

# **Source routines**

An introduction to a new approach to source routines

Beginner online training, Fall 2020

# 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"
- UR are beyond the scope of this course because of intrinsic difficulties
- Nevertheless, we have a started an effort to make URs more user-friendly
- We want to introduce here the first effort in this direction: a new format for 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)

L LANGE (FILM) - FREE /	* Statement function:	*	Flag this is prompt radiation	Particle generation (1 for primaries)	TKESU	JM = TKESUM + TKEFLK (NPFLKA) * WTFLK (NPFLKA)
<pre>Label VEX.100 / Label VEX.10 - 1 / Label VEX.1</pre>	LISNUT (IJ) = INDEX ( PRNAME (IJ), 'NEUTRI' ) .GT. 0		LRADDC (NPFLKA) = .FALSE.	LOFLK (NPFLKA) = 1	*	
<ul> <li>MAID VERING</li> <li>Pack Signal Solution</li> <li>Pa</li></ul>	**	*				
<pre>     transmission     transmission</pre>	* *					1
<pre>term::::::::::::::::::::::::::::::::::::</pre>	* BASIC VERSION *	*				
<pre>L MART - 4 L MART 1000L - 100L -</pre>	*		IRDAZM (NPFLKA) = 0		TKESU	
<pre>i</pre>	**	*	Particle age (s)		6	* WTFLK (NPFLKA)
<pre>     For this table is table is table is table if the state is table i</pre>	NOMORE = 0		AGESTK (NPFLKA) = +ZERZER	* Extra infos:	*	
<pre>     // If (JUNCT) THEM     // IF (JUNCT</pre>	* +*	*	Kinetic energy of the particle (GeV)	INFSTK (NPFLKA) = 0	* +	*
<pre>       ** The following levels are metaboury ***     Law in the following levels are metaboury ****     Law in the following levels are metaboury ************************************</pre>	* First call initializations:		TKEFLK (NPFLKA) = SQRT ( PBEAM**2 + AM (IONID)**2 )	LNFSTK (NPFLKA) = 0	*	
<pre></pre>	IF ( LFIRST ) THEN		& - AM (IONID)	ANFSTK (NPFLKA) = ZERZER	ELSE	
<pre>Profix further = Halfs UNDER = Field UN</pre>				* Parent variables:	TKESU	JM = TKESUM + TKEFLK (NPFLKA) * WTFLK (NPFLKA)
<pre>LTRMS = FAIG LOOK LTRMS = FAIG LTRMS = FAIG LOOK LTRMS = FAIG LTRMS = FAIG LOOK LTRMS = FAIG LOOK</pre>				IPRSTK (NPFLKA) = 0	END IF	
<pre>Lings:</pre>					* 1	
<pre>L LFUER (WTLAL) = TABLE L DE UT L DE UT</pre>					* +	*
<pre>DBLT DEATH (1075,07124) = DEATH DEATH (1075,07124) = DEATH (1075,07124) = DEATH DEATH (1075,07124) = DEATH (1075,07124) = DEATH DEATH (1075,07124) = DEATH (1075,07124) = DEATH (1075,07124) = DEATH DEATH (1075,07124) = DEATH (1075,07124</pre>					RADDLY	(NPELKA) = ZERZER
<pre></pre>						
<pre></pre>	*		**			
<pre>* Posk one source particle to the stack. How the stack. How the stack on the stack is not thank we retark to a stack of a pase is in the stack context of a pase is in thes</pre>	*					
<pre>public has ye verses a maximum amount of spaces in the stack for the secondaries to be conduct to decurse any time secure is called it has a public his secondaries to decurse any time secure is called it has a public his provide any time secure is called it if (internation secure is a public his provide is the train any time is provide any time secure is a public his provi</pre>						
<pre>* stack for the secondaries to be generated PWFLA = threak controls to be generated PWFLA = threak controls to cores at the stack particle PWFLA = threak controls to cores at the stack particle PWFLA = threak controls to cores at the stack particle PWFLA = threak controls to cores at the stack particle PWFLA = threak controls to cores at the stack particle PWFLA = threak controls to cores at the stack particle PWFLA = threak controls to core at the stack particle PWFLA = threak controls to core at the stack particle PWFLA = threak controls to core at the stack particle PWFLA = threak controls to core at the stack particle PWFLA = threak controls to core at the stack particle PWFLA = threak controls to core at the stack particle PWFLA = threak controls to core at the stack particle PWFLA = threak controls to core at the stack particle PWFLA = threak controls to core at the pwfLA = threak controls to core at the pwfLA = threak controls to core at the pwfLA = threak control to control to core at the pwfLA = threak control to core at the pwf</pre>						
<pre>* Splin is the stack conster of course any time source is called it matk b = 0 writes (First is promit relation) writes (First is promit</pre>						
<ul> <li>mist be = 0         IADAC (NPTLA) = .0.0000         NUTINE = TPRIME         NUTINE = TPRIME         NUTINE = TPRIME         NUTINE = TPRIME         IADAC (NPTLA) = .0.000         NUTINE = TPRIME         IADAC (NPTLA) = .0.000         NUTINE = TPRIME         IADAC (NPTLA) = .0.000         IADAC (N</li></ul>					CALL GEC	
<pre>NPTXA = UPTXA = 1 Note that we there is the type act by the EAN Note that we there is the type act by the EAN Note that the type act by t</pre>					<i>a b b b b b b b b b b</i>	
<pre>* We is the weight of the particle WFTXA (UPFXA) = 0 WFTXA (U</pre>						
WTEXK (NFTLA) = 0.00000F       **       Parent radioacture incodeps: INDAM NUTLAN = 0       NEURET (NFTLA) = 0       NEURET (NFTLA) = 0       CALL SETUR (NFTLA) = 0         Particle type (lipton). I/Deam is the type are by the BEAN       *       Particle type (lipton).       NEURET (NFTLA) = 0       CALL SETUR (NFTLA) = 0       CALL SETUR (NFTLA) = 0         If (LISER INFTLA) = NOAL       *       Y Lints (NFTLA) = 0       CALL SETUR (NFTLA) = 0       CALL SETUR (NFTLA) = 0         If (LISER INFTLA) = NOAL       *       NOTON = AN (COND) *       TATLK (NFTLA) = 0       CALL SETUR (NFTLA) = 0         If (LISER (NFTLA) = 0       NOTON = AN (COND) *       *       NOTON = AN (COND) *       TATLK (NFTLA) = 0         If (LISER (NFTLA) = NOAL       *       NOTON = AN (COND) *       TATLK (NFTLA) = 0       NOTON = AN (COND *         If (LISER (NFTLA) = NOAL       *       *       NOTON = AN (COND *       NOTON = AN (COND *         If (LISER (NFTLA) = NOAL       *       *       NOTON = AN (COND *       *       *         If (LISER (NFTLA) = NOAL       *       *       NOTON * AN (COND *       *       *       *         If (LISER (NFTLA) = NOAL       *       *       NOTON * AN (COND *       *       *       *       *       *         If (LISER *       INSER (NFTLA) = NOAL       *		*				
<pre>VEIPIFI = WEIPAFI = W</pre>						
Particle type (1=proton). 1 jbesm is the type act by the BEAM       *       Particle age (1)       Additional control isotope:       *to this point don't change anything       ************************************		*				
<pre>card ADSTR NETLAND, = ADSTR NETLAND</pre>						EVSV
<pre> * (Radioactive) isotope: IF (INERAM. 80, -2. AND. LABEL ) THEN IF (INERAM. 80, -2. AND. LABEL ) IF (INERM. 80, -2. AND.</pre>		*				
<pre>*   (Radioactive) isotope: If ( LISEM .E02 .AND. LROBEN .THEN</pre>	* card			AKNSHR (NPFLKA) = -TWOTWO		broutine Source ====================================
<pre>if ( LIDEAM 100 - 2 AND. LEDBEA ) THEM if ( LIDEAM 100 - 2 AND. LEDBEA ) THEM if ( LIDEAM 100 - 2 AND. LEDBEA ) THEM if ( LIDEAM 100 - 2 AND. LEDBEA ) THEM if ( LIDEAM 100 - 4 KIREAV if ( LIDEAM 100 + 100</pre>	* +*	*	Kinetic energy of the particle (GeV)	* Cosines (tx,ty,tz)	END	
IARES = IPROA       *       Particle momentum         IIARES = IPROA       *       Particle momentum         IISRES = IPROA       *       PARTICLE ADDATE         CALL SPIEND (IARES, IERES,						
IERES       IEROX       PHOFUK (NFFLKA) = FBEAM       T TELK (NFFLKA) = SORT (NFFLKA) = SORT (NFFLKA) = CTELK (NFFLKA) * (TELK (NFFLKA) = CTELK (NFFLKA) * (TELK (NFFLKA) = CTELK (NFFLKA) * (TELK (NFFLKA) = CTELK (NFFLKA)				TYFLK (NPFLKA) = VBEAM		
IISRES = IRPOM       * PROMIX (NPFLKA) * (TKFLKA) (TKFLKA) * (TKFLKA) * (TKFLKA) ** (T	IARES = IPROA	*	Particle momentum	TZFLK (NPFLKA) = WBEAM		
CALL STIENU (IARES, IISRES) LIMION = IDRON (1DROZ, 100) * 1000 + IPROA LIMION * 1000 ( DROZ, 100) * 1000 + IPROA LIMION = JUNION * 1000 ( DROZ, 100) * 1000 + IPROA LFRPHN (NPFLKA) = .FALSE. LIMION = IDROM (NPFLKA) = .FALSE. LIMION = IDROM * 10000 + MOD (1PROZ, 100) * 1000 + IPROA LIMION = JUNION * 1000 + MOD (1PROZ, 100) * 1000 + IPROA LIMION = JUNION * 1000 + MOD (1PROZ, 100) * 1000 + IPROA LIMION = JUNION * 1000 + MOD (1PROZ, 100) * 1000 + IPROA LIMION = JUNION * 10000 + MOD (1PROZ, 100) * 1000 + IPROA LIMION = JUNION * 10000 + MOD (1PROZ, 100) * 1000 + IPROA LIMION = JUNION * 1000 + MOD (1PROZ, 100) * 1000 + IPROA LIMION = JUNION * 1000 + MOD (1PROZ, 100) * 1000 + IPROA LIMION = JUNION * 1000 + MOD (1PROZ, 100) * 1000 + IPROA LIMION = JUNION * 1000 + MOD (1PROZ, 100) * 1000 + IPROA LIMION = JUNION * 1000 + MOD (1PROZ, 100) * 1000 + IPROA LIMION = JUNION * 1000 + MOD (1PROZ, 100) * 1000 + IPROA LIMION = JUNION * 10000 + MOD (1PROZ, 100) * 1000 + IPROA LIMION = JUNION * 1000 + MOD (1PROZ, 100) * 1000 + IPROA LIMION = JUNION * 1000 + MOD (1PROZ, 100) * 1000 + IPROA LIMION = JUNION * 1000 + MOD (1PROZ, 100) * 1000 + IPROA LIMION = JUNION * 1000 + MOD (1PROZ, 100) * 1000 + IPROA LIMION = JUNION * 1000 + MOD (1PROZ, 100) * 1000 + IPROA LIMION = JUNION * 1000 + MOD (1PROZ, 100) * 1000 + IPROA LIMION = JUNION * 1000 + MOD (1PROZ, 100) * 1000 + IPROA LIMION = JUNION * 1000 + MOD (1PROZ, 100) * 1000 + IPROA LIMION * 1000 + MOD (1PROZ, 100) * 1000 + IPROA LIMION * 1000 + MOD (1PROZ, 100) * 1000 + IPROA LIMION * 1000 + MOD (1PROZ, 100) * 1000 + IPROA LIMION * 1000 + MOD (1PROZ, 100) * 1000 + IPROA LIMION * 1000 + MOD (1PROZ, 100) * 1000 + IPROA LIMION * 1000 + MOD (1PROZ, 100) * 1000 + IPROA LIMION * 1000 + MOD (1PROZ, 100) * 1000 + IPROA LIMION * 1000 + MOD (1PROZ, 100) * 1000 + IPROA LIMION * 1000 + MOD (1PROZ, 100) * 1000 + IPROA LIMION * 1000 + MOD (1PROZ, 100) * 1000 + IPROA LIMION * 1000 + MOD (1PROZ, 100) * 1000 + IPROA LIMION * 1000 + MOD (1PROZ, 100) * 1000 + IPROA LIMION	IZRES = IPROZ		PMOFLK (NPFLKA) = PBEAM	* TZFLK (NPFLKA) = SQRT ( ONEONE - TXFLK (NPFLKA)**2		
IJHION = IRKON * 10000 * MOD (IPROZ, 100 ) * 1000 + IPROA       * (	IISRES = IPROM	*	PMOFLK (NPFLKA) = SQRT ( TKEFLK (NPFLKA) * ( TKEFLK (NPFLKA)	* & - TYFLK (NPFLKA)**2 )		
IJHION = LHEON * LOB * KKHEAV       *   Check if it is a neutrino, if so force the interaction       TYPOL (NPFLKA) = *2EXER         IONID = LHEON * LOB * LABEAV       *   Check if it is a neutrino, if so force the interaction       TYPOL (NPFLKA) = *2EXER         CALL DCDION ( TONID )       IF ( LISBUNT (IJBEAM) .AND. LAUFIN ) THEN       FXFLER * CONSTRAINT         LFRPHN (NPFLKA) = .FALSE.       I         ************************************	CALL STISBM ( IARES, IZRES, IISRES )	*	& + TWOTWO * AM (IONID) ) )	* Polarization cosines:		
IONID = LJHION       *   (unless the relevant flag has been disabled)       TZDCL (COPELAG) = XERZER         CALL DCDION (IONID)       IF (LISNUT (JJEAM). ANDL INVETN) THEN       *         CALL SETION (IONID)       IFPHN (NPFLKA) = .TRUE.       */         LEFRPHN (NPFLKA) = .TRUE.       */         */       */       */         */ <t< td=""><td>IJHION = IPROM * 100000 + MOD ( IPROZ, 100 ) * 1000 + IPROA</td><td>*</td><td>**</td><td>TXPOL (NPFLKA) = -TWOTWO</td><td></td><td></td></t<>	IJHION = IPROM * 100000 + MOD ( IPROZ, 100 ) * 1000 + IPROA	*	**	TXPOL (NPFLKA) = -TWOTWO		
CALL DEDION (IONID) CALL SETION (IONID) LFRPHM (NPFLKA) = .FALSE. *   * * * * * * * * * * * * * * * * * *	IJHION = IJHION * 100 + KXHEAV	*	Check if it is a neutrino, if so force the interaction	TYPOL (NPFLKA) = +ZERZER		
CALL SETION ( IONID )       LFRPHN (NPFLKA) = .TRUE.       XFLK (NPFLKA) = XBEAM         LFRPHN (NPFLKA) = .PALSE.       YIK (NPFLKA) = XBEAM         V       YIK (NPFLKA) = ZBEAM         V       Not a neutrino         ELSE IF ( IJBEAM .EQ2 ) THEN       LFRPHN (NPFLKA) = .FALSE.         IJHION = IJHON * 10000 + MOD ( IPROZ, 100 ) * 1000 + IPROA       LFRPHN (NPFLKA) = .FALSE.         IJHION = IJHON * 1000 + KXHEAV       LFRPHN (NPFLKA) = .FALSE.         CALL DEDION ( IONID )       END IF         LIAPIN (NPFLKA) = JHION       YERK (NPFLKA) = .FALSE.         ILOPIK (NPFLKA) = JHION       YERK (NPFLKA) = .FALSE.         ILOPIK (NPFLKA) = .FALSE.       IFRON THEN         ILOPIK (NPFLKA) = .FA	IONID = IJHION	*	(unless the relevant flag has been disabled)	TZPOL (NPFLKA) = +ZERZER		
CALL SETION ( IONID )     LFRPHN (NPFLKA) = .TRUE.     XFLK (NPFLKA) = XBEAM       LFRPHN (NPFLKA) = .FALSE.     Not a neutrino     YELK (NPFLKA) = XBEAM       Meavy ion:     ELSE     ELSE       ELSE IF ( IJBEAM .EQ 2 ) THEN     ELSE       IJHION = IDHON * 100000 + MOD (IPROZ, 100) * 1000 + IPROA     ELSE       IJHION = IJHION * 1000 + KAHEAV     END IF       CALL SETION ( IONID )     END IF       CALL SETION ( IONID )     END IF       CALL SETION ( IONID )     END IF       IJHION = IJHION     END IF       ILOFLK (NFFLKA) = JHION     END IF       ILAPLOK (NFFLKA) = JHION     Image: Calculation       ILAPLOK (NFFLKA) = .FALSE.     Image: Calculate in calculate i	CALL DCDION ( IONID )		IF ( LISNUT (IJBEAM) .AND. LNUFIN ) THEN			
LFRPHN (NPFLKA) = .FALSE.       *           *         *         *         *         YEK (NPFLKA) = YBEAM         *         *         *         Not a neutrino       ELSE       Calculate the total kinetic energy of the primaries: don't change         *         *         *         Not a neutrino       ELSE       Calculate the total kinetic energy of the primaries: don't change         *         LFRPHN (NPFLKA) = .FALSE.       ERD IF        (Radioactive) isotope:         IJHION = IJHION * 100 + KXHEAV       END IF            CALL DCDION (IONID)       END IF	CALL SETION ( IONID )		LFRPHN (NPFLKA) = .TRUE.			
<pre>* # # # # # # # # # # # # # # # # # # #</pre>	LFRPHN (NPFLKA) = .FALSE.					
<pre>** *   Not a neutrino *   Heavy ion: ELSE IF ( IJBEAM .EQ2 ) THEN IJH:ON = IPROM * 100000 + MOD ( IPROZ, 100 ) * 1000 + IPROA IJH:ON = IPHION * 100000 + MOD ( IPROZ, 100 ) * 1000 + IPROA IJH:ON = IJH:ON * KXHEAV CALL DCDION ( IONID ) CALL SETION ( I</pre>	*	*	**			
<pre>* Heavy ion: ELSE IF ( IJBEAM .EQ2 ) THEN IJHION = IPROM * 100000 + MOD ( IPROZ, 100 ) * 1000 + IPROA IJHION = IJHION * 100 + KXHEAV IONID = IJHION CALL DCDION ( IONID ) CALL SETION ( IONID</pre>	**	*	Not a neutrino			
ELSE IF ( IJBEAM .802 ) THEN       LFRPHN (NPFLKA) = .FALSE.       *   (Radioactive) isotope:         IJHION = IPHON * 10000 + MOD ( IPROZ, 100 ) * 1000 + IPROA       END IF       IF ( IJBEAM .802 .AND. LRDBEA ) THEN         IONID = IJHION       END IF       *           CALL DCDION ( IONID )       ELSE IF ( ILOFLK (NPFLKA) .EQ2 .OR.       *           ILOPLK (NPFLKA) = IJHION       *         *           *   Flag this is prompt radiation       * From this point       *           LRADDC (MPFLKA) = .FALSE.       * Particle generation (1 for primaries)       *           LRADDC (NPFLKA) = .FALSE.       * Particle generation (1 for primaries)       *	*   Heavy ion:		ELSE			
IJHION = IPROM * 10000 + MOD (IPROZ, 100 ) * 1000 + IPROA       END IF         IJHION = IJHION * 100 + KXHEAV       *           IONID = IJHION * 100 + KXHEAV       *           CALL DCDION (IONID )       END IF         CALL DCDION (IONID )       END IF         CALL SETION (INFICKA) = FALSE.       From this point         CARDOC (NPEKA) = -ALSE.       Farticle generation (1 for primaries)			LFRPHN (NPFLKA) = .FALSE.			
IJHION = IJHION * 100 + KXHEAV       *         IONID = IJHION       *         CALL DCDION (IONID)       END IF         CALL SETION (IONID)       *         CALL SETION (IONID)       *         ILOPLK (NPFLKA) = IJHION       *         *       Flag this is prompt radiation         LRADDC (NPFLKA) = .FALSE.       *         *       Particle generation (1 for primaries)         V       USEK (NPFLKA) = .						
IONID = IJHION       * +		*		* 1 ( 1992) 1987 2 ( 1992) 1 ( 1992)		
CALL DCDION (IONID)       *         CALL SETION (IONID)       *         CALL SETION (IONID)       *         LIDPLK (NPFLKA) = JHION       *         *       Flag this is prompt radiation         LRADDC (NPFLKA) = .FALSE.       *         *       Farticle generation (1 for primaries)         *       VEFLKA) = .FALSE.		*	*	* +*		
CALL SETION ( IONID ) * * * * * * * * * * * * * * * * * *			END IF			
ILOFLK (NPFLKA) = IJHION     * +*     6     ILOFLK (NPFLKA) .GT. 100000 ) THEN       *   Flag this is prompt radiation     * From this point     * TKESUM = TKESUM + TKEFLK (NPFLKA) * WTFLK (NPFLKA)       LRADDC (NPFLKA = .FALSE.     * Particle generation (1 for primaries)     *         *   Grown number for "low" set to 0 anymay     *       *		*				
* Flag this is prompt radiation * From this point LRADDC (NPFLKA) = .FALSE. * Particle generation (1 for primaries) * UDVEK (NPFLKA) = * UDVEK (NPFLKA) * WTFLK (NPFLKA) * WTFLK (NPFLKA) * * * * * * * * * * * * * * * * * * *			*			
LRADDC (NPFLKA) = .FALSE. * Particle generation (1 for primaries) *						
* Group number for "low" energy neutrons set to 0 anumary TOPTY / NOPTYAL - 1				TRESUM - TRESUM + TREFLK (NPFLKA) * WTFLK (NPFLKA)		
-:%%- source.f 22% (72,0) (Fortran) -:%%- source.f 46% (97,0) (Fortran) -:%%- source.f 68% (151,0) (Fortran) -:%%- source.f Bot (204,0) (Fortran)						
	-:%%- source.f 22% (72,0) (Fortran)	-: %	8- source.f 46% (97,0) (Fortran)	-:%%- source.f 68% (151,0) (Fortran)	-: 88- source.	.f Bot (204,0) (Fortran)



#### **Source routines**

# The "new" source routine

- To be distributed in the next release
- Simplified appearance
- Long & meaningful names for variables and routines
- Use of implicit none (see later)
- Abundant comments (removed in the snapshot)
- 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

Comments removed for clarity in this snapshot

double precision sample gaussian distribution double precision sample flat distribution

nomore = 0

end

-:--- source layer short.f

if (lfirst) then call initialization( lfirst ) end if

Beginning of customizable code \_\_\_\_\_

particle code = IJBEAM momentum energy = PBEAM energy logical flag = .false. particle weight = ONEONE divergence x = DIVBM divergence y = DIVBM gaussian divergence logical flag = LDVGSS coordinate x = XBEAM coordinate y = YBEAM coordinate z = ZBEAM direction cosx = UBEAM direction cosy = VBEAM direction cosz = WBEAM direction flag = 0 polarization cosx = -TWOTWO polarization cosy = ZERZER polarization cosz = ZERZER particle\_age = ZERZER kshort\_component = -TWOTWO delayed radioactive decay = ZERZER End of customizable code - Do not change below \_\_\_\_\_ call set internal flags() call set\_beam\_type( particle\_code, ionid ) call set\_particle\_momentum\_energy\_weight( particle\_code, ionid, momentum\_energy, energy\_logical\_flag, particle\_weight ) call set\_particle\_coordinates( coordinate\_x, coordinate\_y, coordinate z ) δ. call set\_particle\_direction( direction\_cosx, direction\_cosy, direction\_cosz, direction\_flag, divergence\_x, divergence\_y, gaussian divergence logical flag ) call set particle polarization( polarization cosx, polarization\_cosy, polarization\_cosz ) δ. call set\_particle\_age( particle\_code, particle\_age, & kshort component, delayed radioactive decay ) call search starting region() return \*=== End of subroutine Source

Bot (76.0



### The "new" source routine

- Without removing comments (notice the ratio code\_lines / comment\_lines)
- Note: the snapshot is not meant to be read
- A step by step look will follow

beam momentum. Other beam card parameters are obsolete. *	double precision sample_gaussian_distribution		<ul> <li>* The Z coordinate of the beam's starting position</li> <li>* Defaults:</li> </ul>	* Input variables:
<b>1</b>	double precision sample_flat_distribution	* Implemented samplings functions	* XBEAM, YBEAM, ZBEAM: Coordinates set on the BEAMPOS card	<pre>* - e_min, e_max, intensity_ratio = (int_e_min / int_e_max) * Output variables:</pre>
Output variables: *		· · · · · · · · · · · · · · · · · · ·	ABEAM, IBEAM, ZEEAM: COOldinates set on the BEAMPOS Card	<ul> <li>irection cosx, direction cosy, direction cosz</li> </ul>
	nomore = 0	* momentum_energy =	coordinate x = XBEAM	<ul> <li>call sample_isotropic_direction( direction_cosx, direction_cosx</li> </ul>
nomore = if > 0 the run will be terminated *	16 ( ) Minet ) then	<pre>* = sample_flat_momentum_energy( [min], [max] )</pre>	coordinate y = YBEAM	* & direction cosz)
	if (lfirst) then	* = sample_gaussian_momentum_energy( [mean], [fwhm] )	coordinate_z = ZBEAM	
*	call initialization( lfirst )	<pre>* = sample_maxwell_boltzmann_energy ( [temperature] )</pre>	coordinate_z = zbbaw	* Other changeable parameters
testate testate the second state	end if	* temperature is given in GeV	* Implemented samplings functions	*
include 'source_library.inc'		<pre>* = sample_energy_from_histogram( [filename], [energy_unit])</pre>	· Implemented samplings functions	* For most of the uses none of these three should be changed from
	* Beginning of customizable code * ====================================	* possible [energy_unit]s: "TeV", "GeV", "MeV", "keV', "eV"	<pre>* coordinate xyz =</pre>	* the defaults
* Source =========*	*		<pre>* = sample flat distribution( [min], [max] )</pre>	- che defaults
		<ul> <li>Implemented sampling subroutines</li> </ul>	<ul> <li>* = sample gaussian distribution( [mman], [max])</li> <li>* = sample gaussian distribution( [mean], [fwhm])</li> </ul>	* polarization cos :
subroutine SOURCE ( nomore )	<ul> <li>Accessing variables from the SOURCE card</li> </ul>		· sample_gaussian_distribution( [mean], [iwnm] )	<ul> <li>* The three inputs indicate the direction cosines of the particle</li> </ul>
	*	<pre>* sample_exponential_energy_importance:</pre>	the two laws to be a sub-sub-sub-sub-	<ul> <li>The three inputs indicate the direction cosines of the particle</li> <li>polarization</li> </ul>
use source_library	<ul> <li>Values set on the SOURCE card can be accessed with the following</li> </ul>	* Input variables:	* Implemented sampling subroutines	
	* variables:	* - e_min, e_max, intensity_ratio = (int_e_min / int_e_max)		<pre>* particle_age:</pre>
implicit none	* Numerical values (double precision):	* Output variables:	<pre>* sample_annular_location:</pre>	<ul> <li>Set the starting age of the primary particle in seconds</li> </ul>
	* WHASOU(1), WHATSOU(2),, WHASOU(18)	<pre>* _ momentum_energy, particle_weight</pre>	<ul> <li>Input variables:</li> </ul>	<pre>* kshort_component:</pre>
logical lfirst	* SDUM text (8 character):	* call sample_exponential_energy_importance( [e_min], [e_max],	* - rmin, rmax [cm]	<ul> <li>The Kshort component of the K0/K0bar</li> </ul>
save lfirst	* SDUSOU	* & [intensity_ratio], momentum_energy, particle_weight)	* Output variables:	* delayed_radioactive_decay:
data lfirst / .true. /			<ul> <li>Two coordinates of the sampled loaction</li> </ul>	* the delay for the radioactive decaye with respect to the
	* Particle type	* Beam angular divergence	<pre>* call sample_annular_location( [rmin], [rmax], coordinate_x,</pre>	<ul> <li>standard primary zero time</li> </ul>
integer nomore	*	*	* & coordinate_y )	
	* particle code:	* divergence x:		polarization_cosx = -TWOTWO
integer particle code, ionid	* FLUKA particle code of the primary	* Beam angular divergence in the X-Z plane	* Beam direction	polarization cosy = ZERZER
	* See section 5.1 of the FLUKA manual for the list	* (before applying beam diretion)	*	polarization cosz = ZERZER
logical energy logical flag	<ul> <li>of particle code</li> </ul>	* divergence y:	* direction cosx:	
double precision momentum energy, particle weight	* Default:	* Beam angular divergence in the Y-Z plane	* Direction cosine of the beam with respect to the X-axis	particle age = ZERZER
	* IJBEAM: Particle code of the primary defined on the	* gaussian divergence logical flag:	* direction cosy:	kshort component = -TWOTWO
logical gaussian divergence logical flag	* BEAM card	<ul> <li>If .true., divergence values are taken as fwhm of a Gaussian</li> </ul>	* Direction cosine of the beam with respect to the Y-axis	delayed radioactive decay = ZERZER
double precision divergence x, divergence y	- DEAN CALU	* distribution	* direction cosz:	dorajoa_ranoacoro_docaj beneen
double precision divergence_x, divergence_y	particle code = IJBEAM	<ul> <li>If .false., divergence values are taken as full angle of a flat</li> </ul>	<ul> <li>Direction cosine of the beam with respect to the Z-axis</li> </ul>	* End of customizable code - Do not change below
deuble encodierte a condicate a condicate a	particle_code = 1355AW	<ul> <li>If faise, divergence values are taken as full angle of a flat</li> <li>distribution</li> </ul>	<ul> <li>direction flag:</li> </ul>	*
double precision coordinate_x, coordinate_y, coordinate_z	* Deptice execution ( encourse and unight	* Note:	* Possible values:	
latered disasting flag	* Partice momentum / energy and weight		<ul> <li>* 0: All 3 direction cosines are taken into account (values will</li> </ul>	call set internal flags()
integer direction_flag	*	* If one of the divergences is equal to zero, while the other is		
double precision direction_cosx, direction_cosy, direction_cosz	* momentum_energy:	<ul> <li>not then the sampling will fail and crash. If divergence in</li> </ul>	* be normalized)	<pre>call set_beam_type( particle_code, ionid )</pre>
	<ul> <li>Set the momentum or the kinetic energy of the primary</li> </ul>	<ul> <li>* only one axis is desired, set the other to an infinitesimal</li> </ul>	<ul> <li>1: Only direction cosines with respect to the X- and Y-axis</li> </ul>	call set_particle_momentum_energy_weight( particle_code, ionid,
double precision polarization_cosx, polarization_cosy	<pre>* energy_logical_flag:</pre>	* one, like 1.0D-12.	<ul> <li>are taken into account, the third cosine is calculated</li> </ul>	& momentum_energy, energy_logical_flag, particle_weight )
double precision polarization_cosz	* Possible values:	* Defaults:	* with a positive sign	call set_particle_coordinates( coordinate_x, coordinate_y,
	<ul> <li>.false. : The momentum_energy variable contains the momentum</li> </ul>	<ul> <li>DIVBM: Divergence set on the BEAM card [in rad]</li> </ul>	<ul> <li>* 2: Only direction cosines with respect to the X- and Y-axis</li> </ul>	<pre>&amp; coordinate_z )</pre>
double precision particle_age	* of the particle	<ul> <li>LDVGSS: Flag indicating Gaussian divergence, set on the BEAM</li> </ul>	<ul> <li>are taken into account, the third cosine is calculated</li> </ul>	call set_particle_direction( direction_cosx, direction_cosy,
double precision kshort_component	* .true. : The momentum_energy variable contains the kinetic	* card	<ul> <li>with a negative sign</li> </ul>	& direction_cosz, direction_flag, divergence_x, divergence_y,
double precision delayed_radioactive_decay	* energy of the particle		* Defaults:	& gaussian_divergence_logical_flag )
	* particle weight:	divergence_x = DIVBM	<ul> <li>* UBEAM, VBEAM, WBEAM: Direction cosines set on the BEAMPOS card</li> </ul>	call set_particle_polarization( polarization_cosx,
Function declarations	<ul> <li>Sets the initial weight of the primary</li> </ul>	divergence y = DIVBM		<pre>&amp; polarization_cosy, polarization_cosz )</pre>
	* Default:	gaussian_divergence_logical_flag = LDVGSS	direction_cosx = UBEAM	call set particle_age( particle_code, particle_age,
double precision sample flat momentum energy	* PBEAM: Particle momentum set on the BEAM card (even if		direction cosy = VBEAM	& kshort component, delayed radioactive decay )
double precision sample gaussian momentum energy	<ul> <li>energy was used on the BEAM card it)</li> </ul>	* Beam starting position	direction cosz = WBEAM	call search starting region()
double precision sample maxwell boltzmann energy		· · · · · · · · · · · · · · · · · · ·		
double precision sample energy from histogram	momentum energy = PBEAM	* coordinate x:	direction flag = 0	return
· · · · · · · · · · · · · · · · · · ·	energy logical flag = .false.	* The X coordinate of the beam's starting position		*=== End of subroutine Source ====================================
double precision sample gaussian distribution	24 2 C C C C C C C C C C C C C C C C C C	<ul> <li>coordinate y:</li> </ul>	<ul> <li>* Implemented sampling subroutines</li> </ul>	end
double precision sample_flat_distribution	particle_weight = ONEONE	<ul> <li>* The Y coordinate of the beam's starting position</li> </ul>	*	
acare process subre rac ascribación	Farance - and an	<ul> <li>the f coordinate of the beam s starting position</li> <li>coordinate z:</li> </ul>	<pre>* sample isotropic direction:</pre>	
nomore = 0	<ul> <li>Implemented samplings functions</li> </ul>	The Z coordinate of the beam's starting position	<ul> <li>Input variables:</li> </ul>	
HOMOLG = V	impromented samplings functions	The z coordinate of the beam's starting position	input variables:     e min e may intensity ratio = (int e min / int e may)	
	-: source_layer.f 27% (122,0) (Fortran)	-: source_layer.f 43% (175,0) (Fortran)	-: source_layer.f 63% (229,0) (Fortran)	-: source layer.f Bot (281,0) (Fortran)



# **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 set to 132
- 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

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...

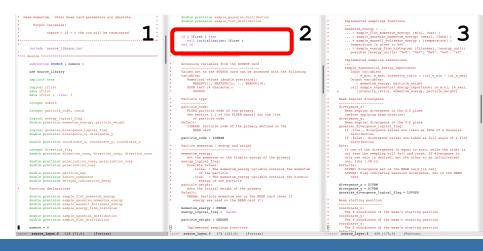
Full list available in the dblprc.inc include file



### **Source routine – initialization**

```
if ( lfirst ) then
      call initialization( lfirst )
end if
```

- Initialization of internal variables
- Only performed the first time the routine is called

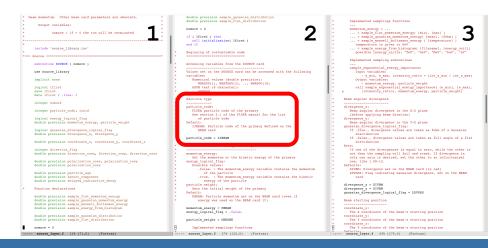




# **Source routine – particle type**

#### particle\_code = IJBEAM

- By default the particle type given in the BEAM card is taken (IJBEAM variable)
- The particle type can be overridden in the source routine
- Possible application: beam made of more than one type particles
- Particle codes explained in Fluka manual section 5.1





### Source routine – particle momentum/energy

momentum\_energy = PBEAM
energy\_logical\_flag = .false.

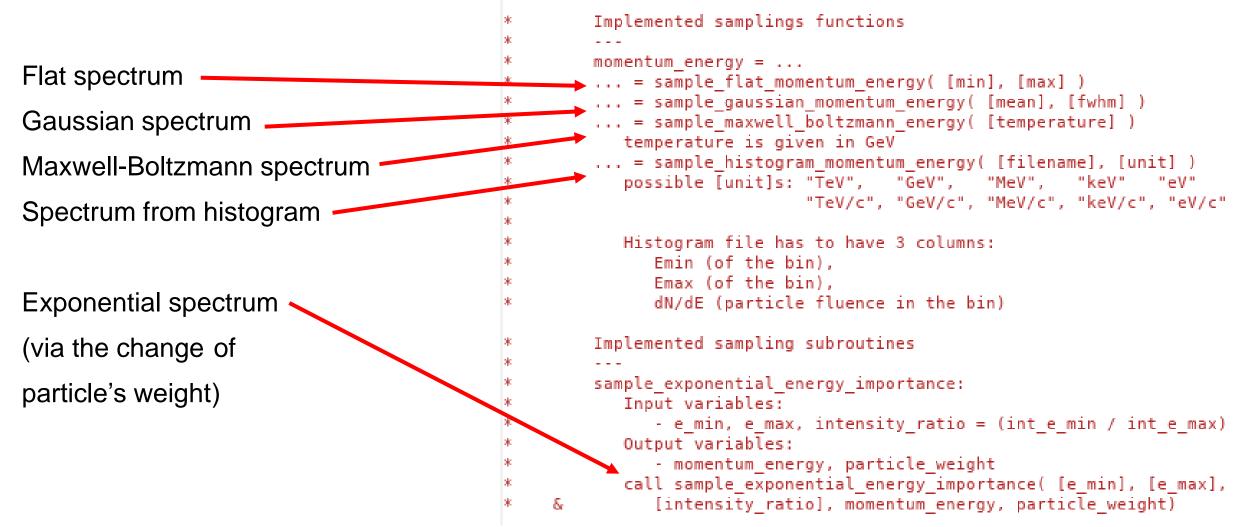
- By default the particle momentum given in the **BEAM** card is taken (**PBEAM** variable)
- **PBEAM** is calculated internally by Fluka
- PBEAM is always the momentum even if energy was provided in the BEAM card
- Energy can be given in the source routine by setting the logical flag as true

beam momentum. Other beam card parameters are obsolete.	double precision sample gaussian distribution double precision sample flat distribution	· Implemented samplings functions
Output variables:	under protector sample_trat_uractioncion	*
nomore = if > 0 the run will be terminated	nomore = 0	<pre>* momentum_energy = * = sample flat momentum_energy( [min], [max] )</pre>
	if (lfirst ) then	<ul> <li> = sample gaussian momentum energy( [mean], [fwhm] )</li> </ul>
	call initialization( lfirat )	* = sample_maxwell_boltzmann_energy ( [temperature] )
include 'source library.inc'	end if	<ul> <li>temperature is given in GeV</li> <li> = sample energy from histogram( [filename], [energy unit])</li> </ul>
Include Source_library.inc.	<ul> <li>Beginning of customizable code</li> </ul>	<ul> <li>sample_energy_iron_miscogram( [rilename], [energy_unic])</li> <li>possible [energy_unit]s: "TeV", "GeV", "MeV", "keV", "eV"</li> </ul>
Source		<ul> <li>Implemented sampling subroutines</li> </ul>
subrostine SOURCE ( nomore )	· Accessing variables from the SOURCE card	implemented sampling subroutines
(	•	sample exponential energy importance;
use source_library	<ul> <li>Values set on the SOURCE card can be accessed with the following</li> </ul>	<ul> <li>Input variables)</li> </ul>
	<ul> <li>variables:</li> </ul>	<ul> <li>e_min, e_max, intensity_ratio = (int_e_min / int_e_max)</li> </ul>
implicit none	<ul> <li>Numerical values (double precision):</li> <li>WHASOU(1), WHATSOU(2),, WHASOU(18)</li> </ul>	<ul> <li>Output variables:</li> <li>momentum energy, particle weight</li> </ul>
logical lfirst	<ul> <li>SDUM text (8 character);</li> </ul>	<ul> <li>call sample exponential energy importance( [e min], [e max].</li> </ul>
save lfirst	<ul> <li>spusou</li> </ul>	<ul> <li>a [intensity ratio], momentum energy, particle weight]</li> </ul>
data lfirst / .true. /		
	<ul> <li>Particle type</li> </ul>	<ul> <li>Beam angular divergence</li> </ul>
integer nomore	· · · · · · · · · · · · · · · · · · ·	*
integer particle code, ionid	<ul> <li>particle_code:</li> <li>FLOWA particle code of the primary</li> </ul>	<ul> <li>divergence_x:</li> <li>Beam angular divergence in the X-2 plane</li> </ul>
runder berever-"compt source	<ul> <li>See section 5.1 of the FLUKA manual for the list</li> </ul>	<ul> <li>(before applying beam diretion)</li> </ul>
logical energy logical flag	<ul> <li>of particle code</li> </ul>	<ul> <li>divergence y:</li> </ul>
double precision momentum energy, particle weight	* Default:	<ul> <li>Beam angular divergence in the Y-2 plane</li> </ul>
	<ul> <li>IJBEAM: Particle code of the primary defined on the</li> </ul>	<ul> <li>gaussian_divergence_logical_flag;</li> </ul>
logical gaussian_divergence_logical_flag double precision divergence_x, divergence_y	• BEAN card	<ul> <li>If .true., divergence values are taken as fwhm of a Gaussian</li> <li>distribution</li> </ul>
double precision coordinate x, coordinate y, coordinate z	particle_code = IJBEAN	<ul> <li>If .false., divergence values are taken as full angle of a flat distribution</li> </ul>
double precision coordinate_x, coordinate_y, coordinate_z	<ul> <li>Partice momentum / energy and weight</li> </ul>	* Note:
integer direction flag	·	<ul> <li>If one of the divergences is equal to zero, while the other is</li> </ul>
double precision direction cosx, direction cosy, direction cosx	<ul> <li>momentum energy:</li> </ul>	not then the sampling will fail and crash. If divergence in
	* Set the momentum or the kinetic energy of the primary	<ul> <li>only one axis is desired, set the other to an infinitesimal</li> </ul>
double precision polarization_cosx, polarization_cosy	energy_logical_flag;	• one, like 1.00-12.
double precision polarization_cosz	<ul> <li>Possible values:         <ul> <li>.false.: The momentum energy variable contains the momentum</li> </ul> </li> </ul>	<ul> <li>Defaults:</li> <li>DIVEN: Divergence set on the BEAN card (in rad)</li> </ul>
double precision particle age	of the particle	<ul> <li>Divisit Divergence set on the BLAN card [in rad]</li> <li>LDVG55: Flag indicating Gaussian divergence, set on the BEAN</li> </ul>
double precision kshort component	<ul> <li>true, : The momentum energy variable contains the kinetic</li> </ul>	<ul> <li>card</li> </ul>
double precision delayed radioactive decay	<ul> <li>energy of the particle</li> </ul>	
	<ul> <li>particle weight:</li> </ul>	divergence_x = DIVBM
Function declarations	<ul> <li>Sets the initial weight of the primary</li> </ul>	divergence_y = DIVBM
double precision sample flat momentum energy	Default:	gaussian_divergence_logical_flag = LEWGES
double precision sample riat momentum energy double precision sample gaussian momentum energy	energy was used on the BEAN CARD LT	<ul> <li>Beam starting position</li> </ul>
double precision sample maxwell_boltzmann_energy		and the second
double precision sample energy from histogram	momentum_energy = PBEAM	* coordinate_x:
	energy_logical_flag = .false.	<ul> <li>The X coordinate of the beam's starting position</li> </ul>
double precision sample_gaussian_distribution double precision sample flat distribution	particle weight = ONEONE	<ul> <li>coordinate_y:</li> <li>The Y coordinate of the beam's starting position</li> </ul>
nonnie hisoreron sambie trat orgehightion	harcrore_werkur - Augoug	<ul> <li>The F coordinate of the beak's starting position</li> <li>coordinate z:</li> </ul>
nomore = 0	Implemented samplings functions	The 2 coordinate of the beam's starting position
		Defaulte.



# Source routine – particle momentum/energy

• Some predefined routines (4 functions and 1 subroutine) are already available





# **Source routine – particle weight**

#### particle\_weight = ONEONE

- Allows to set the weight of the primary particles
- 99% of the times weight=1 is ok
- Can be changed to distort the distribution of primaries (e.g. exponential distribution)
- Can be useful if dealing with more than one single type of primaries
- Not for a beginners' use, mentioned here for completeness

beam momentum. Other beam card parameters are obsolete. *	double precision sample_gaussian_distribution double precision sample_flat_distribution	<ul> <li>Implemented samplings functions</li> </ul>
Output variables:	-	
	nomore = 0	<ul> <li>momentum_energy =</li> </ul>
nomore = if > 0 the run will be terminated *		<ul> <li> = sample flat momentum energy( [min], [max] )</li> </ul>
	<pre>if (lfirst ) then</pre>	<ul> <li> = sample gaussian momentum energy( [mean], [fwhm] )</li> </ul>
	call initialization( lfirat )	= sample maxwell boltzmann energy ( [temperature] )
	end if	<ul> <li>temperature is given in GeV</li> </ul>
include 'source library.inc'		<ul> <li> = sample energy from histogram( [filename], [energy unit])</li> </ul>
incluse boulde_library like	<ul> <li>Beginning of customizable code</li> </ul>	<ul> <li>possible [energy unit]s: "TeV", "GeV", "HeV", "keV", "eV"</li> </ul>
Source	Beginning of customizable code	<ul> <li>bostore (energy_unities) iev, dev, nev, kev, ev</li> </ul>
adure		
	I show the second secon	<ul> <li>Implemented sampling subroutines</li> </ul>
subrostine SOURCE ( nomore )	<ul> <li>Accessing variables from the SOURCE card</li> </ul>	
	•	<ul> <li>sample_exponential_energy_importance;</li> </ul>
use source_library	<ul> <li>Values set on the SOURCE card can be accessed with the following</li> </ul>	<ul> <li>Input variables;</li> </ul>
	<ul> <li>variables:</li> </ul>	<ul> <li>e min, e max, intensity ratio = (int e min / int e max</li> </ul>
implicit none	<ul> <li>Numerical values (double precision):</li> </ul>	<ul> <li>Output variables:</li> </ul>
	<ul> <li>WHABOU(1), WHATBOU(2),, WHABOU(18)</li> </ul>	<ul> <li>momentum energy, particle weight</li> </ul>
logical lfirst	<ul> <li>SDUM text (8 character))</li> </ul>	<ul> <li>call sample exponential energy importance( (e min), (e max)</li> </ul>
save lfirst	• SDUSOU	<ul> <li>* a [intensity ratio], momentum energy, particle weight]</li> </ul>
data lfirst / .true, /		a finesers?_correly momentary parents werded
were access / .com. /	· Particle type	<ul> <li>Beam angular divergence</li> </ul>
	varticle type	beam angular divergence
integer nomore		
	<ul> <li>particle_code;</li> </ul>	<ul> <li>divergence_x;</li> </ul>
integer particle_code, ionid	<ul> <li>FLUEA particle code of the primary</li> </ul>	<ul> <li>Beam angular divergence in the X-2 plane</li> </ul>
	<ul> <li>See section 5.1 of the FLUKA manual for the list</li> </ul>	<ul> <li>(before applying beam diretion)</li> </ul>
logical energy logical flag	<ul> <li>of particle code</li> </ul>	<ul> <li>divergence y:</li> </ul>
double precision momentum energy, particle weight	* Default:	<ul> <li>Beam angular divergence in the Y-2 plane</li> </ul>
	<ul> <li>IJBEAM: Particle code of the primary defined on the</li> </ul>	<ul> <li>gaussian divergence logical flag;</li> </ul>
logical gaussian divergence logical flag	<ul> <li>BEAM card</li> </ul>	<ul> <li>If .true., divergence values are taken as fwhm of a Gaussian</li> </ul>
double precision divergence x, divergence y		* distribution
and here a	particle code = IJBEAM	<ul> <li>If .false., divergence values are taken as full angle of a fla</li> </ul>
double precision coordinate x, coordinate y, coordinate z	hererere cone - round	<ul> <li>If .faise., divergence values are taken as full angle of a file</li> <li>distribution</li> </ul>
double precision coordinate_x, coordinate_y, coordinate_z		* Note:
	<ul> <li>Partice momentum / energy and weight</li> </ul>	
integer direction_flag	· · · · · · · · · · · · · · · · · · ·	<ul> <li>If one of the divergences is equal to zero, while the other is</li> </ul>
double precision direction_cosx, direction_cosy, direction_cosx	<ul> <li>momentum_energy:</li> </ul>	* not then the sampling will fail and crash. If divergence in
	* Set the momentum or the kinetic energy of the primary	<ul> <li>only one axis is desired, set the other to an infinitesimal</li> </ul>
double precision polarization cosx, polarization cosy	<ul> <li>energy logical flag;</li> </ul>	<ul> <li>one, like 1.40-12.</li> </ul>
double precision polarization cosz	<ul> <li>Possible values:</li> </ul>	Defaults:
	<ul> <li>.false. : The momentum energy variable contains the momentum</li> </ul>	<ul> <li>DIVEN: Divergence set on the BEAM card [in rad]</li> </ul>
double precision particle age	· of the particle	<ul> <li>LDVG55: Flag indicating Gaussian divergence, set on the BEAN</li> </ul>
double precision kshort component	<ul> <li>true, : The momentum energy variable contains the kinetic</li> </ul>	<ul> <li>card</li> </ul>
double precision delayed radioactive decay	<ul> <li>energy of the particle</li> </ul>	- CHAM
double precision delayed radioaccive decay		
	<ul> <li>particle weight:</li> </ul>	divergence_x = DIVBM
Function declarations	<ul> <li>Sets the initial weight of the primary</li> </ul>	divergence_y = DIVBM
	* Default:	gaussian_divergence_logical_flag = LOVGSS
double precision sample_flat_momentum_energy		
double precision sample_gaussian_momentum_energy	energy was used on the BEAN card 10	<ul> <li>Beam starting position</li> </ul>
double precision sample maxwell boltzmann energy		· · · · · · · · · · · · · · · · · · ·
double precision sample energy from histogram	momentum energy = 202AN	<ul> <li>coordinate x:</li> </ul>
· · · · · · · · · · · · · · · · · · ·	energy logical flag = .false.	<ul> <li>The X coordinate of the beam's starting position</li> </ul>
double precision sample gaussian distribution		<ul> <li>coordinate vi</li> </ul>
double precision sample flat distribution	particle weight = ONEONE	<ul> <li>The Y coordinate of the beam's starting position</li> </ul>
double precision sample_list_distribution	harcrote_sether - ourous	
		<pre>coordinate_z:</pre>
nomore = 0	Implemented samplings functions	The I coordinate of the beam's starting position
source layer.f 11% (71,0) (Fortran)	-: ROUTCE Layer.1 278 (122,0) (FOFLISS)	-: source layer.f 43% (175,0) (Fortran)



### **Source routine – beam divergence**

- By default, values are taken from the **BEAM** card
- Divergence values are taken
  - As Gaussian FWHM, if flag set .true.
  - As flat distribution full angle, if flag set .false.

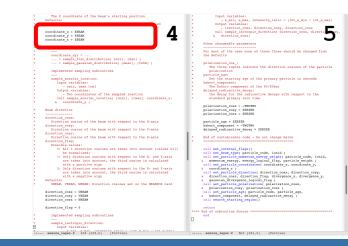
beam momentum. Other beam card parameters are obsolete.	<ul> <li>double precision sample_gaussian_distribution</li> <li>double precision sample flat distribution</li> </ul>	· Implemented samplings functions
Output variables:	undre precision ampre-rate dractioneron	Inplemented subjirings tenetions
output variables.	pomore = 0	momentum energy =
nomore = if > 0 the run will be terminated	nomore = 0	= sample flat momentum energy( [min], [max] )
HOROTE - 11 > 0 the fun will be cerminated	if (lfirst) then	<pre>* = sample gaussian momentum energy( [min], [max] ) * = sample gaussian momentum energy( [mean], [fwhm] )</pre>
	call initialization( lfirst )	<ul> <li> = sample_gaussian_momentum_energy( [mean], [rwnn] )</li> </ul>
		• = sample_maxwell_boltzmann_energy ( [temperature] )
	end if	<ul> <li>temperature is given in GeV</li> </ul>
include 'source_library.inc'		* = sample_energy_from_histogram( [filename], [energy_unit])
	<ul> <li>Beginning of customizable code</li> </ul>	* possible [energy_unit]s: "TeV", "GeV", "HeV", "keV", "eV"
Source	• • • • • • • • • • • • • • • • • • • •	
		<ul> <li>Implemented sampling subroutines</li> </ul>
subroutine SOURCE ( nomore )	<ul> <li>Accessing variables from the SOURCE card</li> </ul>	<ul> <li>A second sec second second sec</li></ul>
	· · · · · · · · · · · · · · · · · · ·	<ul> <li>sample exponential energy importance;</li> </ul>
use source library	<ul> <li>Values set on the SOURCE card can be accessed with the following</li> </ul>	<ul> <li>Input variables;</li> </ul>
abe boarce_ribrary	* variables:	<ul> <li>e min, e max, intensity ratio = (int e min / int e max</li> </ul>
	<ul> <li>Numerical values (double precision):</li> </ul>	
implicit none		<ul> <li>Output variables:</li> </ul>
	<ul> <li>WHABOU(1), WHATBOU(2),, WHABOU(18)</li> </ul>	<ul> <li>momentum_energy, particle_weight</li> </ul>
logical lfirst	<ul> <li>SDUM text (8 character);</li> </ul>	<ul> <li>call sample_exponential_energy_importance( (e_min), (e_max)</li> </ul>
save lfirst	<ul> <li>spusou</li> </ul>	<ul> <li>s (intensity_ratio), momentum_energy, particle_weight)</li> </ul>
data lfirst / .true. /		
	<ul> <li>Particle type</li> </ul>	<ul> <li>Beam angular divergence</li> </ul>
integer momore	· · · · · · · · · · · · · · · · · · ·	· · · · · · · · · · · · · · · · · · ·
	<ul> <li>particle code;</li> </ul>	<ul> <li>divergence xi</li> </ul>
integer particle code, ionid	<ul> <li>FLUXA particle code of the primary</li> </ul>	<ul> <li>Beam angular divergence in the X-2 plane</li> </ul>
	See section 5.1 of the FLUXA manual for the list	<ul> <li>(before applying beam diretion)</li> </ul>
logical energy logical flag	<ul> <li>of particle code</li> </ul>	<ul> <li>divergence v:</li> </ul>
double precision momentum energy, particle weight	* Default:	<ul> <li>Beam angular divergence in the Y-2 plane</li> </ul>
conners breetston momentum energy, buttlete weight	<ul> <li>IJBEAM: Particle code of the primary defined on the</li> </ul>	<ul> <li>gaussian divergence logical flag;</li> </ul>
	<ul> <li>IJELAN: Particle code of the primary defined on the</li> <li>BEAN card</li> </ul>	
logical gaussian_divergence_logical_flag	· BEAR CAIG	<ul> <li>If .true., divergence values are taken as fwhm of a Gaussian</li> </ul>
double precision divergence_x, divergence_y		<ul> <li>distribution</li> </ul>
	particle_code = IJBEAM	<ul> <li>If .false., divergence values are taken as full angle of a fla</li> </ul>
double precision coordinate_x, coordinate_y, coordinate_z		<ul> <li>distribution</li> </ul>
	<ul> <li>Partice momentum / energy and weight</li> </ul>	* Note:
integer direction flag	· · · · · · · · · · · · · · · · · · ·	<ul> <li>If one of the divergences is equal to zero, while the other is</li> </ul>
double precision direction cosx, direction cosy, direction cosx	<ul> <li>momentum energy:</li> </ul>	* not then the sampling will fail and crash. If divergence in
	<ul> <li>Set the momentum or the kinetic energy of the primary</li> </ul>	<ul> <li>only one axis is desired, set the other to an infinitesizal</li> </ul>
double precision polarization cosx, polarization cosy	<ul> <li>energy logical flag;</li> </ul>	<ul> <li>one, like 1.40-12.</li> </ul>
double precision polarization cosz	* Possible values:	* Defaults:
goante hreeteres herertes 0005	<ul> <li>false. : The momentum energy variable contains the momentum</li> </ul>	UTTOTAL COLUMN C
hadde and data and data and	<ul> <li>.false. : the Bomentum energy variable contains the Bomentum</li> <li>of the particle</li> </ul>	LOVGES: Miss indicating Gaussian divergence, set on a BEAN
double precision particle_age		
double precision kshort_component	<ul> <li>true. : The momentum_energy variable contains the kinetic</li> </ul>	card
double precision delayed radioactive decay	<ul> <li>energy of the particle</li> </ul>	
	<ul> <li>particle weight:</li> </ul>	divergence_x = DIVBM
Function declarations	<ul> <li>Sets the initial weight of the primary</li> </ul>	divergence_y = DIVEM
	* Default:	qaussian divergence logical flag = LOVGSS
double precision sample flat momentum energy	<ul> <li>PBEAM: Particle momentum set on the BEAM card (even if</li> </ul>	
double precision sample gaussian momentum energy	<ul> <li>energy was used on the BEAN card it)</li> </ul>	Beam starting position
double precision sample maxwell boltzmann energy		
double precision sample energy from histogram	momentum energy = FDEAN	<ul> <li>coordinate x:</li> </ul>
	energy logical flag = .false.	The X coordinate of the beam's starting position
double precision sample caussian distribution	energy_rodrom_trad - trans.	<ul> <li>The A coordinate of the beam's starting position</li> <li>coordinate y:</li> </ul>
		coordinate_yi
double precision sample_flat_distribution	particle_weight = ONEONE	<ul> <li>The Y coordinate of the beam's starting position</li> </ul>
		<pre>coordinate_z:</pre>
nomore = 0	Implemented samplings functions	The I coordinate of the beam's starting position
source layer.f 11% (71,0) (Fortran)	-: source layer.f 278 (122,0) (Fortran)	-: source layer.f 43% (175,0) (Fortran)



## **Source routine – beam starting position**

coordinate\_x = XBEAM
coordinate\_y = YBEAM
coordinate\_z = ZBEAM

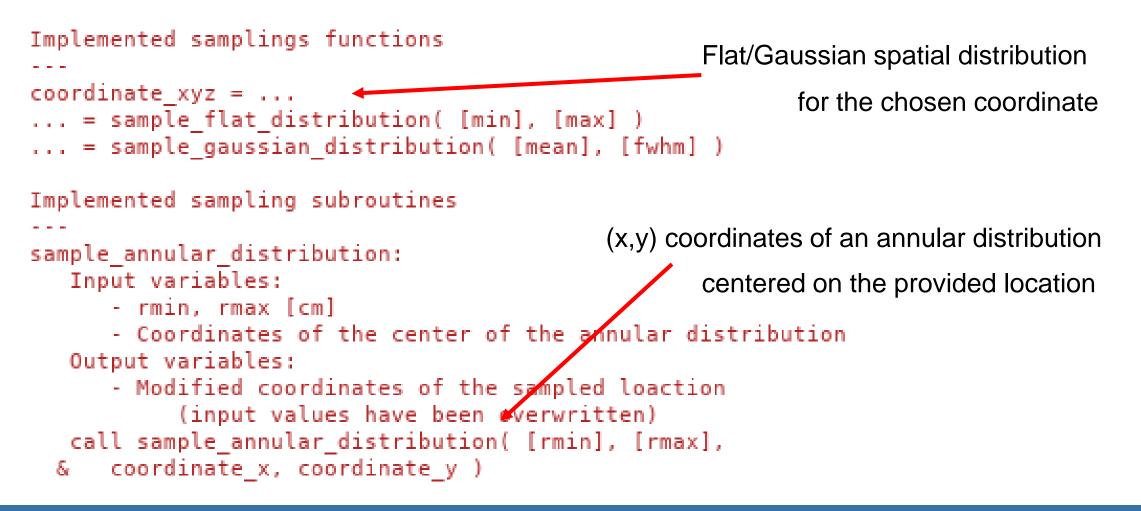
- By default, values are taken from the **BEAMPOS** card
- Extended sources can be defined using different starting positions





# **Source routine – beam starting position**

• Some predefined routines (2 functions and 1 subroutine) are already available

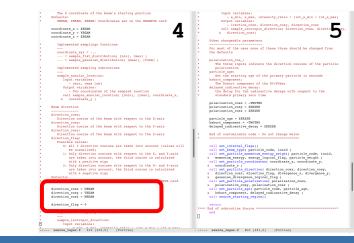




```
direction_cosx = UBEAM
direction_cosy = VBEAM
direction_cosz = WBEAM
direction_flag = 0
```

- By default, values are taken from the **BEAM** card
- If direction\_flag = 0:

all 3 director cosines are considered (normalization is performed in a subroutine)

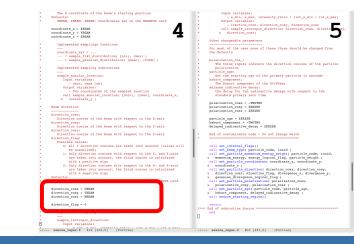




```
direction_cosx = UBEAM
direction_cosy = VBEAM
direction_cosz = WBEAM
direction_flag = 0
```

- By default, values are taken from the **BEAM** card
- If direction\_flag = 1:

direction\_cosz is calculated from the other 2 and assumed positive

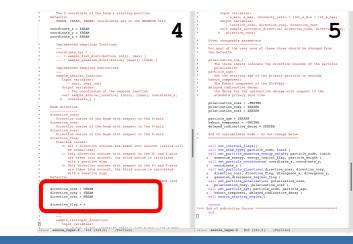




```
direction_cosx = UBEAM
direction_cosy = VBEAM
direction_cosz = WBEAM
direction_flag = 0
```

- By default, values are taken from the **BEAM** card
- If direction\_flag = 2:

direction\_cosz is calculated from the other 2 and assumed negative





• A predefined subroutine is are already available

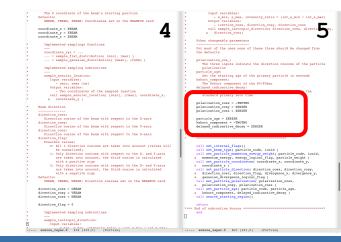
```
Isotropic direction
          Implemented sampling subroutines
)00
)(0
          sample isotropic direction:
寒.
             Output variables:
Э¢С,
                 - direction cosx, direction_cosy, direction_cosz
寒.
             call sample isotropic direction( direction cosx, direction cosy,
寒.
                direction cosz)
DØC.
            δε.
```



### **Source routine – other parameters**

```
polarization_cosx = -TWOTWO
polarization_cosy = ZERZER
polarization_cosz = ZERZER
particle_age = ZERZER
kshort_component = -TWOTWO
delayed_radioactive_decay = ZERZER
```

- Variable names are pretty self-explanatory
- Not for beginners' use

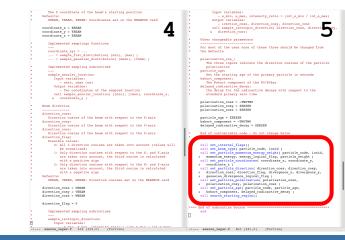




# Source routine – lines not to be touched

- call set\_internal\_flags()
- call set\_beam\_type(...)
- call set\_particle\_momentum\_energy\_weight(...)
- call set\_particle\_coordinates(...)
- call set\_particle\_direction(...)
- call set\_particle\_polarization(...)
- call set\_particle\_age(...)
- call search\_starting\_region()
- These calls pass the provided inputs to Fluka
- Not to be touched for any reason







# Some predefined FLUKA random sampling routines

• Fluka offers some 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



# Compile

- 1. Add the routine
- 2. Verify that it appears
- 3. Insert the name of your executable
- Warning: the library file (source\_library.inc) must be

in the same directory of the source file (**source\_layer.f**)

🤊 🗣 🗧 💭 🖓 Input 💕 Geometry 🐁 Run 🛄 Plot 🛕 Compile Calculator 🔻 Output × Remove 🖁 Cut 🔡 99 ы 5 Move Up Viewer Editor Ifluka Clean Build Paste 🗈 Copy 🔐 Save As Add - Database - 🛛 Move Down Action Clipboard Files View Compile 0 ▲ × 3 Executable: my executable.exe × Options: File Size Date Type source layer.f 2020.09.23 11:36:28 Fortran 10418 Files: 1 **B** 🕺



4. Select the compiler

5. Build your executable

#### **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 can be passed (SDUSOU)
   SOURCE card and BEAM card can coexist
- 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 internally as default for some scoring

SOURCE	#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:



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

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





