - TWOPIP =  $2\pi$  = 6.283185...

Full list available in the dblprc.inc include file



### **Source routine – Initialization**

154	1	
155	1	BEGINNING of user declared variables
156	1	
157		
158		
159		
160	1	
161	1	END of user declared variables
162	1	

• Dedicated space for the declaration of user variables (and functions)



### **Source routine – Initialization**

```
166 if ( lfirst ) then
167 call initialization()
168 lfirst = .false.
169 end if
```

- Initialization of internal variables
- Only performed the first time the routine is called
- To overwrite the default values the relevant lines needs to be uncommented, by removing the '\*' at the beginning of the line.
   (See next slides)



### **Source routine – Primary particle**

196 \* particle\_code = ...

- By default, the particle type given in the **BEAM** card is taken
- Particle codes explained in FLUKA manual section 5.1
- Possible application: beam made of more than one type particles

206	*	heavyion_atomic_number =
207	*	heavyion_mass_number =
208	*	heavyion_isomer =

- Only used if primary particle is set to HEAVYION or ISOTOPE
- Default values are set on the **HI-PROPE** card, or for <sup>12</sup>C if the card is missing



### **Source routine – Energy / momentum**

236 \* momentum\_energy = ...

- By default, the particle <u>momentum</u> is expected
- The default value is based on the **BEAM** card (Automatically converted into momentum if energy is given on the **BEAM** card)
- If energy is specified in the source routine, the following logical value must be set .true.

248 \* energy\_logical\_flag = .true.



## **Source routine – Energy / momentum**

- The momentum divergence set on the **BEAM** card is not retained
- It in necessary to specify in the source routine
- It is easy with the supplied functions / subroutine

Flat spectrum:	267 *	<pre>momentum_energy = sample_flat_momentum_energy( [min], [max]</pre>
Gaussian spectrum:	273 *	<pre>momentum_energy = sample_gaussian_momentum_energy( [mean], [</pre>
Maxwell-Boltzmann spectrum:	280 *	<pre>momentum_energy = sample_maxwell_boltzmann_energy( [temperat</pre>
Spectrum from histogram:	292 *	<pre>momentum_energy = sample_histogram_momentum_energy( [filenam</pre>
Exponential spectrum: (biased sampling)	305 *	<pre>call sample_exponential_energy_weight( [e_min], [e_max], [in</pre>



### **Source routine – Particle weight**

257 \* particle\_weight = ...

- Monte Carlo concept for biased sources
- The default value (particle\_weight = 1.0) is usually sufficient
- Not for a beginners' use, mentioned here for completeness
- Note: The exponential spectrum sampling subroutine, uses variable particle weight



### **Source routine – Beam divergence**

319 \*divergence\_x = ...320 \*divergence\_y = ...

- By default:
  - values are taken from the **BEAM** card
  - It is assumed to be a flat angular distribution
- For Gaussian divergence the following logical value must be set .true.

332 \* gaussian\_divergence\_logical\_flag = .true.



## **Source routine – Beam starting position**

345	*	coordinate_x =	•••
346	*	coordinate_y =	•••
347	*	coordinate_z =	•••

• By default, values are taken from the **BEAMPOS** card

- Beam shape set on the **BEAM** card, and
- Extended sources specified on additional **BEAMPOS** cards are not implemented



## **Source routine – Beam starting position**

Some predefined routines (2 functions and 1 subroutine) are already available:
 Flat distribution:

coordinate\_[a] = sample\_flat\_distribution( [min], [max] ) 358 \* Gaussian distribution: coordinate\_[a] = sample\_gaussian\_distribution( [mean], [fwhm] ) 365 \* Annular distribution: call sample\_annular\_distribution( [rmin], [rmax], coordinate\_[a], 379 \* Remember the values must be in double precision (1.0D0).

Note: If annular sampling is used, the coordinates has to be set manually as well.



### **Source routine – Beam direction**

392	*	direction_cosx =	•••
393	*	direction_cosy =	•••
394	*	direction_cosz =	•••

• By default, values are taken from the **BEAMPOS** card

#### • If the direction\_flag is set to: 409 \* direction\_flag = ...

- 0 : All three values are considered and the they are normalized automatically (Default)
- 1 : The manually set value of the z direction is disregarded. Instead, it is calculated from the x and y direction cosines with a positive sign.
- 2 : As with option 1, but negative sign is used.
- A predefined subroutine is are already available for isotropic direction sampling
- 422 \* call sample\_isotropic\_direction( direction\_cosx, direction\_cosy, direction\_cosz )



## **Source routine – Debugging**

- To help debug the source routine, the major particle parameters can be printed
- To enable this feature, set

549 \* debug\_logical\_flag = .true.

- The printed parameters:
  - Energy / momentum
  - Coordinates
  - Direction
  - Weight
- The number of primaries printed can be set with:

558 \* debug\_lines = 100



## Some predefined FLUKA random sampling routines

- FLUKA offers some useful, predefined routines for random sampling
- my\_variable = FLRNDM(XDUMMY)

Assigns a 64-bit random number in [0,1)

• call FLNRRN (gauss1)

Returns a Gaussian distributed random number

• call FLNRR2 (gauss1, gauss2)

Returns two uncorrelated Gaussian distributed random numbers

• call SFECFE (sint, cost)

Returns sine and cosine of a random azimuthal angle



### **SOURCE card and passing parameters**

- To invoke a source routine, it is necessary to add a **SOURCE** card
- A **SOURCE** card can be empty or can be used to pass parameters to the routine
- Max. 18 numerical values (WHASOU (ii)) and 1 string (max. 8 characters) (SDUSOU) can be

<b><sup>‡</sup> SOURCE</b>	#1:7.	#2: 250.	#3: 12.5
sdum: linksour	#4: 3.75	#5:	#6:
	#7:	#8:	#9:
	#10:	#11:	#12:
	#13:	#14:	#15:
	#16:	#17:	#18:

#### Good practice advice:

Even if the beam energy / momentum is defined in the source routine, specify it in the **BEAM** card as it is used for internal initialization. Set a momentum value higher than the maximum possible one.



# Adding the user routine to the project folder

- 1. Open [Compile] tab
- 2. It is maybe hidden in the dropdown menu
- Click the [Database] button (Use [Add] for an existing file)
- 4. Select the user routine you want to use
- 5. Click [Copy to Project]

The copied user routine will be in the Flair projects directory

💮 + /mnt/d/WSL/Run/course/sou				- 🗆 🗙
🔜 🄄 👻 📔 🙀 Flair 🛛 🔯 Ir	put 📸 Geometry 🛛	🛕 Compile 🛛 🚴 Ru	n 🔝 Plot	🛛 🛛 🕼 Viewer 🔻 🗊
Paste Copy : Save As	Add + Database + Move C 3 Files	Јр 🕘 🖲 🥒 🦉	uka V Clean Build	2
©		Compile	Action	▼ ×
Executable:	×	Options:		
File	Туре	Size	Date	
	- 71			
X FLUKA User rout	nes			×
File▲ Siz	e Date		Desc	
rflctv.f 1149				A
rfrndx.f 1441			tical photons)	
soevsv.f 2554				
source.f 9203		generate any distribu	ition for source particles	
	4 2021.02.12 08:12:4			
stupre.f 3909				
stuprf.f 1667			rons, muons and neutrinos)	
ubsset.f 5265			g parameters	
udcdrl.f 2107				
usimbs.f 3017			e biasing	
usrein.f 1233				
usreou.f 1463				
usrglo.f 1460				
usrhsc.f 2700			isity) Scaling factors	
usrini.f 1314				
usrmed.f 2438			biasing and/or particle sele	ctions on a
usrout.f 1225			addual auglei ageria -	
usrrnc.f 1128			esiduai nuclei scoring	
ustckv.f 908	2021.02.12 08:12:4 Us		hoton production	
wvlnsh.f 2489	2021.02.12 08:12:4 Wa	ave Length Shifted p	noton production	
5 <u>C</u> opy to Project	<u>S</u> can Input <u>V</u> iew	Close		
		Files: 1		🖬 🎽



# **Compiling a custom FLUKA executable**

- 1. Verify that the user routine is in the list
- 2. Name your custom executable
- 3. Select the appropriate linker:
  - a. Use Ifluka by default
  - b. Use *ldpmqmd* if DPMJET or RQMD models are needed
- 4. Compile the executable

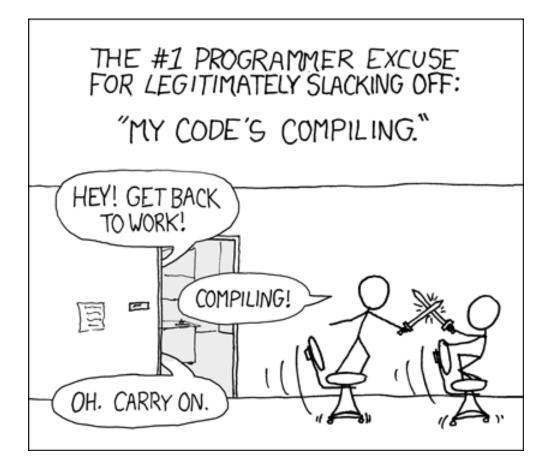
The custom executable should be set default on the [Run] tab automatically

💭 + /mnt/d/WSL/Run/course/sou	urce_routine.flair - flair			– 🗆 🗙
🔜 🧐 👻 🤗   🛛 🌉 Flair 🛛 🔯 Ir	nput 💕 Geometry 🛕			🕠 Viewer 🔻 🕥
Paste Copy Copy Copy Copy Copy Copy Copy	Add   Add   Files	00 🥒 👘	3 uka Clean Build Action	
©		Compile		▼ ×
Executable: myfluka.exe 2		Options:		
File	Туре	Size	Date	
source_newgen.f	Fortran	19124	2021.06.02 13:25:11	<u> </u>
1				
		Files: 1		🔳 🕺



### Time to do some hands-on practice!

• We will now see together a few small examples of "new" source routine



xkcd.com/303



Source routines

