SRIMfit
Users Manual

Update Log :

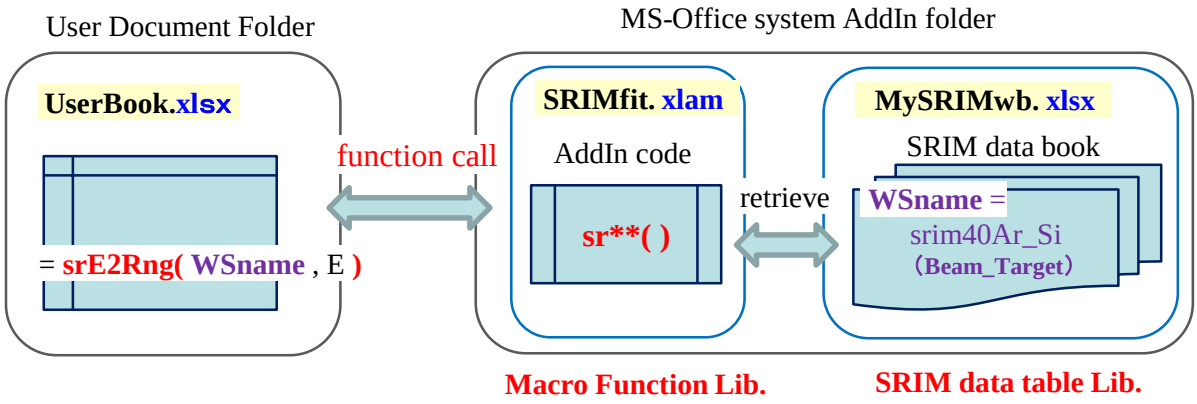
2018.10/22 ver 3.01.03E English edition

SRIMfit Users Manual

Contents

SRIMfit User I/F

SRIMfit is a macro function library for MS-Excel which calculate range, stopping power and energy loss of ions in a matter. The function reads a stopping/range table provided by SRIM-2013 [1] and returns a fitted result at a certain energy.



- "SRIMfit.xlam" is macro-function libraries corded by using Excel Visual Basic.
- "MySRIMwb.xlsx" is stopping/range data tables provided by SRIM-2013 code.
- SRIMfit modules are installed as an Excel "system Add-In macro" function

All SRIMfit functions are named as "sr**( )". For example, the range calculation function named srE2Rng( ) needs two parameters; Energy of the beam and WSnm in order to specify the combination of beam nuclide and target material. The function retrieves the SRIM data book using the specified WSnm, then reads its stopping/range table and calculates range at a specified energy using a simple linear interpolation. Finally, the calculated range-value is returned to user's spread sheet.

- "SRIMfit.xlam" provides about 80 functions.
- "MySRIMwb.xlsx" can include many work-sheets. The name of the work-sheet (WSnm) can be defined as "BeamNuclide_TargetMaterial", for example, "srim40Ar_Si".

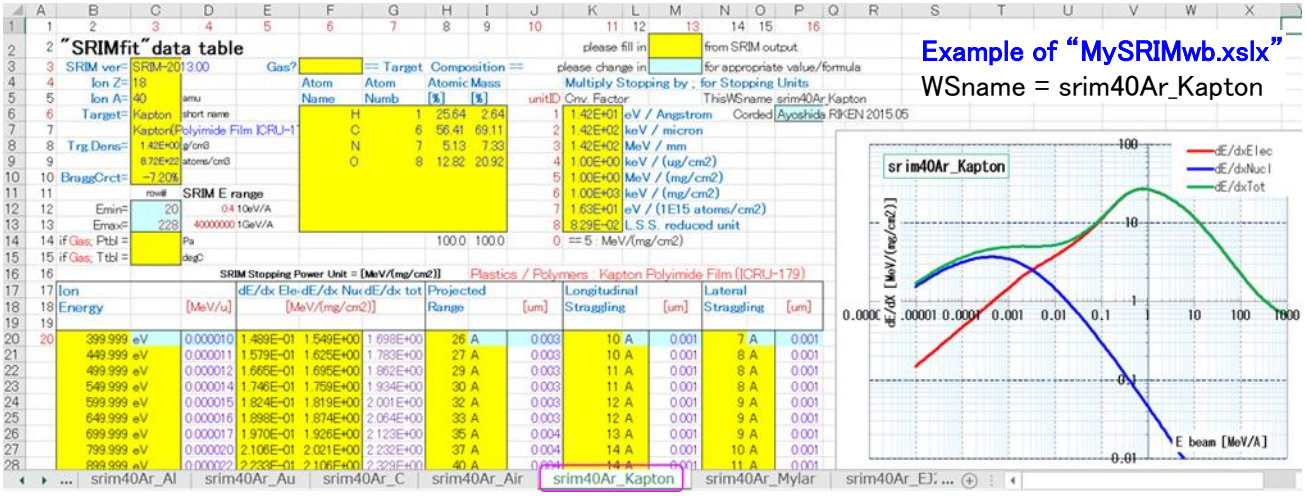
In this manner, one function can calculate many combinations of beam and target, just changing the WSnm parameter.

- Operating environment:
  MS-Excel later than ver. 2003 running on
  MS-Windows OS later than XP or MacOS (*)

(*) MacOS 10.14 + Excel for Mac 2016 has been tested.

Ref.) 1) J.F. Ziegler, SRIM-2013 code home page; http://srim.org/

SRIM-2013 data book (MySRIMwb.xlsx)

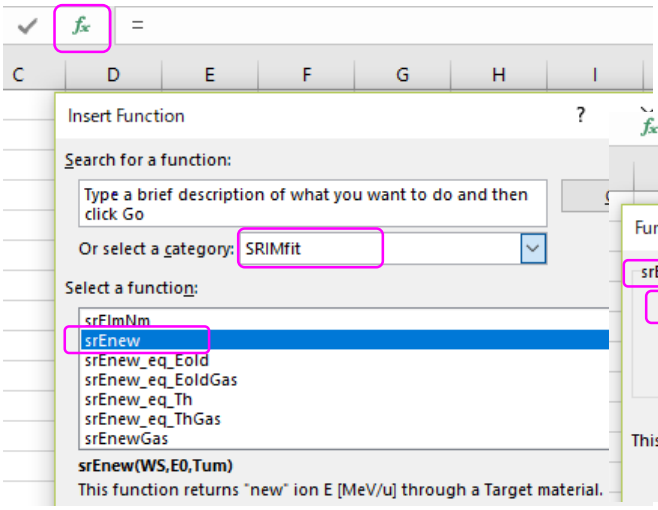


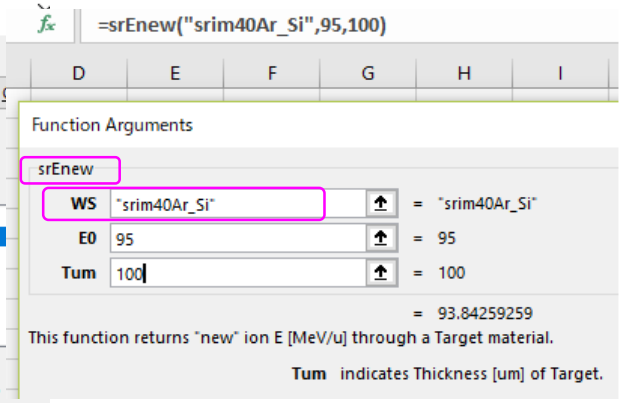
In a work sheet named "srim40Ar_Kapton", the values in yellow cells are all copied from the stopping/range table provided by SRIM code. The format of the work sheet is strictly defined in the macro code. Not only E vs. Range table, but also dE/dX and straggling table and all other information are included in this sheet. That is to say, this one work sheet has object data structure which includes all calculation conditions and results using SRIM code.

- "MySRIMwb.xlsx" data-base file should be prepared manually as users own purpose.
  e.g.) combination of beams (Ar, Kr and Xe) times
  combination of targets (Kapton, Mylar, Air, Plastic scintillator, Al and Si) makes
  18 data sheets should be prepared in advance by using SRIM-2013 code.
- A macro utility "srOut2Ws.xlsm" helps you to convert
  a SRIM-2013 output.txt file to "MySRIMwb.xlsx" work-sheet format.

The nickname "SRIMfit" is named as "macro function library for fitting SRIM-2013 like output". You may be noticed, as long as the format of the data-base sheet is identical, you can use not only the SRIM output but can use any stopping/range table provided by other codes.

Function help message





Function help messages are available.
- Click "fx" insert function icon
- Category = "SRIMfit"

Then you can see a list of SRIMfit functions.

Almost every function requires *WSname* argument to specify a retrieval data sheet.

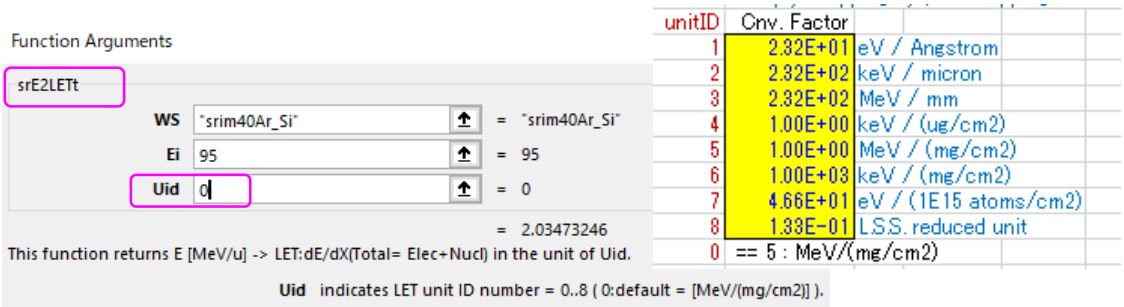
➢ Unit system for function arguments.

- Beam Energy [MeV/u]
- Range, Straggling [μm]
- Thickness [μm] or [mm] for Gas target
- LET (dE/dX) [MeV/(mg/cm2)] as default.

➢ Unit conversion

- [MeV/u] = E [MeV] / srInfoIonA(*WSname*)
- [μm] = srmg2um(*WSname*, *T [mg/cm2]*) , [mg/cm2] = srum2mg(*WSname*, *T [μm]*)

- For the unit of LET (dE/dX), SRIM-2013 supports eight kinds of unit system. The "Uid = 0 .. 8" argument specify the unit system.



e.g.) [keV/μm] = LET [MeV/(mg/cm2)] * srLETCnvF(*WSname*, *Uid=2*)

Function List (1) Fundamental functions

Original list below is included in SRIMfit.xlsm

< Type of Param, Return >		< Variant type function : Error return values >		
I	Integer			
D	Double	#NUM!	xlErrNum 2036	Invalid NUMber as parameter
B	Boolean	#N/A	xlErrNA 2042	return value Not Available
S	String			
V	Variant type			

< Notation for Parameters >			< Private Variables defined in this macro code >		
WS	S	SRIMoutput WorkSheet name	Clm*		Column number in a WS
E, Eu	D	Beam Energy [MeV/u]	Row*		Row number in a WS
Et	D	Beam Energy [MeV]	MySRwbNow		Current WorkBook pointer
R	D	Range [μm]	MySRwsNow		Current WorkSheet pointer
Uid	I	LET unit ID# 0..8	WSnow		Current WorkSheet name
LET	D	defa. [MeV/(mg/cm2)] dep.on Uid		Cur.WS parameters	
Th,Tum,Tmm	D	Thickness [μm] | [mm]	IxE[min|max], E[min|max], R[min|max],		
Pa	D	Gas Pressure [Pa]	St[Lng|Ltr][min|max], SPfct()		
dgC	D	Gas Temperature [degC]			

● mark indicates most useful functions.

Category							
Return type	Func.name				Func. Description		
		params.	type	Param. Description	Return value	Error conditions	Comments
Macro Init / Term procedures							
Sub	**srMySRwb_open**				SRIMoutput WorkBook (MySRwb) Open procedure		
		MyFn	S	MySRwb file name	As default, Const MySRwbFn ="MySRIMwb.xlsx" is used		
					Its Path is MySRwbDir= ThisWorkbook.Path as a default.		
Sub	**srMySRwb_close**				SRIMoutput WorkBook (MySRwb) Close procedure		
Macro informations							
S	**srMcrVer**				SRIMfit version number		
					defined as Const SRIMfitVer		
S	**srMcrPath**				Installed Path of MySRwb		
					returns ThisWorkbook.Path		
S	**srMcrWBname**				File name of current MySRwb		
					returns MySRwbFnNow		
S	**srMcrWBname**				File path of current MySRwb		
					returns MySRwbDirNow		
I	**srMcrWScount**				Num. of Sheets included in MySRwb		
S()	**srMcrWSlist**				returns All-Sheet.names included in MySRwb		
					in 1-dim string array format		
Sub	**srMcr_WSlist**				make a list of All-Sheet.names included in MySRwb		
		sToRngS	S	Top cell position for the list. Eg.) "A10"			
WS independent functions							
S	**srElmNm**				Element symbol		
		Z	I	Atomic number Z= 1..118			

Category							
Return type	Func.name				Func. Description		
	params.	type	Param. Description	Return value	Error conditions	Comments	

WS information(1) : directly read from a current WS

S	**srInfoVer**			SRIM version num.	
	WS	*S*	WSname		
I	**srInfoIonZ**			Beam Z ; Atomic number	
	WS	*S*	WSname		
I	**srInfoIonA**			Beam A ; Mass number	
	WS	*S*	WSname		
S	**srInfoTrgName**			Target name (in short format)	
	WS	*S*	WSname		
S	**srInfoTrgNameL**			Target name (in long format)	
	WS	*S*	WSname		
D	**srInfoTrgDens**			Target density in [g/cm3]	
	WS	*S*	WSname		
D	**srInfoTrgDensA**			Target density in [atoms/cm3]	
	WS	*S*	WSname		
D	**srInfoBrgC**			Bragg Correction in [%]	
	WS	*S*	WSname		
D	**srInfoTrgPtbl**			if GasTarget, a specified gas pressure in [Pa]	
	WS	*S*	WSname	when you calculated this WS using SRIM-2013	
D	**srInfoTrgTtbl**			if GasTarget, a specified gas temperature in [℃]	
	WS	*S*	WSname	when you calculated this WS using SRIM-2013	
B	**srInfoIsGas**			returns 'TRUE' when the WS is for Gas target	
	WS	*S*	WSname		
S	**srInfoTgCmAtmNm**			Target Composition List: Atomic name	
	WS	*S*	WSname		
	i	*I*	Columb number (1..8)		
I	**srInfoTgCmAtmNo**			Target Composition List: Atomic number	
	WS	*S*	WSname		
	i	*I*	Columb number (1..8)		
D	**srInfoTgCmAtmPct**			Target Composition List: Atomic [%]	
	WS	*S*	WSname		
	i	*I*	Columb number (1..8)		
D	**srInfoTgCmMasPct**			Target Composition List: Mass [%]	
	WS	*S*	WSname		
	i	*I*	Columb number (1..8)		
S	**srInfoWScorded**			Corded information of this WS	
	WS	*S*	WSname		

Category							
Return type	Func.name				Func. Description		
	params.	type	Param. Description		Return value	Error conditions	Comments
Unit conversion functions							
V,S	**srLETUNm**				LET Unit Name		
	Uid	I	LET Unit ID (0..8, =0 defa)		#NUM!	Uid<0 | >8	Uid invalid
V,D	**srLETCnvF**				LET Unit Conversion Factor for [Uid=0 MeV/(mg/cm2)]		
	WS	S	WS name		#NUM!	Uid<0 | >8	Uid invalid
	Uid	I	LET Unit ID (0..8, =0 defa)				
D	**srum2mg**				Target Thickness unit conversion [um] --> [mg/cm2]		
	WS	S	WS name			using srInforTrgDens()	
	um	D	Target thickness [μm]				
D	**srmg2um**				Target Thickness unit conversion [mg/cm2] --> [um]		
	WS	S	WS name			using srInforTrgDens()	
	mg	D	Target thickness [mg/cm2]				
Work Sheet Info (2) Table min/max							
D	**srMinE**				min. of Beam E table [MeV/u]		
	WS	S	WS name				
D	**srMaxE**				max. of Beam E table [MeV/u]		
	WS	S	WS name				
D	**srMinRng**				min. of Range table [μm]		
	WS	S	WS name				
D	**srMaxRng**				max. of Range table [μm]		
	WS	S	WS name				
D	**srMinStLng**				min. of Straggling Longitudinal table [μm]		
	WS	S	WS name				
D	**srMaxStLng**				max. of Straggling Longitudinal table [μm]		
	WS	S	WS name				
D	**srMinStLtr**				min. of Straggling Lateral table [μm]		
	WS	S	WS name				
D	**srMaxStLtr**				max. of Straggling Lateral table [μm]		
	WS	S	WS name				

Category							
Return type	*Func.name*				*Func. Description*		
	params.	*type*	*Param. Description*		*Return value*	*Error conditions*	*Comments*
Work Sheet Look up (1) E <-> LETe,n,t							
V,D	**srE2LETe**				Look up E -> LETe [in Uid unit] ; Electric Stopping Power		
	WS	*S*	WS name		#NUM!	Uid<0 | >8	Uid invalid
	E	*D*	Beam E [MeV/u] as look-up key		#N/A		E is out of range in the WS
	Uid	*I*	LET Unit ID (0..8) for the return value				
V,D	**srE2LETn**				Look up E -> LETn [in Uid unit] ; Nuclear Stopping Power		
	WS	*S*	WS name		#NUM!	Uid<0 | >8	Uid invalid
	E	*D*	Beam E [MeV/u] as look-up key		#N/A		E is out of range in the WS
	Uid	*I*	LET Unit ID (0..8) for the return value				
V,D	**srE2LETt**				Look up E -> LETt [in Uid unit] ;Total= Nuclear+Electric Stopping Powe		
	WS	*S*	WS name		#NUM!	Uid<0 | >8	Uid invalid
	E	*D*	Beam E [MeV/u] as look-up key		#N/A		E is out of range in the WS
	Uid	*I*	LET Unit ID (0..8) for the return value				

LET subscription { e | n | t } indicates
{ electronic | nuclear | total = electric + nuclear } Stopping Power

V,D	**srLETe2E**				Look up LETe -> E [MeV/u]		
	WS	*S*	WS name		#NUM!	Uid<0 | >8	Uid invalid
	Lt	*D*	LETe [Uid] as look-up key		#N/A		LETe is out of range in the WS
	Uid	*I*	LET Unit ID (0..8) for the look-up key				
	Ehl	*I*	={+1|-1} = E-search from {Ehigh|Elow} side				
V,D	**srLETn2E**				Look up LETn -> E [MeV/u]		
	WS	*S*	WS name		#NUM!	Uid<0 | >8	Uid invalid
	Lt	*D*	LETn [Uid] as look-up key		#N/A		LETn is out of range in the WS
	Uid	*I*	LET Unit ID (0..8) for the look-up key				
	Ehl	*I*	={+1|-1} = E-search from {Ehigh|Elow} side				
V,D	**srLETt2E**				Look up LETt -> E [MeV/u]		
	WS	*S*	WS name		#NUM!	Uid<0 | >8	Uid invalid
	Lt	*D*	LETt [Uid] as look-up key		#N/A		LETt is out of range in the WS
	Uid	*I*	LET Unit ID (0..8) for the look-up key				
	Ehl	*I*	={+1|-1} = E-search from {Ehigh|Elow} side				

Category						
Return type	*Func.name*			*Func. Description*		
	params.	*type*	*Param. Description*	*Return value*	*Error conditions*	*Comments*

Work Sheet Look up (1) E <-> LETe,n,t

V,D	**srMaxLETe**		max. of LETe [Uid] table	
WS	*S*	WS name	#NUM!	Uid<0 | >8	Uid invalid
Uid	*I*	LET Unit ID (0..8) for the return value			
V,D	**srMaxLETn**		max. of LETn [Uid] table	
WS	*S*	WS name	#NUM!	Uid<0 | >8	Uid invalid
Uid	*I*	LET Unit ID (0..8) for the return value			
V,D	**srMaxLETt**		max. of LETt [Uid] table	
WS	*S*	WS name	#NUM!	Uid<0 | >8	Uid invalid
Uid	*I*	LET Unit ID (0..8) for the return value			
D	**srMaxLETe2E**		E [MeV/u] at max LETe	
WS	*S*	WS name			
D	**srMaxLETn2E**		E [MeV/u] at max LETn	
WS	*S*	WS name			
D	**srMaxLETt2E**		E [MeV/u] at max LETt	
WS	*S*	WS name			

Work Sheet Look up (2) E <-> Range

V,D	**srE2Rng**		Look up E -> Range [μm]	
WS	*S*	WS name	#NUM!	E<0	
E	*D*	Beam E [MeV/u] as look-up key	#N/A		E is out of range (>Emax)
			=0	E=0	
V,D	**srRng2E**		Look up Range -> E [MeV/u]	
WS	*S*	WS name	#NUM!	Rng<0	
Rng	*D*	Range [μm] as look-up key	#N/A		Rng is out of range (>Rmax)
			=0	Rng=0	

These two functions are important functions and are often used in the following combination functions.

Category						
Return type	*Func.name*			*Func. Description*		
	params.	*type*	*Param. Description*	*Return value*	*Error conditions*	*Comments*
Work Sheet Look up (3) E <-> Straggling						
V,D	**srE2StLng**			Look up E -> Straggling Longitudinal [μm]		
	WS	*S*	WS name	#NUM!	E<0	
	E	*D*	Beam E [MeV/u] as look-up key	#N/A		E is out of range (>Emax)
				=0	E=0	
V,D	**srE2StLtr**			Look up E -> Straggling Lateral [μm]		
	WS	*S*	WS name	#NUM!	E<0	
	E	*D*	Beam E [MeV/u] as look-up key	#N/A		E is out of range (>Emax)
				=0	E=0	
V,D	**srStLng2E**			Look up Straggling Longitudinal -> E [MeV/u]		
	WS	*S*	WS name	#NUM!	Strg<0	
	Strg	*D*	Strag. Long. [μm] as look-up key	#N/A		Strg is out of range (>StLngmax)
				=0	Strg=0	
V,D	**srStLtr2E**			Look up Straggling Lateral -> E [MeV/u]		
	WS	*S*	WS name	#NUM!	Strg<0	
	Strg	*D*	Strag. Later. [μm] as look-up key	#N/A		Strg is out of range (>StLtrmax)
				=0	Strg=0	

Gas Target : "Standard" Pressure & Temperature						
"Standard"is at P=srInfoTrgPtbl() and at T=srInfoTrgTtbl() where the WS is calculated by SRIM-2013						
V,D	**srThkStd**			Conversion coeff. for the standard P & T		
	WS	*S*	WS name for Gas Target	#NUM!	WS<>Gas	WS is not for Gas Target
	Pa	*D*	Gas Pressure [Pa]	#NUM!	Pa<0 Pa | dgC<0 K	
	dgC	*D*	Gas Temperature [℃]	= (Pa/P0)*(273.15+T0)/(273.15+dgC) <-- This is the coeff.		

Concerning Gas-Target work sheet

Gas functions calculate a range thickness in the gas as following formula ;

Thick [μm] @(P, T) = srE2Rng(WS, E) / srThkStd(WS, P, T)

Category							
Return type	*Func.name*				*Func. Description*		
	params.	*type*	*Param. Description*		*Return value*	*Error conditions*	*Comments*
for Debugging public functions							
S	**srIxEmin**				get IxEmin		
	WS	*S*	WS name				
S	**srIxEmax**				get IxEmax		
	WS	*S*	WS name				
V	**srE2Ix**				E -> ixE search using sr_E2Ix()		
	WS	*S*	WS name		#N/A	ts_errE	
	Ei	*D*	E [MeV/u]				
V	**srVal2Ix**				Val -> ixE search using sr_Val2Ix()		
	WS	*S*	WS name		#N/A	ts_errV	
	Vc	*D*	Value				
	Clm	*I*	Clm# = 1..7 ; E, SPe, SPn, SPt, Rng, StLng, StLtr				
V	**srIx2Val**				(ixE, Clm#) -> Value		
	WS	*S*	WS name		#N/A		ixE | Cim invalid
	ixE	*I*	ixE				
	Clm	*I*	Clm# = 1..7 ; E, SPe, SPn, SPt, Rng, StLng, StLtr				
V,D()	**srIx2ValAry**				ixE -> Value(Clm=1..7) Double Array		
	WS	*S*	WS name		#N/A		ixE invalid
	ixE	*I*	ixE				

SRIMfit Function List (2a) Combination functions

< Notation for Parameters >		
WS1,WS2	S	WSname
Eu	D	Beam Energy [MeV/u]
Et	D	Beam Energy [MeV]
dEu	D	Beam Energy Loss [MeV/u]
dEt	D	Beam Energy Loss [MeV]
Tum,Tmm	D	Thickness [μm] | [mm]
Pa	D	Gas Pressure [Pa]
dgC	D	Gas Temperature [degC]

< reason why _eq_() function returns '#N/A' error >		
case-1)	Eu10 > Emax	out of E table in WS1
case-2)	Eu11 < 0	Eu20, Th2 becomes indefinite
case-3)	Et11 > Et20	
case-5)	dEt1 > Et20	
case-8)		out of Rng(E) table in WS1
case-9)		out of Rng(E) table in WS2
case-11)		dE peak search error
case-12)	dEu2max < dEu1	
case-14)		dE2 ixE search error

Category							
Return type	*Func.name*				*Func. Description*		
		params.	*type*	*Param. Description*	*Return value*	*Error conditions*	*Comments*

Combination Func. (2a) equivalent E <-> Rng

V,D	**srEnew**			Beam E [MeV/u] AFTER passing through Th[μm] of the target		
	WS	*S*	WSname for Eu10 & Th1	#NUM!	Eu10<=0	
	Eu10	*D*	E [MeV/u] before passing Th1	#N/A	-Th >Rmax	
	Th1	*D*	Target Thickness [μm]	=0	Eu10=0 | Th1>=Rng(Eu10)	
				=Eu10	Th1=0	
				=srEold(Eu11,Th1)	Th1<0	same as srEold()

As function	As calculation
parameter	assumption
return	look-up key

	before	*through*	*after*
	Eold	Thick	Enew
	MeV/u	μm	MeV/u
WS1	Eu10	Th1	Eu11
prm.	prm.	prm.	ret.

(Equivalent Equations)

```
srEnew(WS,Eu10,Th1) {
    R10= srE2Rng(WS,Eu10)
    if((R11= R10 - Th1)<=0) return( 0 )
    E11u= srRng2E(WS,R11)
    return( E11u )
}
```

V,D	**srEold**			Beam E [MeV/u] BEFORE passing through Th[μm] of the target	
	WS	*S*	WSname for Eu11 & Th1	#NUM!	Eu11<=0 | Th1<0
	Eu11	*D*	E [MeV/u] after passing Th1	#N/A	E1>=Emax
	Th1	*D*	Target Thickness [μm]	=Eu11	Th1=0
				=srRng2E(Th1)	Th1=0 & E11=0

	before	*through*	*after*
	Eold	Thick	Enew
	MeV/u	μm	MeV/u
WS1	Eu10	Th1	Eu11
prm.	ret.	prm.	prm.

```
srEold(WS,Eu11,Th1) {
    R11= srE2Rng(WS,Eu11)
    R10= R11 + Th1
    Eu10= srRng2E(WS,R10)
    return( Eu10 )
}
```

As function	As calculation
parameter	assumption
return	look-up key

Category						
Return type	*Func.name*			*Func. Description*		
	params.	*type*	*Param. Description*	*Return value*	*Error conditions*	*Comments*
V,D	**srRng_eq_Eold**			Range equivalent Eold [MeV/u]		
	WS1	*S*	WSname for Eu10	#NUM!	Eu10<=0	
	WS2	*S*	WSname for return Eold	#N/A	case-1), -9)	
	Eu10	*D*	E [MeV/u] before passing through			

	before	*gives*
	Eold	Range
	MeV/u	μm
WS1	Eu10	Rng1
prm.	prm.	
WS2	Eu20	Rng2
prm.	ret.	=Rng1

```
srRng_eq_Eold(WS1,WS2,Eu10) {
    R10= srE2Rng(WS1,Eu10)
    R20=R10
    Eu20= srRng2E(WS2,R20)
    return( Eu20 )
}
```

V,D	**srEnew_eq_Th**			Enew [MeV/u] equivalent thickness [μm]		
	WS1	*S*	WSname for Eu10 & Th1	#NUM!	Eu10<=0 | Th1<0	
	WS2	*S*	WSname for return Thick	#N/A	Th1>=Rng(Eu10)	case-2) Eu20 becomes indefinite
	Eu10	*D*	E [MeV/u] before passing Th1	#N/A	case-1), -3), -8), -9)	
	Th1	*D*	Target Thickness [μm]	=0	Th1=0	as Eu20=Eu10

	before	*through*	*after*	*E total*
	Eold	Thick	Enew	Enewt
	MeV/u	μm	MeV/u	MeV
WS1	Eu10	Th1	Eu11	Et11
prm.	prm.	prm.		
WS2	Eu20	Th2	Eu21	Et21
prm.	=Eu10	ret.	=Eu11	

```
srEnew_eq_Th(WS1,WS2,Eu10,Th1) {
    Eu11= srEnew(WS1,Eu10,Th1)
    if(Eu11 <= 0) return( #N/A )
    Eu20= Eu10
    R20= srE2Rng(WS2,Eu20)
    Eu21= Eu11
    R21= srE2Rng(WS2,Eu21)
    Th2= R20 - R21
    return( Th2 )
}
```

V,D	**srEnewt_eq_Th**			Enewt [MeV] equivalent thickness [μm]		
	WS1	*S*	WSname for Eu10 & Th1	#NUM!	Eu10<=0 | Th1<0	
	WS2	*S*	WSname for return Thick	#N/A	Th1>=Rng(Eu10)	case-2) Eu20 becomes indefinite
	Eu10	*D*	E [MeV/u] before passing Th1	#N/A	case-1), -3), -8), -9)	
	Th1	*D*	Target Thickness [μm]	=0	Th1=0	as Eu20=Eu10

	before	*through*	*after*	*E total*
WS1	Eu10	Th1	Eu11	Et11
prm.	prm.	prm.		
WS2	Eu20	Th2	Eu21	Et21
prm.	=Eu10	ret.		=Et11

```
srEnewt_eq_Th(WS1,WS2,Eu10,Th1) {
    A1= srInfoIonA(WS1)
    A2= srInfoIonA(WS2)
    Eu11= srEnew(WS1,Eu10,Th1)
    Et11= Eu11*A1
    Eu20= Eu10
    if(Et11 > Eu20*A2) return( #N/A )
    R20= srE2Rng(WS2,Eu20)
    Et21= Et11
    Eu21= Et21/A2
    R21= srE2Rng(WS2,Eu21)
    Th2= R20 - R21
    return( Th2 )
}
```

As function	As calculation
parameter	assumption
return	look-up key

Category							
Return type	*Func.name*				*Func. Description*		
		params.	*type*	*Param. Description*	*Return value*	*Error conditions*	*Comments*
V,D	**srEnew_eq_Eold**				Enew [MeV/u] equivalent Eold [MeV/u]		
	WS1		*S*	WSname for Eu10 & Th1	#NUM!	Eu10<=0 | Th1<0	
	WS2		*S*	WSname for return Eold	#N/A	Th1>=Rng(Eu10)	case-2) Eu20 becomes indefinite
	Eu10		*D*	E [MeV/u] before passing Th1	#N/A	case-1), -8), -9)	
	Th1		*D*	Target Thickness [μm]	=Eu11	Th1=0	as Eu20=Eu21=Eu11=Eu10

	before	*through*	*after*	*E total*
	Eold	Thick	Enew	Enewt
	MeV/u	μm	MeV/u	MeV
WS1	Eu10	Th1	Eu11	Et11
prm.	prm.	prm.		
WS2	Eu20	Th2	Eu21	Et21
prm.	ret.	=Th1	=Eu11	

```
srEnew_eq_Eold(WS1,WS2,Eu10,Th1) {
    Eu11= srEnew(WS1,Eu10,Th1)
    if(Eu11 <= 0) return( #N/A )
    Th2= Th1
    Eu21= Eu11
    if(Eu21==0) Eu20= srRng2E(WS2,Th2)
    else        Eu20= srEold(WS2,Eu21,Th2)
    return( Eu20 )
}
```

V,D	**srEnewt_eq_Eold**				Enewt [MeV/u] equivalent Eold [MeV/u]		
	WS1		*S*	WSname for Eu10 & Th1	#NUM!	Eu10<=0 | Th1<0	
	WS2		*S*	WSname for return Eold	#N/A	Th1>=Rng(Eu10)	case-2) Eu20 becomes indefinite
	Eu10		*D*	E [MeV/u] before passing Th1	#N/A	case-1), -8), -9)	
	Th1		*D*	Target Thickness [μm]	=Eu11	Th1=0	as Eu20=Eu21=Eu11=Eu10

	before	*through*	*after*	*E total*
WS1	Eu10	Th1	Eu11	Et11
prm.	prm.	prm.		
WS2	Eu20	Th2	Eu21	Et21
prm.	ret.	=Th1		=Et11

```
srEnewt_eq_Eold(WS1,WS2,Eu10,Th1) {
    A1= srInfoIonA(WS1)
    A2= srInfoIonA(WS2)
    Eu11= srEnew(WS1,Eu10,Th1)
    Et11= Eu11*A1
    Th2= Th1
    Et21= Et11
    if(Et21==0) Eu20= srRng2E(WS2,Th2)
    else        Eu20= srEold(WS2,Eu21,Th2)
    return( Eu20 )
}
```

As function	As calculation
parameter	assumption
return	look-up key

Category							
Return type	*Func.name*				*Func. Description*		
		params.	*type*	*Param. Description*	*Return value*	*Error conditions*	*Comments*
V,D	**srDEu_eq_Th**			is equivalent to srEnew_eq_Th()	dEu [MeV/u]	equivalent Thickness [μm]	
	WS1		*S*	WSname for Eu10 & Th1	#NUM!	Eu10<=0 | Th1<0	
	WS2		*S*	WSname for return Thick	#N/A	Th1>=Rng(Eu10)	case-2) Eu20 becomes indefinite
	Eu10		*D*	E [MeV/u] before passing Th1	#N/A	case-1), -3), -8), -9)	
	Th1		*D*	Target Thickness [μm]	=0	Th1=0	as Eu20=Eu10

	before	*through*	*after*	*E total*	*E loss*	*E loss*
	Eold	Thick	Enew	Enewt	dEu	dEt
	MeV/u	μm	MeV/u	MeV	MeV/u	MeV
WS1	Eu10	Th1	Eu11	Et11	dEu1	dEt1
prm.	prm.	prm.				
WS2	Eu20	Th2	Eu21	Et21	dEu2	dEt2
prm.	=Eu10	ret.	=Eu11		=dEu1	

```
srDEu_eq_Th(WS1,WS2,Eu10,Th1) {
    Th2= srEnew_eq_Th(WS1,WS2,Eu10,Th1)
    return( Th2 )
}
```

∵ Eu20=Eu10

∴ dEu1=(Eu10-Eu11)=dEu2=(Eu10-Eu21)

∴ Eu21=Eu11

∴ equivalent to srEnew_eq_Th()

V,D	**srDEt_eq_Th**				dEt [MeV]	equivalent Thickness [μm]	
	WS1		*S*	WSname for Eu10 & Th1	#NUM!	Eu10<=0 | Th1<0	
	WS2		*S*	WSname for return Thick	#N/A	Th1>=Rng(Eu10)	case-2) Eu20 becomes indefinite
	Eu10		*D*	E [MeV/u] before passing Th1	#N/A	case-1), -5), -8), -9)	
	Th1		*D*	Target Thickness [μm]	=Eu11	Th1=0	as Eu20=Eu10

	before	*through*	*after*	*E total*	*E loss*	*E loss*
WS1	Eu10	Th1	Eu11	Et11	dEu1	dEt1
prm.	prm.	prm.				
WS2	Eu20	Th2	Eu21	Et21	dEu2	dEt2
prm.	=Eu10	ret.				=dEt1

```
srDEt_eq_Th(WS1,WS2,Eu10,Th1) {
    A1= srInfoIonA(WS1)
    A2= srInfoIonA(WS2)
    Eu11= srEnew(WS1,Eu10,Th1)
    dEt1= (Eu10-Eu11)*A1
    Eu20= Eu10
    R20= srE2Rng(WS2,Eu20)
    dEt2= dEt1
    dEu2= dEt2/A2
    Eu21= Eu20 - dEu2
    R21= srE2Rng(WS2,Eu21)
    Th2= R20 - R21
    return( Th2 )
}
```

As function	As calculation
parameter	assumption
return	look-up key

Category							
Return type	*Func.name*				*Func. Description*		
		params.	*type*	*Param. Description*	*Return value*	*Error conditions*	*Comments*
V,D	**srDEu_eq_Eold**				dEu [MeV/u]	equivalent Eold [MeV/u]	
	WS1	*S*	WSname for Eu10 & Th1		#NUM!	Eu10<=0 | Th1<0	
	WS2	*S*	WSname for return Eold			Th1>=Rng(Eu10)	is acceptable
	Eu10	*D*	E [MeV/u] before passing Th1		#N/A	case-1), -8), -11), -12), -14)	
	Th1	*D*	Target Thickness [μm]		note)return value(Eu20) has error of a few [%]		
	Ehl	*I*	={+1|-1} = E-search from {Ehigh|Elow} side		because this is a linear interpolation between E vs. dE not between E vs. Range		

	before	*through*	*after*	*E total*	*E loss*	*E loss*
	Eold	Thick	Enew	Enewt	dEu	dEt
	MeV/u	μm	MeV/u	MeV	MeV/u	MeV
WS1	Eu10	Th1	Eu11	Et11	dEu1	dEt1
prm.	prm.	prm.				
WS2	Eu20	Th2	Eu21	Et21	dEu2	dEt2
prm.	ret.	=Th1			=dEu1	

V,D	**srDEt_eq_Eold**				dEt [MeV/u]	equivalent Eold [MeV/u]	
	WS1	*S*	WSname for Eu10 & Th1		#NUM!	Eu10<=0 | Th1<0	
	WS2	*S*	WSname for return Eold			Th1>=Rng(Eu10)	is acceptable
	Eu10	*D*	E [MeV/u] before passing Th1		#N/A	case-1), -8), -11), -12), -14)	
	Th1	*D*	Target Thickness [μm]		note)return value(Eu20) has error of a few [%]		
	Ehl	*I*	={+1|-1} = E-search from {Ehigh|Elow} side		because this is a linear interpolation between E vs. dE not between E vs. Range		

	before	*through*	*after*	*E total*	*E loss*	*E loss*
	Eold	Thick	Enew	Enewt	dEu	dEt
	MeV/u	μm	MeV/u	MeV	MeV/u	MeV
WS1	Eu10	Th1	Eu11	Et11	dEu1	dEt1
prm.	prm.	prm.				
WS2	Eu20	Th2	Eu21	Et21	dEu2	dEt2
prm.	ret.	=Th1				=dEt1

```
srDEu_eq_Eold(WS1,WS2,Eu10,Th1) {
    Eu11= srEnew(WS1,Eu10,Th1)
    dEu1= Eu10-Eu11
    Th2= Th1
    Eu20stop= srRng2E(WS2,Th2)
    ~ sr_dEfpk(-1) : find dEu2 peak from Emax side ~
    ~ -> then get dEu2pk & EdEu2pk ~
    if( dEu2pk > Eu20stop) {
            dEu2max= dEu2pk; EdEu2max= EdEu2pk
    }else{  dEu2max= EdEu2max= Eu20stop }
    if(dEu2max < dEu1) return( N/A ) : No Answer
    if(dEu2max = dEu1)  : Only One Answer
            return( Eu20= EdEu2max )
    dEu2= dEu1
    ~ sr_dE2ixE() : find dEu2  for all Eu20 ~
    ~ calculating Eu20 -> dEu2= Eu20 - Enew(Eu20,Th2) ~
    ~ sr_dE2ip() : interpolation at found dEu2  ~
    ~ using the calculated E vs dE=E-Enew() table ~
    Eu20 = interpolated value at found dEu2
    return( Eu20 )
}
```

```
srDEt_eq_Eold(WS1,WS2,Eu10,Th1) {
    A1= srInfoIonA(WS1)
    A2= srInfoIonA(WS2)
    Eu11= srEnew(WS1,Eu10,Th1)
    dEt1= (Eu10-Eu11)*A1
    Th2= Th1
    Eu20stop= srRng2E(WS2,Th2)
    ~ sr_dEfpk(-1) : find dEu2 peak from Emax side ~
    ~ -> then get dEu2pk & EdEu2pk ~
    if( dEu2pk > Eu20stop) {
            dEu2max= dEu2pk; EdEu2max= EdEu2pk
    }else{  dEu2max= EdEu2max= Eu20stop }
    dEt2max= dEu2max*A2
    if(dEt2max < dEt1) return( N/A ) : No Answer
    if(dEt2max = dEt1)  : Only One Answer
            return( Eu20= EdEu2max )
    dEt2= dEt1
    dEu2= dEt2 / A2
    ~ sr_dE2ixE() : find dEu2  for all Eu20 ~
    ~ calculating Eu20 -> dEu2= Eu20 - Enew(Eu20,Th2) ~
    ~ sr_dE2ip() : interpolation at found dEu2  ~
    ~ using the calculated E vs dE=E-Enew() table ~
    Eu20 = interpolated value at found dEu2
    return( Et20 = Eu20*A2 )
}
```

SRIMfit Function List (2b) Combination functions for Gas Target

< Notation for Parameters >		
Tmm	*D*	Thickness **in [mm] unit**
Pa	*D*	Gas Pressure [Pa]
dgC	*D*	Gas Temperature [℃ degC]
other param.s are same as _eq_() func.		

< reason why _eq_Gas() function returns '#N/A' error >		
#NUM!	WS<>Gas	WS is not for Gas Target
other reasons are same as _eq_() func.		

As function	As calculation
parameter	assumption
return	look-up key

Concerning Gas-Target combination functions

- Gas thickness is in the unit of [mm]
- Gas pressure [Pa] and temperature [℃] need to specified.

Others are same as solid-target functions.

Category							
Return type	*Func.name*				*Func. Description*		
		params.	*type*	*Param. Description*	*Return value*	*Error conditions*	*Comments*

Combination Func. (2b) equivalent E <-> Rng

V,D	**srEnewGas**			Beam E [MeV/u] AFTER passing through Th[**mm**] of the Gas target		
	WS	*S*	WSnameGas for Eu10～dgC1	#NUM!	WS<>Gas	WS is not for Gas Target
	Eu10	*D*	E [MeV/u] before passing Th1	#NUM!	Eu10<=0	
	Tmm1	*D*	Gas Target Thickness **[mm]**	#N/A	-Th >Rmax	
	Pa1	*D*	Gas Trg Pressure [Pa]	=0	Eu10=0 | Th1>=Rng(Eu10)	
	dgC1	*D*	Gas Trg Temperature [℃]	=Eu10	Th1=0	
			=srEold(Eu11,Th1)	Th1<0		same as srEold()

	before	*through*			*after*
	Eold	Thick			Enew
	MeV/u	mm	Pa	℃	MeV/u
WS1	Eu10	Tmm1	Pa1	dgC1	Eu11
prm.	prm.	prm.	prm.	prm.	ret.

```
srEnewGas(WS,Eu10,Tmm1,Pa1,dgC1) {
    Th1= Tmm1 * 1000 * srThkStd(WS,Pa,dgC)
    R10= srE2Rng(WS,Eu10)
    if((R11= R10 - Th1)<=0) return( 0 )
    E11u= srRng2E(WS,R11)
    return( E11u )
}
```

V,D	**srEoldGas**			Beam E [MeV/u] BEFORE passing through Th[**mm**] of the Gas target		
	WS	*S*	WSnameGas for Eu10～dgC1	#NUM!	WS<>Gas	WS is not for Gas Target
	Eu11	*D*	E [MeV/u] after passing Th1	#NUM!	Eu11<=0 | Th1<0	
	Tmm1	*D*	Gas Target Thickness **[mm]**	#N/A	E1>=Emax	
	Pa1	*D*	Gas Trg Pressure [Pa]	=Eu11	Th1=0	
	dgC1	*D*	Gas Trg Temperature [℃]			
			=srRng2E(Th1*srThkStd(Pa1,dgC1))	Th1=0 & E11=0		

	before	*through*			*after*
	Eold	Thick			Enew
	MeV/u	mm	Pa	℃	MeV/u
WS1	Eu10	Th1	Pa1	dgC1	Eu11
prm.	ret.	prm.	prm.	prm.	prm.

```
srEoldGas(WS,Eu10,Tmm1,Pa1,dgC1) {
    Th1= Tmm1 * 1000 * srThkStd(WS,Pa,dgC)
    R11= srE2Rng(WS,Eu11)
    R10= R11 + Th1
    Eu10= srRng2E(WS,R10)
    return( Eu10 )
}
```

As function	As calculation
parameter	assumption
return	look-up key

Category							
Return type	*Func.name*				*Func. Description*		
		params.	*type*	*Param. Description*	*Return value*	*Error conditions*	*Comments*

V,D **srRng_eq_EoldGas() is not implemented.** Please use srRng_eq_Eold().

V,D **srEnew_eq_ThGas** — Enew [MeV/u] equivalent thickness [mm]

WS1	*S*	WSnameGas for Eu10～dgC1	#NUM!	WS1,WS2<>Gas	WS1,2 are not for Gas Target
WS2	*S*	WSnameGas for Pa2～dgC2	#NUM!	Eu10<=0 | Th1<0	
Eu10	*D*	E [MeV/u] after passing Th1	#N/A	Th1>=Rng(Eu10)	case-2) Eu20 becomes indefinite
Tmm1	*D*	Gas Trg1 Thickness [mm]	#N/A	case-1), -3), -8), -9)	
Pa1	*D*	Gas Trg1 Pressure [Pa]	=0	Th1=0	as Eu20=Eu10
dgC1	*D*	Gas Trg1 Temperature [℃]			
Pa2	*D*	Gas Trg2 Pressure [Pa]			
dgC2	*D*	Gas Trg2 Temperature [℃]			

	before	*through*			*after*	*E total*
	Eold	Thick			Enew	Enewt
	MeV/u	mm	Pa	℃	MeV/u	MeV
WS1	Eu10	Th1	Pa1	dgC1	Eu11	Et11
prm.	prm.	prm.	prm.	prm.		
WS2	Eu20	Th2	Pa2	dgC2	Eu21	Et21
prm.	=Eu10	ret.	prm.	prm.	=Eu11	

V,D **srEnewt_eq_ThGas** — Enewt [MeV] equivalent thickness [mm]

WS1	*S*	WSnameGas for Eu10～dgC1	#NUM!	WS1,WS2<>Gas	WS1,2 are not for Gas Target
WS2	*S*	WSnameGas for Pa2～dgC2	#NUM!	Eu10<=0 | Th1<0	
Eu10	*D*	E [MeV/u] after passing Th1	#N/A	Th1>=Rng(Eu10)	case-2) Eu20 becomes indefinite
Tmm1	*D*	Gas Trg1 Thickness [mm]	#N/A	case-1), -3), -8), -9)	
Pa1	*D*	Gas Trg1 Pressure [Pa]	=0	Th1=0	as Eu20=Eu10
dgC1	*D*	Gas Trg1 Temperature [℃]			
Pa2	*D*	Gas Trg2 Pressure [Pa]			
dgC2	*D*	Gas Trg2 Temperature [℃]			

	before	*through*			*after*	*E total*
WS1	Eu10	Th1	Pa1	dgC1	Eu11	Et11
prm.	prm.	prm.	prm.	prm.		
WS2	Eu20	Th2	Pa2	dgC2	Eu21	Et21
prm.	=Eu10	ret.	prm.	prm.		=Et11

```
srEnew_eq_ThGas(WS1,WS2,Eu10,Tmm1,
        Pa1,dgC1,Pa2,dgC2) {
  Th1= Tmm1 * 1000 * srThkStd(WS1,Pa1,dgC1)
  Eu11= srEnew(WS1,Eu10,Th1)
  if(Eu11 <= 0) return( #N/A )
  Eu20= Eu10
  R20= srE2Rng(WS2,Eu20)
  Eu21= Eu11
  R21= srE2Rng(WS2,Eu21)
  Th2= R20 - R21
  Tmm2= Th2 / (1000 * srThkStd(WS2,Pa2,dgC2))
  return( Tmm2 )
}
```

```
srEnewt_eq_ThGas(WS1,WS2,Eu10,Tmm1,
        Pa1,dgC1,Pa2,dgC2) {
  A1= srInfoIonA(WS1)
  A2= srInfoIonA(WS2)
  Th1= Tmm1 * 1000 * srThkStd(WS1,Pa1,dgC1)
  Eu11= srEnew(WS1,Eu10,Th1)
  Et11= Eu11*A1
  Eu20= Eu10
  if(Et11 > Eu20*A2) return( #N/A )
  R20= srE2Rng(WS2,Eu20)
  Et21= Et11
  Eu21= Et21/A2
  R21= srE2Rng(WS2,Eu21)
  Th2= R20 - R21
  Tmm2= Th2 / (1000 * srThkStd(WS2,Pa2,dgC2))
  return( Tmm2 )
}
```

As function	As calculation
parameter	assumption
return	look-up key

Category							
Return type	*Func.name*				*Func. Description*		
		params.	*type*	*Param. Description*	*Return value*	*Error conditions*	*Comments*
V,D	**srEnew_eq_EoldGas**				Enew [MeV/u] equivalent Eold [MeV/u]		
		WS1	*S*	WSnameGas for Eu10～dgC1	#NUM!	WS1,WS2<>Gas	WS1,2 are not for Gas Target
		WS2	*S*	WSnameGas for Pa2～dgC2	#NUM!	Eu10<=0 | Th1<0	
		Eu10	*D*	E [MeV/u] after passing Th1	#N/A	Th1>=Rng(Eu10)	case-2) Eu20 becomes indefinite
		Tmm1	*D*	Gas Trg1 Thickness **[mm]**	#N/A	case-1), -8), -9)	
		Pa1	*D*	Gas Trg1 Pressure [Pa]	=Eu11	Th1=0	as Eu20=Eu21=Eu11=Eu10
		dgC1	*D*	Gas Trg1 Temperature [℃]			
		Pa2	*D*	Gas Trg2 Pressure [Pa]			
		dgC2	*D*	Gas Trg2 Temperature [℃]			

	before	*through*			*after*	*E total*
	Eold	Thick			Enew	Enewt
	MeV/u	mm	Pa	℃	MeV/u	MeV
WS1	Eu10	Th1	Pa1	dgC1	Eu11	Et11
prm.	prm.	prm.	prm.	prm.		
WS2	Eu20	Th2	Pa2	dgC2	Eu21	Et21
prm.	ret.	=Th1	prm.	prm.	=Eu11	

V,D	**srEnewt_eq_EoldGas**				Enewt [MeV/u] equivalent Eold [MeV/u]		
		WS1	*S*	WSnameGas for Eu10～dgC1	#NUM!	WS1,WS2<>Gas	WS1,2 are not for Gas Target
		WS2	*S*	WSnameGas for Pa2～dgC2	#NUM!	Eu10<=0 | Th1<0	
		Eu10	*D*	E [MeV/u] after passing Th1	#N/A	Th1>=Rng(Eu10)	case-2) Eu20 becomes indefinite
		Tmm1	*D*	Gas Trg1 Thickness **[mm]**	#N/A	case-1), -8), -9)	
		Pa1	*D*	Gas Trg1 Pressure [Pa]	=Eu11	Th1=0	as Eu20=Eu21=Eu11=Eu10
		dgC1	*D*	Gas Trg1 Temperature [℃]			
		Pa2	*D*	Gas Trg2 Pressure [Pa]			
		dgC2	*D*	Gas Trg2 Temperature [℃]			

	before	*through*			*after*	*E total*
WS1	Eu10	Th1	Pa1	dgC1	Eu11	Et11
prm.	prm.	prm.	prm.	prm.		
WS2	Eu20	Th2	Pa2	dgC2	Eu21	Et21
prm.	ret.	=Th1	prm.	prm.		=Et11

```
srEnew_eq_EoldGas(WS1,WS2,Eu10,Tmm1,
        Pa1,dgC1,Pa2,dgC2) {
  Th1= Tmm1 * 1000 * srThkStd(WS1,Pa1,dgC1)
  Eu11= srEnew(WS1,Eu10,Th1)
  if(Eu11 <= 0) return( #N/A )
  Th2= Tmm1* 1000 * srThkStd(WS2,Pa2,dgC2)
  Eu21= Eu11
  if(Eu21==0) Eu20= srRng2E(WS2,Th2)
  else        Eu20= srEold(WS2,Eu21,Th2)
  return( Eu20 )
}
```

```
srEnewt_eq_EoldGas(WS1,WS2,Eu10,Tmm1,
        Pa1,dgC1,Pa2,dgC2) {
  A1= srInfoIonA(WS1)
  A2= srInfoIonA(WS2)
  Th1= Tmm1 * 1000 * srThkStd(WS1,Pa1,dgC1)
  Eu11= srEnew(WS1,Eu10,Th1)
  Et11= Eu11*A1
  Th2= Tmm1* 1000 * srThkStd(WS2,Pa2,dgC2)
  Et21= Et11
  if(Et21==0) Eu20= srRng2E(WS2,Th2)
  else        Eu20= srEold(WS2,Eu21,Th2)
  return( Eu20 )
}
```

As function	As calculation
parameter	assumption
return	look-up key

Category							
Return type	*Func.name*				*Func. Description*		
		params.	*type*	*Param. Description*	*Return value*	*Error conditions*	*Comments*
V,D	**srDEu_eq_ThGas**			is equivalent to srEnew_eq_ThGas()	dEu [MeV/u]	equivalent Thickness [mm]	
		WS1	*S*	WSnameGas for Eu10～dgC1	#NUM!	WS1,WS2<>Gas	WS1,2 are not for Gas Target
		WS2	*S*	WSnameGas for Pa2～dgC2	#NUM!	Eu10<=0 | Th1<0	
		Eu10	*D*	E [MeV/u] after passing Th1	#N/A	Th1>=Rng(Eu10)	case-2) Eu20 becomes indefinite
		Tmm1	*D*	Gas Trg1 Thickness [mm]	#N/A	case-1), -3), -8), -9)	
		Pa1	*D*	Gas Trg1 Pressure [Pa]	=0	Th1=0	as Eu20=Eu10
		dgC1	*D*	Gas Trg1 Temperature [℃]			
		Pa2	*D*	Gas Trg2 Pressure [Pa]			
		dgC2	*D*	Gas Trg2 Temperature [℃]			

	before	*through*			*after*	*E total*	*E loss*	*E loss*
	Eold	Thick			Enew	Enewt	dEu	dEt
	MeV/u	mm	Pa	℃	MeV/u	MeV	MeV/u	MeV
WS1	Eu10	Th1	Pa1	dgC1	Eu11	Et11	dEu1	dEt1
prm.	prm.	prm.	prm.	prm.				
WS2	Eu20	Th2	Pa2	dgC2	Eu21	Et21	dEu2	dEt2
prm.	=Eu10	ret.	prm.	prm.	=Eu11		=dEu1	

V,D	**srDEt_eq_ThGas**				dEt [MeV]	equivalent Thickness [mm]	
		WS1	*S*	WSnameGas for Eu10～dgC1	#NUM!	WS1,WS2<>Gas	WS1,2 are not for Gas Target
		WS2	*S*	WSnameGas for Pa2～dgC2	#NUM!	Eu10<=0 | Th1<0	
		Eu10	*D*	E [MeV/u] after passing Th1	#N/A	Th1>=Rng(Eu10)	case-2) Eu20 becomes indefinite
		Tmm1	*D*	Gas Trg1 Thickness [mm]	#N/A	case-1), -5), -8), -9)	
		Pa1	*D*	Gas Trg1 Pressure [Pa]	=Eu11	Th1=0	as Eu20=Eu10
		dgC1	*D*	Gas Trg1 Temperature [℃]			
		Pa2	*D*	Gas Trg2 Pressure [Pa]			
		dgC2	*D*	Gas Trg2 Temperature [℃]			

	before	*through*			*after*	*E total*	*E loss*	*E loss*
WS1	Eu10	Th1	Pa1	dgC1	Eu11	Et11	dEu1	dEt1
prm.	prm.	prm.	prm.	prm.				
WS2	Eu20	Th2	Pa2	dgC2	Eu21	Et21	dEu2	dEt2
prm.	=Eu10	ret.	prm.	prm.				=dEt1

```
srDEu_eq_ThGas(WS1,WS2,Eu10,Tmm1,
        Pa1,dgC1,Pa2,dgC2) {
    Th2= srEnew_eq_ThGas(WS1,WS2,Eu10,Tmm1,
        Pa1,dgC1,Pa2,dgC2) {
    return( Th2 )
}
    ∵ Eu20=Eu10
    ∴ dEu1=(Eu10-Eu11)=dEu2=(Eu10-Eu21)
    ∴ Eu21=Eu11
    ∴ equivalent to srEnew_eq_Th()
```

```
srDEt_eq_ThGas(WS1,WS2,Eu10,Tmm1,
        Pa1,dgC1,Pa2,dgC2) {
    A1= srInfoIonA(WS1)
    A2= srInfoIonA(WS2)
    Th1= Tmm1 * 1000 * srThkStd(WS1,Pa1,dgC1)
    Eu11= srEnew(WS1,Eu10,Th1)
    dEt1= (Eu10-Eu11)*A1
    Eu20= Eu10
    R20= srE2Rng(WS2,Eu20)
    dEt2= dEt1
    dEu2= dEt2/A2
    Eu21= Eu20 - dEu2
    R21= srE2Rng(WS2,Eu21)
    Th2= R20 - R21
    Tmm2= Th2 / (1000 * srThkStd(WS2,Pa2,dgC2))
    return( Tmm2 )
}
```

As function	As calculation
parameter	assumption
return	look-up key

Category							
Return type	*Func.name*				*Func. Description*		
		params.	*type*	*Param. Description*	*Return value*	*Error conditions*	*Comments*

V,D	**srDEu_eq_EoldGas**			dEu [MeV/u] equivalent Eold [MeV/u]		
	WS1	*S*	WSnameGas for Eu10～dgC1	#NUM!	WS1,WS2<>Gas	WS1,2 are not for Gas Target
	WS2	*S*	WSnameGas for Pa2～dgC2	#NUM!	Eu10<=0 | Th1<0	
	Eu10	*D*	E [MeV/u] after passing Th1		Th1>=Rng(Eu10)	is acceptable
	Tmm1	*D*	Gas Trg1 Thickness **[mm]**	#N/A	case-1), -8), -11), -12), -14)	
	Ehl	*I*	={+1|-1} = E-search from {Ehigh|Elow} side	note)return value(Eu20) has error of a few [%]		
	Pa1	*D*	Gas Trg1 Pressure [Pa]	because this is a linear interpolation between E vs. dE		
	dgC1	*D*	Gas Trg1 Temperature [℃]	not between E vs. Range		
	Pa2	*D*	Gas Trg2 Pressure [Pa]			
	dgC2	*D*	Gas Trg2 Temperature [℃]			

	before	*through*			*after*	*E total*	*E loss*	*E loss*
	Eold	Thick			Enew	Enewt	dEu	dEt
	MeV/u	mm	Pa	℃	MeV/u	MeV	MeV/u	MeV
WS1	Eu10	Th1	Pa1	dgC1	Eu11	Et11	dEu1	dEt1
prm.	prm.	prm.	prm.	prm.				
WS2	Eu20	Th2	Pa2	dgC2	Eu21	Et21	dEu2	dEt2
prm.	ret.	=Th1	prm.	prm.			=dEu1	

V,D	**srDEt_eq_EoldGas**			dEt [MeV/u] equivalent Eold [MeV/u]		
	WS1	*S*	WSnameGas for Eu10～dgC1	#NUM!	WS1,WS2<>Gas	WS1,2 are not for Gas Target
	WS2	*S*	WSnameGas for Pa2～dgC2	#NUM!	Eu10<=0 | Th1<0	
	Eu10	*D*	E [MeV/u] after passing Th1		Th1>=Rng(Eu10)	is acceptable
	Tmm1	*D*	Gas Trg1 Thickness **[mm]**	#N/A	case-1), -8), -11), -12), -14)	
	Ehl	*I*	={+1|-1} = E-search from {Ehigh|Elow} side	note)return value(Eu20) has error of a few [%]		
	Pa1	*D*	Gas Trg1 Pressure [Pa]			
	dgC1	*D*	Gas Trg1 Temperature [℃]			
	Pa2	*D*	Gas Trg2 Pressure [Pa]			
	dgC2	*D*	Gas Trg2 Temperature [℃]			

	before	*through*			*after*	*E total*	*E loss*	*E loss*
	Eold	Thick			Enew	Enewt	dEu	dEt
	MeV/u	mm	Pa	℃	MeV/u	MeV	MeV/u	MeV
WS1	Eu10	Th1	Pa1	dgC1	Eu11	Et11	dEu1	dEt1
prm.	prm.	prm.	prm.	prm.				
WS2	Eu20	Th2	Pa2	dgC2	Eu21	Et21	dEu2	dEt2
prm.	ret.	=Th1	prm.	prm.				=dEt1

```
srDEu_eq_EoldGas(WS1,WS2,Eu10,Tmm1,
        Pa1,dgC1,Pa2,dgC2) {
    Th1= Tmm1 * 1000 * srThkStd(WS1,Pa1,dgC1)
    Eu11= srEnew(WS1,Eu10,Th1)
    dEu1= Eu10-Eu11
    Th2= Tmm1* 1000 * srThkStd(WS2,Pa2,dgC2)
    Eu20stop= srRng2E(WS2,Th2)
    ~ sr_dEfpk(-1) : find dEu2 peak from Emax side ~
    ~ -> then get dEu2pk & EdEu2pk ~
    if( dEu2pk > Eu20stop) {
        dEu2max= dEu2pk; EdEu2max= EdEu2pk
    }else{  dEu2max= EdEu2max= Eu20stop }
    if(dEu2max < dEu1) return( N/A ) : No Answer
    if(dEu2max = dEu1)  : Only One Answer
        return( Eu20= EdEu2max )
    dEu2= dEu1
    ~ sr_dE2ixE() : find dEu2  for all Eu20 ~
    ~ calculating Eu20 -> dEu2= Eu20 - Enew(Eu20,Th2) ~
    ~ sr_dE2ip() : interpolation at found dEu2  ~
    ~ using the calculated E vs dE=E-Enew() table ~
    Eu20 = interpolated value at found dEu2
    return( Eu20 )
}
```

```
srDEt_eq_EoldGas(WS1,WS2,Eu10,Tmm1,
        Pa1,dgC1,Pa2,dgC2) {
    A1= srInfoIonA(WS1)
    A2= srInfoIonA(WS2)
    Th1= Tmm1 * 1000 * srThkStd(WS1,Pa1,dgC1)
    Eu11= srEnew(WS1,Eu10,Th1)
    dEt1= (Eu10-Eu11)*A1
    Th2= Tmm1* 1000 * srThkStd(WS2,Pa2,dgC2)
    Eu20stop= srRng2E(WS2,Th2)
    ~ sr_dEfpk(-1) : find dEu2 peak from Emax side ~
    ~ -> then get dEu2pk & EdEu2pk ~
    if( dEu2pk > Eu20stop) {
        dEu2max= dEu2pk; EdEu2max= EdEu2pk
    }else{  dEu2max= EdEu2max= Eu20stop }
    dEt2max= dEu2max*A2
    if(dEt2max < dEt1) return( N/A ) : No Answer
    if(dEt2max = dEt1)  : Only One Answer
        return( Eu20= EdEu2max )
    dEt2= dEt1
    dEu2= dEt2 / A2
    ~ sr_dE2ixE() : find dEu2  for all Eu20 ~
    ~ calculating Eu20 -> dEu2= Eu20 - Enew(Eu20,Th2) ~
    ~ sr_dE2ip() : interpolation at found dEu2  ~
    ~ using the calculated E vs dE=E-Enew() table ~
    Eu20 = interpolated value at found dEu2
    return( Et20 = Eu20*A2 )
}
```

How to prepare MySRIMwb

➢ "MySRIMwb.xlsx" data-base file should be prepared manually as your own purpose.

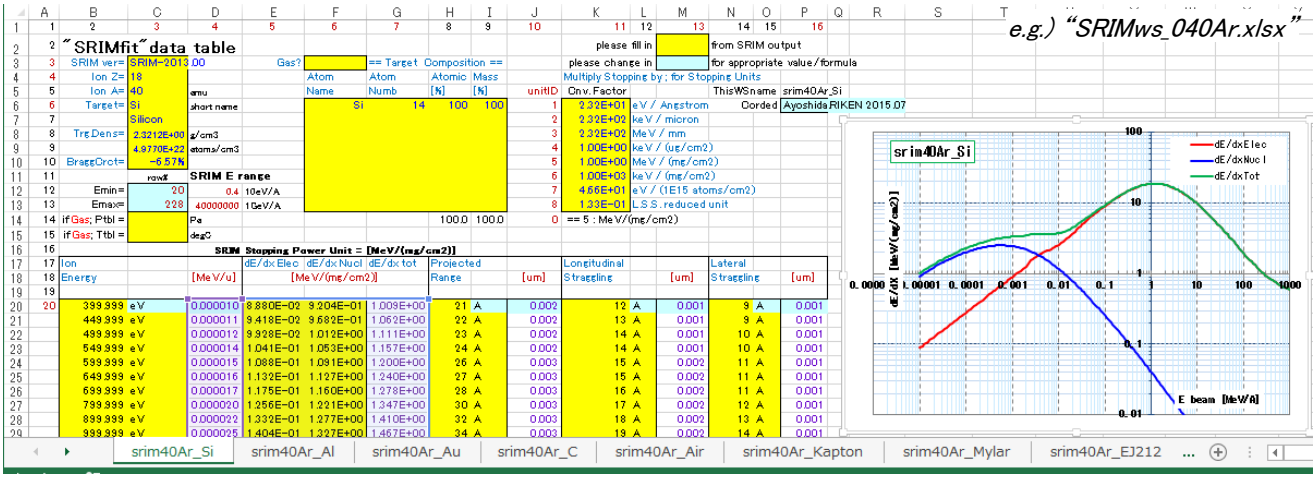
- The installer includes a sample data-base file.
- Other samples are available in SRIMfit download page.
- Or you can make your own data-base file as following instruction.

➢ How to prepare your own MySRIMwb

Step1) Make a stopping/range table (say "SRIMoutput.txt") using SRIM-2013 code.
Step2) Convert the table file to MySRIMwb formatted work sheet using "srOut2WsE.xlsm" utility.
Step3) Pick and choose from the work sheets and arrange your own MySRIMwb.

➢ Instructions for MySRIMwb

e.g.) "SRIMws_040Ar.xlsx"



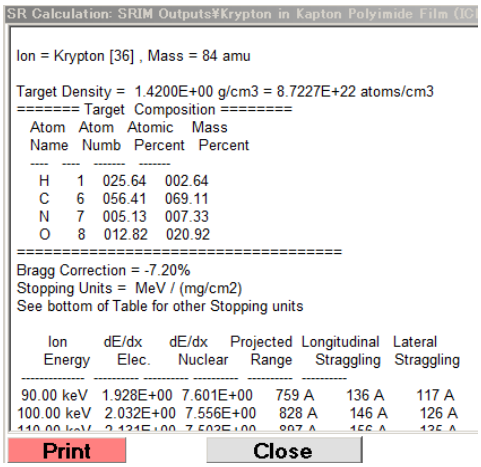
- Please do NOT change the row & column number in a work sheet.
- Yellow Cells : fill in values copied from SRIM-2013 output file.
- Some of the Boxed Cells contains formula.
- Other cells are available for writing any memo note.

- How many sheets are needed in the MySRIMwb.xlsx book file ?
  (Number of WS needed) = (Num. of Beams) x (Num. of Target) for your purpose.
- Does the WSname has some restriction ?
  No, but it is better to use the following notation so that it would be easy to understand.
  e.g.) "srim40Ar_Kapton" = "srim" + "Beam" + "_" + "Target"
- Each WSname should be identical in the book file.
  There is no restriction for the number of sheet in the book. (See Excel system restriction)
  Order of sheets is not important as the SRIMfit functions refer the sheet as WSname.

[Step1] SRIM-2013 calculation

Run the SRIM-2013 code and click “Stopping / Rage Tables”

- Ion Mass “A [amu]” is better to use an “integer number” for the isotope.
- Ion Energy Range [keV] is recommended as ; eg.) A=84
  Lowest : 10 eV/u * 84 = 0.84 keV , Highest : 1 GeV/u * 84 = 84,000,000 keV
  Small Lowest value is important for a precise calculation near beam stopping.
  Highest value should be enough larger than the Bragg Peak energy.
  Energy steps are automatically determined by SRIM-2013 code.



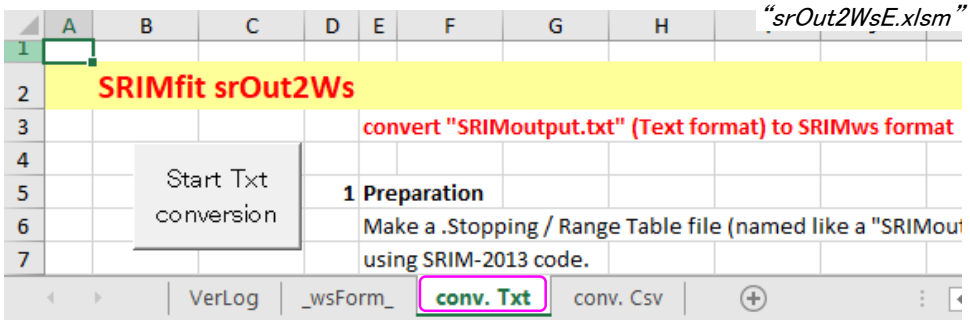
- Target Description ; Compound target material is possible.
  If the target is a Gas material,
  check “Gas Tgt.” and specify an appropriate density at a certain temperature and pressure.

- Stopping power unit MUST BE “MeV/(mg/cm2)”.

- Click “Calculate Table” then a text file named as “Target Description” box is created in the ¥SRIM Outputs folder.

[Step2-a] convert “SRIMoutput.txt” to SRIMwb format

[1] Open “srOut2WsE.xlsm” and show “conv.Txt” sheet.

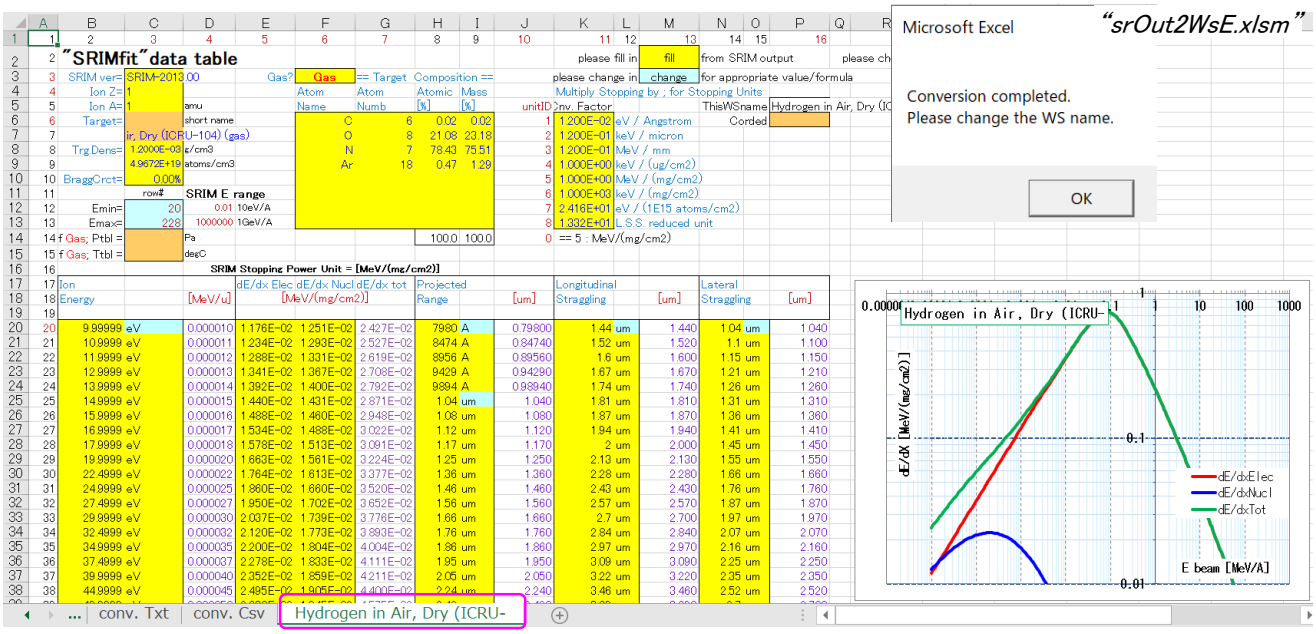


Note) Please do not change the contents of “_wsForm_” sheet, because the macro refers it as a prototype format for the new WS.

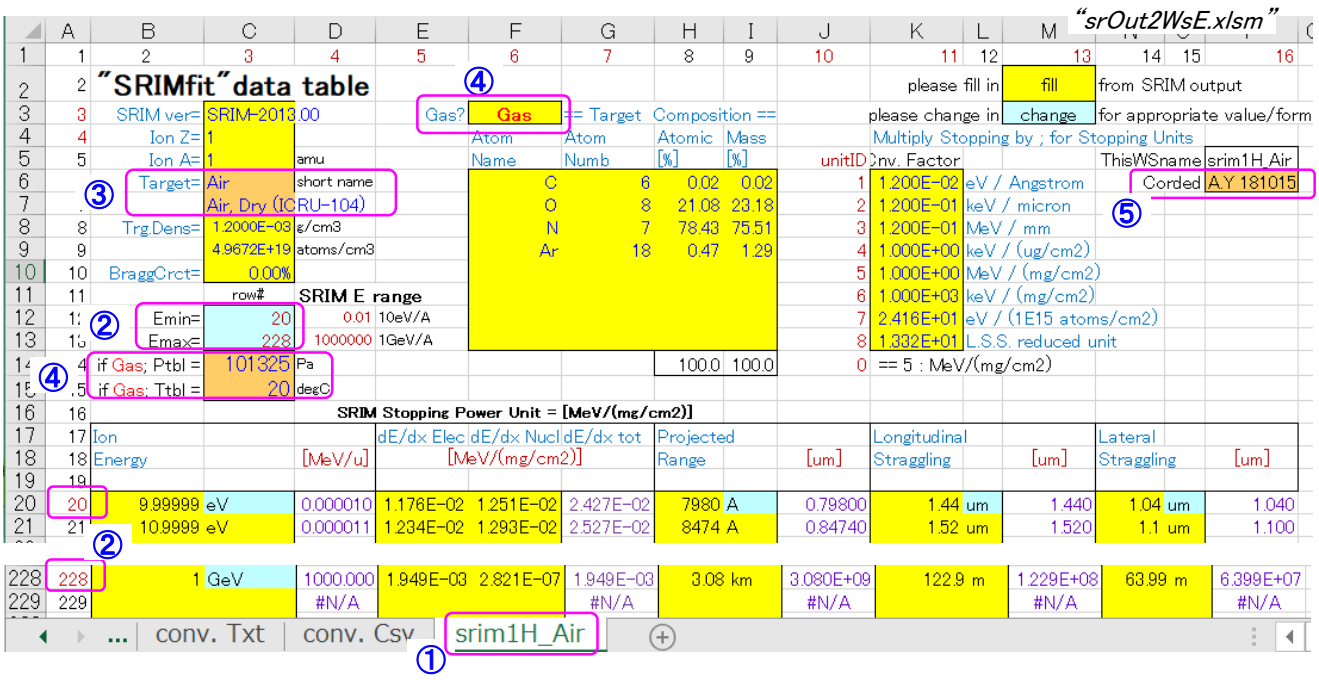
[2] Push “Start Txt conversion” button and Select the previously made “SRIMoutput.txt” file.

Some sample files for the “SRIMoutput.txt” are included in this folder.

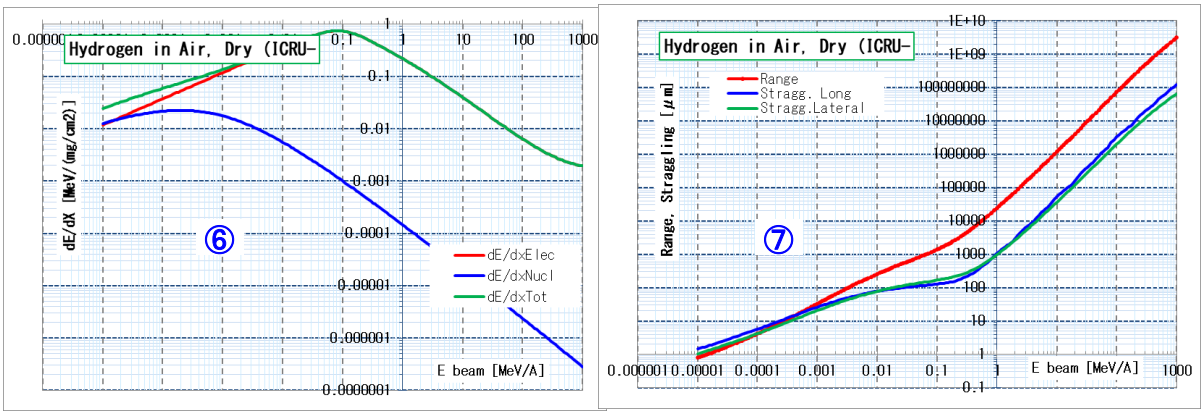
[3] After the conversion finished, a new sheet will be appeared.



[4] Edit the new sheet.



① change the name of the work sheet.
② check the "row#" of Emin and Emax.
③ fill in the cells of Target name short- and long-name.
④ If it is Gas target, fill in appropriate values for Ptbl and Ttbl. They are the Pressure [Pa] and Temperature [℃], respectively, when you specified the Density of the Gas for SRIM-2013 calculation.
⑤ fill in corded information.

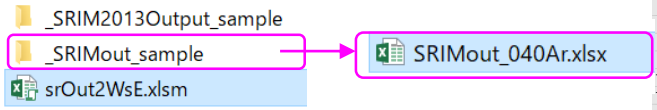


⑥、⑦ check the curves in two graphs are continuous.
If there are some bends, please check and compare the energy-dE/dX-range-straggling table between this sheet and SRIMoutput.txt.

[Step2 -b] convert "SRIMout.xlsx" (CSV file) to SRIMwb format

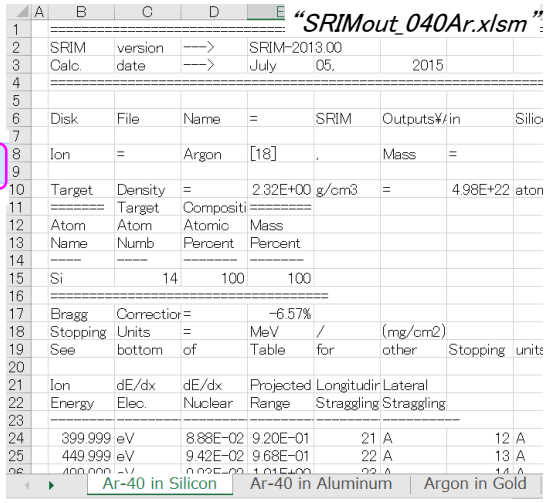
If you have an Excel book (like "SRIMout_*.xlsx") which contains many "SRIMoutput.txt" files, the following utility is convenient.

Some sample files for the "SRIMout.xslx" are included in this folder.

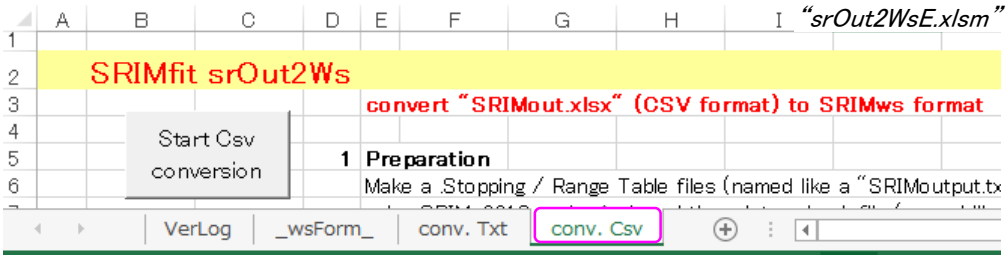


Other samples can be downloaded from SRIMfit Home Page.

cf.) How to read a "SRIMoutput.txt" file into "SRIMout_*.xslx" , use File menu and [Open: Text Import Wizard: Delimited : Space].



[1] Open "srOut2WsE.xlsm" and show "conv.Csv" sheet.



[2] Push "Start CSV conversion" button and Select the "SRIMout * .xlsx" file.

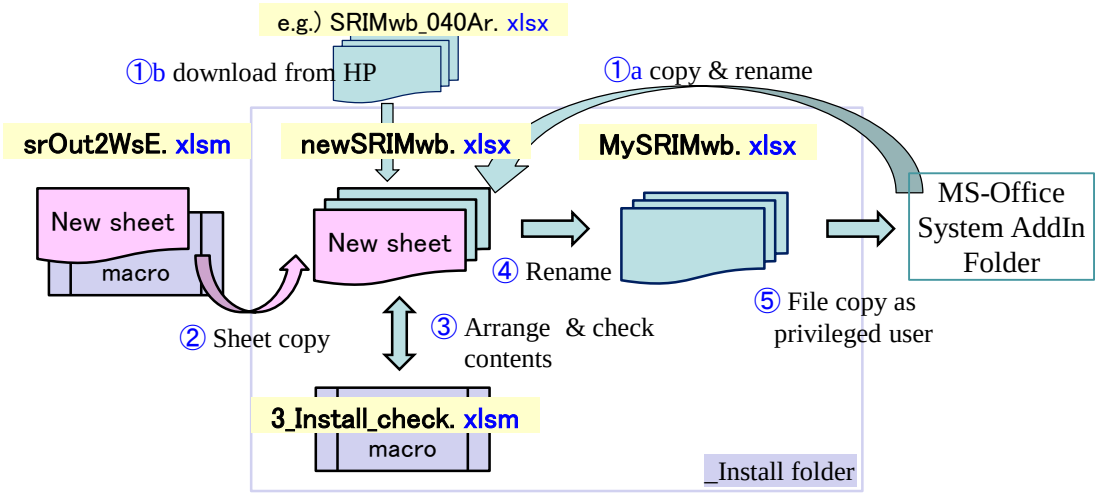
All sheets included in the "SRIMout *.xlsx" will be converted.
As it takes time, please wait for a while.

[3] After the conversion finished, new sheets will be appeared.

[4] Edit the new sheets.

This procedure is same as mentioned before.

[Step3] arrange and check MySRIMwb.xlsx



[1] Work flow for arranging MySRIMwb

First, prepare a temporary SRIMwb book named, for example, "newSRIMwb.xlsx".
Here, a different file name from "MySRIMwb.xlsx" is needed as the Excel can not doubly open files with same file name. You can make it by ①a coping "MySRIMwb.xlsx" from the system AddIn folder, or by ①b downloading from SRIMfit Home Page.

② copy new sheet created in Step2) from "srOut2WsE.xlsm" to the temporary book.
Because a macro code included in the "srOut2WsE.xlsm" is not needed.

③ Check & arrange the content of the temporary book by using "3_Install_check.xlsm".

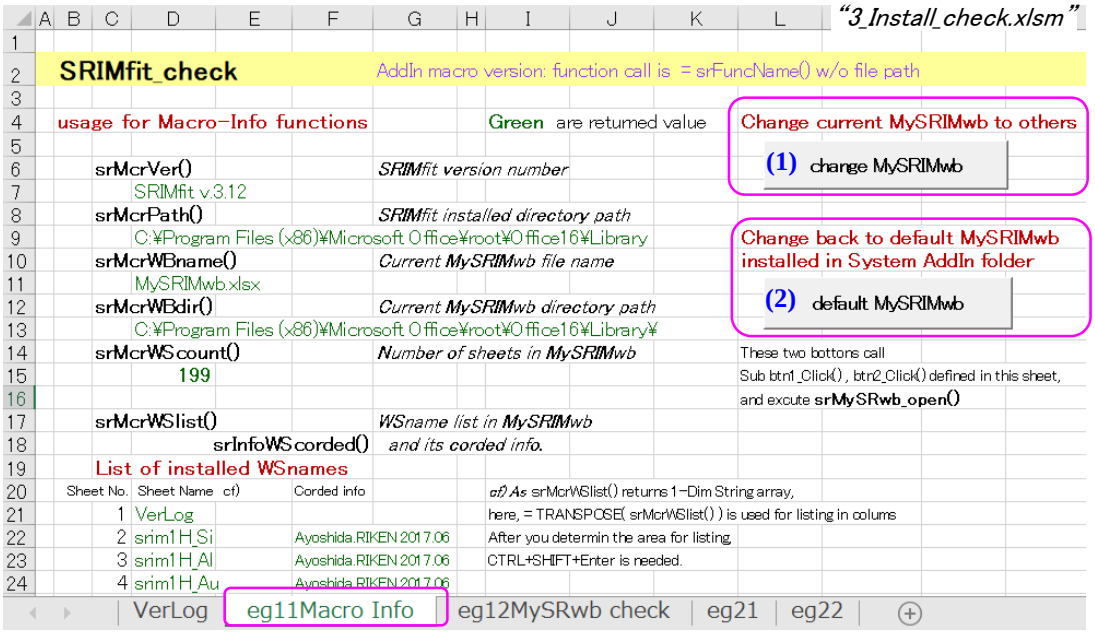
④ Rename the temporary book as "MySRIMwb.xslx"

⑤ and copy it into the system AddIn folder as a privileged user mode.

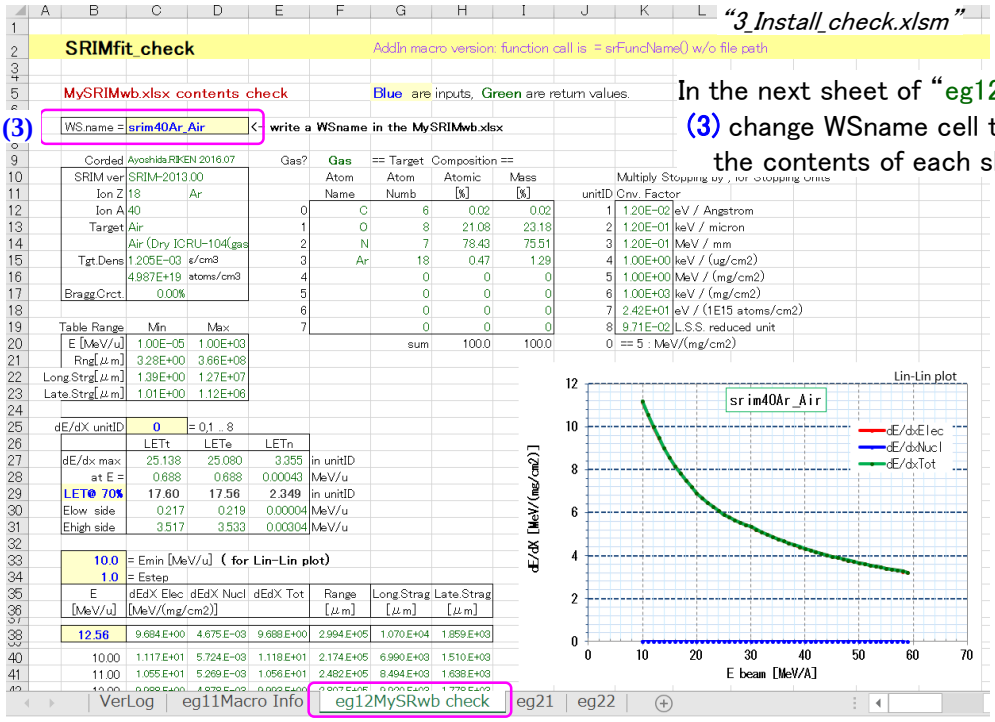
[2] Arrange and check the temporary book ③

You can arrange the temporary SRIMwb book named "newSRIMwb.xlsx".
You can pick and choose needed sheets, and eliminate others.
A minimum number of sheets will speed up for starting Excel program.
In order to check the contents of the temporary book,
a macro included book of "3_Install_check.xlsm" included in the "_Install folder" is available.

[2] Arrange and check the temporary book ③ *(cont.)*



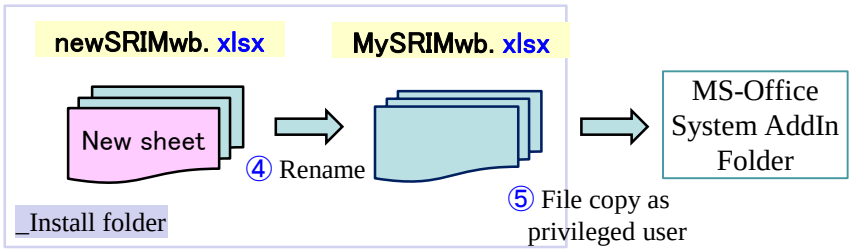
Open a macro included book of "3_Install_check.xlsm" and show "eg11Macro Info" sheet.
push "change MySRIMwb" button (1), and select new SRIMwb of "newSRIMwb.xlsx".
Then the macro information listed in this sheet will change to the new one.



In the next sheet of "eg12MySRIMwb check",
(3) change WSname cell then you can check the contents of each sheet.

You can change back to the default "MySRIMwb.xlsx" when you push (2) button.
or you restart Excel system.

[3] Rename ④ and copy ⑤



④ Rename the temporary book of "newSRIMwb.xlsx" as "MySRIMwb.xslx"
⑤ and copy it into the system AddIn folder as a privileged user mode.
For the detail, please see "Installation manual"

That's it.
I would be grateful if this SRIMfit may help you.

Sincerely Yours,
Industrial Application Research Team,
RIKEN Nishina Center for Accelerator-Based Science, Japan