

TChem

0.2

Generated by Doxygen 1.7.5.1

Sun Nov 27 2011 21:48:52

Contents

1	Thermo-chemical software library	1
2	Module Index	3
2.1	Modules	3
3	Class Index	5
3.1	Class List	5
4	File Index	7
4.1	File List	7
5	Module Documentation	9
5.1	Equation of state	9
5.1.1	Function Documentation	10
5.1.1.1	TC_getRhoMixMI	10
5.1.1.2	TC_getRhoMixMs	10
5.1.1.3	TC_getTmixMI	11
5.1.1.4	TC_getTmixMs	11
5.1.1.5	TCDND_getRhoMixMI	12
5.1.1.6	TCDND_getRhoMixMs	12
5.1.1.7	TCDND_getTmixMI	12
5.1.1.8	TCDND_getTmixMs	13
5.2	Thermodynamic properties	14
5.2.1	Function Documentation	15

5.2.1.1	TC_getCpSpecMI	15
5.2.1.2	TC_getCpSpecMs	15
5.2.1.3	TC_getHspecMI	16
5.2.1.4	TC_getHspecMs	16
5.2.1.5	TC_getMI2CpMixMI	17
5.2.1.6	TC_getMI2HmixMI	17
5.2.1.7	TC_getMs2CpMixMs	17
5.2.1.8	TC_getMs2CvMixMs	18
5.2.1.9	TC_getMs2HmixMs	18
5.2.1.10	TCDND_getCpSpecMI	19
5.2.1.11	TCDND_getCpSpecMs	19
5.2.1.12	TCDND_getHspecMI	19
5.2.1.13	TCDND_getHspecMs	20
5.2.1.14	TCDND_getMI2CpMixMI	20
5.2.1.15	TCDND_getMI2HmixMI	20
5.2.1.16	TCDND_getMs2CpMixMs	21
5.2.1.17	TCDND_getMs2CvMixMs	21
5.2.1.18	TCDND_getMs2HmixMs	22
5.3	Initialization	23
5.3.1	Function Documentation	23
5.3.1.1	TC_createTables	23
5.3.1.2	TC_initChem	23
5.3.1.3	TC_kmodint_	24
5.3.1.4	TC_setRefVal	24
5.3.1.5	TC_setThermoPres	25
5.4	Jacobians	26
5.4.1	Function Documentation	27
5.4.1.1	TC_getJacRPTYN	27
5.4.1.2	TC_getJacRPTYNanl	27
5.4.1.3	TC_getJacRPTYNnum	28
5.4.1.4	TC_getJacTYN	28

5.4.1.5	TC_getJacTYNanl	28
5.4.1.6	TC_getJacTYNm1	29
5.4.1.7	TC_getJacTYNm1anl	29
5.4.1.8	TCDND_getJacTYN	30
5.4.1.9	TCDND_getJacTYNanl	30
5.4.1.10	TCDND_getJacTYNm1	31
5.4.1.11	TCDND_getJacTYNm1anl	31
5.5	Source terms	32
5.5.1	Function Documentation	32
5.5.1.1	TC_getSrc	32
5.5.1.2	TC_getSrcCons	33
5.5.1.3	TCDND_getSrc	33
5.5.1.4	TCDND_getSrcCons	34
5.6	Max. no. of parameters	35
5.7	No. of reactions	36
5.8	No. of species, variables, etc.	37
6	Class Documentation	39
6.1	element Struct Reference	39
6.1.1	Detailed Description	39
6.2	elemtable Struct Reference	39
6.2.1	Detailed Description	40
6.3	reaction Struct Reference	40
6.3.1	Detailed Description	41
6.4	species Struct Reference	41
6.4.1	Detailed Description	42
7	File Documentation	43
7.1	copyright.h File Reference	43
7.1.1	Detailed Description	43
7.2	TC_chg.c File Reference	43
7.2.1	Detailed Description	44

7.2.2	Function Documentation	44
7.2.2.1	TC_chgArhenFor	44
7.2.2.2	TC_chgArhenForBack	44
7.2.2.3	TC_chgArhenRev	44
7.2.2.4	TC_chgArhenRevBack	45
7.3	TC_defs.h File Reference	45
7.3.1	Detailed Description	52
7.3.2	Function Documentation	52
7.3.2.1	TC_errorINI	52
7.3.2.2	TC_errorMSG	52
7.3.2.3	TC_getReacRates	53
7.4	TC_init.c File Reference	53
7.4.1	Detailed Description	54
7.4.2	Function Documentation	54
7.4.2.1	TC_errorINI	54
7.4.2.2	TC_errorMSG	54
7.5	TC_interface.h File Reference	55
7.5.1	Detailed Description	62
7.5.2	Function Documentation	62
7.5.2.1	TC_chgArhenFor	62
7.5.2.2	TC_chgArhenForBack	62
7.5.2.3	TC_chgArhenRev	63
7.5.2.4	TC_chgArhenRevBack	63
7.5.2.5	TC_getArhenFor	63
7.5.2.6	TC_getArhenRev	63
7.5.2.7	TC_getMI2Ms	64
7.5.2.8	TC_getMI2Wmix	64
7.5.2.9	TC_getMs2Cc	65
7.5.2.10	TC_getMs2MI	65
7.5.2.11	TC_getMs2Wmix	66
7.5.2.12	TC_getRfrb	66

7.5.2.13	TC_getRops	67
7.5.2.14	TC_getSmass	67
7.5.2.15	TC_getSnames	67
7.5.2.16	TC_getSpos	68
7.5.2.17	TC_getStoiCoef	68
7.5.2.18	TC_getStoiCoefReac	68
7.5.2.19	TC_getTXC2RRml	69
7.5.2.20	TC_getTXC2RRms	69
7.5.2.21	TC_getTY2RRml	70
7.5.2.22	TC_getTY2RRms	70
7.5.2.23	TC_removeReaction	71
7.5.2.24	TCDND_getMI2Ms	71
7.5.2.25	TCDND_getMI2Wmix	71
7.5.2.26	TCDND_getMs2Cc	72
7.5.2.27	TCDND_getMs2MI	72
7.5.2.28	TCDND_getMs2Wmix	73
7.5.2.29	TCDND_getTXC2RRml	73
7.5.2.30	TCDND_getTXC2RRms	74
7.5.2.31	TCDND_getTY2RRml	74
7.5.2.32	TCDND_getTY2RRms	74
7.6	TC_kmodint.c File Reference	75
7.6.1	Detailed Description	78
7.6.2	Function Documentation	81
7.6.2.1	checkelemylist	81
7.6.2.2	extractWordLeft	81
7.6.2.3	extractWordLeftauxline	81
7.6.2.4	getreactions	82
7.6.2.5	setperiodictable	82
7.6.2.6	tab2space	82
7.7	TC_mlms.c File Reference	83
7.7.1	Detailed Description	84

7.7.2	Function Documentation	84
7.7.2.1	TC_getMI2Ms	84
7.7.2.2	TC_getMI2Wmix	85
7.7.2.3	TC_getMs2Cc	85
7.7.2.4	TC_getMs2MI	86
7.7.2.5	TC_getMs2Wmix	86
7.7.2.6	TCDND_getMI2Ms	87
7.7.2.7	TCDND_getMI2Wmix	87
7.7.2.8	TCDND_getMs2Cc	88
7.7.2.9	TCDND_getMs2MI	88
7.7.2.10	TCDND_getMs2Wmix	89
7.8	TC_params.h File Reference	89
7.8.1	Detailed Description	90
7.8.2	Define Documentation	90
7.8.2.1	ATMPA	90
7.8.2.2	CALJO	90
7.8.2.3	EVOLT	90
7.8.2.4	KBOLT	90
7.8.2.5	LENGTHOFELEMNAME	90
7.8.2.6	LENGTHOFSPECNAME	90
7.8.2.7	MAX	91
7.8.2.8	MIN	91
7.8.2.9	NAVOG	91
7.8.2.10	NSPECREACMAX	91
7.8.2.11	NTHRDBMAX	91
7.8.2.12	NUMBEROFELEMINSPEC	91
7.8.2.13	REACBALANCE	91
7.8.2.14	RUNIV	92
7.9	TC_rr.c File Reference	92
7.9.1	Detailed Description	93
7.9.2	Function Documentation	94

7.9.2.1	TC_getArhenFor	94
7.9.2.2	TC_getArhenRev	94
7.9.2.3	TC_getReacRates	94
7.9.2.4	TC_getRfrb	95
7.9.2.5	TC_getRops	95
7.9.2.6	TC_getStoiCoef	96
7.9.2.7	TC_getStoiCoefReac	96
7.9.2.8	TC_getTXC2RRml	96
7.9.2.9	TC_getTXC2RRms	97
7.9.2.10	TC_getTY2RRml	97
7.9.2.11	TC_getTY2RRms	98
7.9.2.12	TCDND_getTXC2RRml	98
7.9.2.13	TCDND_getTXC2RRms	99
7.9.2.14	TCDND_getTY2RRml	99
7.9.2.15	TCDND_getTY2RRms	99
7.10	TC_spec.c File Reference	100
7.10.1	Detailed Description	100
7.10.2	Function Documentation	100
7.10.2.1	TC_getSmass	100
7.10.2.2	TC_getSnames	101
7.10.2.3	TC_getSpos	101
7.11	TC_src.c File Reference	101
7.11.1	Detailed Description	103
7.12	TC_thermo.c File Reference	103
7.12.1	Detailed Description	105
7.13	TC_utils.c File Reference	106
7.13.1	Detailed Description	106

Chapter 1

Thermo-chemical software library

Authors

Cosmin Safta, Habib Najm, Omar Knio

The TChem toolkit is a software library that enables numerical simulations using complex chemistry and facilitates the analysis of detailed kinetic models. The toolkit provides capabilities for thermodynamic properties based on NASA polynomials, kinetic model reaction rates (both for individual reactions and for species). It incorporates methods that can selectively modify reaction parameters for sensitivity analysis and uncertainty quantification. The library contains several functions that provide analytically computed Jacobian matrices necessary for the efficient time advancement and analysis of detailed kinetic models.

The software is released subject to the terms of the GNU Lesser General Public License (<http://www.gnu.org/copyleft/lesser.html>) Please read the license information before downloading the software.

Copyright 2011 Sandia Corporation. Under the terms of Contract DE-AC04-94AL85000 with Sandia Corporation, the U.S. Government retains certain rights in this software.

Chapter 2

Module Index

2.1 Modules

Here is a list of all modules:

Equation of state	9
Thermodynamic properties	14
Initialization	23
Jacobians	26
Source terms	32
Max. no. of parameters	35
No. of reactions	36
No. of species, variables, etc.	37

Chapter 3

Class Index

3.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

element	Element data	39
elemtable	Entry in the table of periodic elements	39
reaction	Reaction data	40
species	Species data	41

Chapter 4

File Index

4.1 File List

Here is a list of all documented files with brief descriptions:

copyright.h	Copyright 2011 Sandia Corporation. Under the terms of Contract D-E-AC04-94AL85000 with Sandia Corporation, the U.S. Government retains certain rights in this software	43
TC_chg.c	Functions for changing Arrhenius rate factors	43
TC_defs.h	Definitions of variables names used by the library	45
TC_init.c	Initialize chemical library	53
TC_interface.h	Header file to be included in user's code. Contains function definitions	55
TC_kmodint.c	Collection of functions used to parse kinetic models from files	75
TC_kmodint.h	??
TC_mlms.c	Mass fractions - Mole fractions - Molar concentrations - Molecular weight	83
TC_params.h	Definitions of parameters and constants	89
TC_rr.c	Reaction rate functions	92
TC_spec.c	Species info	100

TC_src.c	Source term and Jacobian functions	101
TC_thermo.c	Equation of state and thermodynamic functions	103
TC_utils.c	Various utilities used by other functions	106

Chapter 5

Module Documentation

5.1 Equation of state

Functions

- int [TCDND_getRhoMixMs](#) (double *scal, int Nvars, double *rhomix)
Computes density based on temperature and species mass fractions using the equation of state. Input temperature is normalized, output density also normalized before exit.
- int [TC_getRhoMixMs](#) (double *scal, int Nvars, double *rhomix)
Computes density based on temperature and species mass fractions using the equation of state.
- int [TCDND_getRhoMixMI](#) (double *scal, int Nvars, double *rhomix)
Computes density based on temperature and species mole fractions using the equation of state. Input temperature is normalized, output density also normalized before exit.
- int [TC_getRhoMixMI](#) (double *scal, int Nvars, double *rhomix)
Computes density based on temperature and species mole fractions using the equation of state.
- int [TCDND_getTmixMs](#) (double *scal, int Nvars, double *Tmix)
Computes temperature based on density and species mass fractions using the equation of state. Input density is normalized, output temperature also normalized before exit.
- int [TC_getTmixMs](#) (double *scal, int Nvars, double *Tmix)
Computes temperature based on density and species mass fractions using the equation of state.
- int [TCDND_getTmixMI](#) (double *scal, int Nvars, double *Tmix)

Computes temperature based on density and species mole fractions using the equation of state. Input density is normalized, output temperature also normalized before exit.

- int `TC_getTmixMI` (double *scal, int Nvars, double *Tmix)

Computes temperature based on density and species mole fractions using the equation of state.

5.1.1 Function Documentation

5.1.1.1 int TC_getRhoMixMI (double * scal, int Nvars, double * rhomix)

Computes density based on temperature and species mole fractions using the equation of state.

Parameters

<code>scal</code>	: array of Nspec +1 doubles (T,X ₁ ,X ₂ ,...,X _{Nspec}), temperature T [K], mole fractions X []
<code>Nvars</code>	: no. of variables = Nspec +1

Returns

`rhomix` : pointer to mixture density [kg/m³]

References `TC_errorMSG()`, `TC_Nspec_`, `TC_Nvars_`, and `TC_sMass_`.

Referenced by `TCDND_getRhoMixMI()`.

5.1.1.2 int TC_getRhoMixMs (double * scal, int Nvars, double * rhomix)

Computes density based on temperature and species mass fractions using the equation of state.

Parameters

<code>scal</code>	: array of Nspec +1 doubles (T,Y ₁ ,Y ₂ ,...,Y _{Nspec}), temperature T [K], mass fractions Y []
<code>Nvars</code>	: no. of variables = Nspec +1

Returns

`rhomix` : pointer to mixture density [kg/m³]

References `TC_errorMSG()`, `TC_Nspec_`, `TC_Nvars_`, and `TC_sMass_`.

Referenced by TC_getJacRPTYNanl(), TC_getJacRPTYNnum(), TC_getJacTYN(), TC_getJacTYNanl(), TC_getJacTYNm1(), TC_getJacTYNm1anl(), TC_getMs2Cc(), TC_getSrc(), and TCDND_getRhoMixMs().

5.1.1.3 int TC_getTmixMI (double * scal, int Nvars, double * Tmix)

Computes temperature based on density and species mole fractions using the equation of state.

Parameters

<i>scal</i>	: array of Nspec +1 doubles (rho,X ₁ ,X ₂ ,...,X _{Nspec}), density rho [kg/m ³], mole fractions X []
<i>Nvars</i>	: no. of variables = Nspec +1

Returns

Tmix : pointer to temperature [K]

References TC_errorMSG(), TC_Nspec_, TC_Nvars_, and TC_sMass_.

Referenced by TCDND_getTmixMI().

5.1.1.4 int TC_getTmixMs (double * scal, int Nvars, double * Tmix)

Computes temperature based on density and species mass fractions using the equation of state.

Parameters

<i>scal</i>	: array of Nspec +1 doubles (rho,Y ₁ ,Y ₂ ,...,Y _{Nspec}), density rho [kg/m ³], mass fractions Y []
<i>Nvars</i>	: no. of variables = Nspec +1

Returns

Tmix : pointer to temperature [K]

References TC_errorMSG(), TC_Nspec_, TC_Nvars_, and TC_sMass_.

Referenced by TC_getSrcCons(), and TCDND_getTmixMs().

5.1.1.5 int TCDND_getRhoMixMI (double * scal, int Nvars, double * rhomix)

Computes density based on temperature and species mole fractions using the equation of state. Input temperature is normalized, output density also normalized before exit.

Parameters

<i>scal</i>	: array of $N_{\text{spec}} + 1$ doubles (T, X_1 , X_2 , ..., $X_{N_{\text{spec}}}$), temperature T [K], mole fractions X []
<i>Nvars</i>	: no. of variables = $N_{\text{spec}} + 1$

Returns

rhomix : pointer to mixture density [kg/m^3]

References TC_errorMSG(), TC_getRhoMixMI(), and TC_Nvars_.

5.1.1.6 int TCDND_getRhoMixMs (double * scal, int Nvars, double * rhomix)

Computes density based on temperature and species mass fractions using the equation of state. Input temperature is normalized, output density also normalized before exit.

Parameters

<i>scal</i>	: array of $N_{\text{spec}} + 1$ doubles (T, Y_1 , Y_2 , ..., $Y_{N_{\text{spec}}}$), temperature T [K], mass fractions Y []
<i>Nvars</i>	: no. of variables = $N_{\text{spec}} + 1$

Returns

rhomix : pointer to mixture density [kg/m^3]

References TC_errorMSG(), TC_getRhoMixMs(), and TC_Nvars_.

5.1.1.7 int TCDND_getTmixMI (double * scal, int Nvars, double * Tmix)

Computes temperature based on density and species mole fractions using the equation of state. Input density is normalized, output temperature also normalized before exit.

Parameters

<i>scal</i>	: array of $N_{\text{spec}} + 1$ doubles (ρ , X_1 , X_2 , ..., $X_{N_{\text{spec}}}$), density ρ [kg/m^3], mole fractions X []
<i>Nvars</i>	: no. of variables = $N_{\text{spec}} + 1$

Returns

Tmix : pointer to temperature [K]

References TC_errorMSG(), TC_getTmixMI(), and TC_Nvars_.

5.1.1.8 int TCDND_getTmixMs (double * scal, int Nvars, double * Tmix)

Computes temperature based on density and species mass fractions using the equation of state. Input density is normalized, output temperature also normalized before exit.

Parameters

<i>scal</i>	: array of N _{spec} +1 doubles (rho, Y ₁ , Y ₂ , ..., Y _{Nspec}), density rho [kg/m ³], mass fractions Y []
<i>Nvars</i>	: no. of variables = N _{spec} +1

Returns

Tmix : pointer to temperature [K]

References TC_errorMSG(), TC_getTmixMs(), and TC_Nvars_.

5.2 Thermodynamic properties

Functions

- int [TCDND_getMs2CpMixMs](#) (double *scal, int Nvars, double *cpmix)
Computes mixture specific heat at constant pressure based on temperature and species mass fractions. Input temperature is normalized, output specific heat is also normalized before exit.
- int [TC_getMs2CpMixMs](#) (double *scal, int Nvars, double *cpmix)
Computes mixture specific heat at constant pressure based on temperature and species mass fractions.
- int [TCDND_getMs2CvMixMs](#) (double *scal, int Nvars, double *cvmix)
Computes mixture specific heat at constant volume based on temperature and species mass fractions. Input temperature is normalized, output specific heat is also normalized before exit.
- int [TC_getMs2CvMixMs](#) (double *scal, int Nvars, double *cvmix)
Computes mixture specific heat at constant volume based on temperature and species mass fractions.
- int [TCDND_getMI2CpMixMI](#) (double *scal, int Nvars, double *cpmix)
Computes mixture heat capacity at constant pressure based on temperature and species mole fractions. Input temperature is normalized, output specific heat is also normalized before exit.
- int [TC_getMI2CpMixMI](#) (double *scal, int Nvars, double *cpmix)
Computes mixture specific heat at constant pressure based on temperature and species mole fractions.
- int [TCDND_getCpSpecMs](#) (double t, int Nspec, double *cpi)
Computes species specific heat at constant pressure based on temperature. Input temperature is normalized, output specific heats are also normalized before exit.
- int [TC_getCpSpecMs](#) (double t, int Nspec, double *cpi)
Computes species specific heat at constant pressure based on temperature.
- int [TCDND_getCpSpecMI](#) (double t, int Nspec, double *cpi)
Computes species heat capacities at constant pressure based on temperature. Input temperature is normalized, output heat capacities are also normalized before exit.
- int [TC_getCpSpecMI](#) (double t, int Nspec, double *cpi)
Computes species heat capacities at constant pressure based on temperature.
- int [TCDND_getMs2HmixMs](#) (double *scal, int Nvars, double *hmix)
Computes mixture specific enthalpy based on temperature and species mass fractions. Input temperature is normalized, output enthalpy is normalized before exit.
- int [TC_getMs2HmixMs](#) (double *scal, int Nvars, double *hmix)
Computes mixture specific enthalpy based on temperature and species mass fractions.
- int [TCDND_getMI2HmixMI](#) (double *scal, int Nvars, double *hmix)
Computes mixture molar enthalpy based on temperature and species mole fractions. Input temperature is normalized, output enthalpy is normalized before exit.

- int `TC_getMI2HmixMI` (double *scal, int Nvars, double *hmix)
Computes mixture molar enthalpy based on temperature and species mole fractions.
- int `TCDND_getHspecMs` (double t, int Nspec, double *hi)
Computes species specific enthalpies based on temperature. Input temperature is normalized, output enthalpies are also normalized before exit.
- int `TC_getHspecMs` (double t, int Nspec, double *hi)
Computes species specific enthalpies based on temperature.
- int `TCDND_getHspecMI` (double t, int Nspec, double *hi)
Computes species molar enthalpies based on temperature. Input temperature is normalized, output enthalpies are also normalized before exit.
- int `TC_getHspecMI` (double t, int Nspec, double *hi)
Computes species molar enthalpies based on temperature.

5.2.1 Function Documentation

5.2.1.1 int TC_getCpSpecMI (double t, int Nspec, double * cpi)

Computes species heat capacities at constant pressure based on temperature.

Parameters

<i>t</i>	: temperature T [K]
<i>Nspec</i>	: no. of species

Returns

cpi : array with species heat capacities at constant pressure [J/(kmol.K)]

References `TC_errorMSG()`, and `TC_Nspec_`.

Referenced by `TC_getMI2CpMixMI()`, and `TCDND_getCpSpecMI()`.

5.2.1.2 int TC_getCpSpecMs (double t, int Nspec, double * cpi)

Computes species specific heat at constant pressure based on temperature.

Parameters

<i>t</i>	: temperature T [K]
<i>Nspec</i>	: no. of species

Returns

cpj : array with species specific heats at constant pressure [J/(kg.K)]

References TC_errorMSG(), and TC_Nspec_.

Referenced by TC_getJacRPTYNnum(), TC_getMs2CpMixMs(), and TCDND_getCp-SpecMs().

5.2.1.3 int TC_getHspecMI (double *t*, int *Nspec*, double * *hi*)

Computes species molar enthalpies based on temperature.

Parameters

<i>t</i>	: temperature T [K]
<i>Nspec</i>	: no. of species

Returns

hi : array with species molar enthalpies [J/kmol]

References TC_errorMSG(), and TC_Nspec_.

Referenced by TC_getMI2HmixMI(), and TCDND_getHspecMI().

5.2.1.4 int TC_getHspecMs (double *t*, int *Nspec*, double * *hi*)

Computes species specific enthalpies based on temperature.

Parameters

<i>t</i>	: temperature T [K]
<i>Nspec</i>	: no. of species

Returns

hi : array with species specific enthalpies [J/kg]

References TC_errorMSG(), and TC_Nspec_.

Referenced by TC_getJacRPTYNani(), TC_getJacRPTYNnum(), TC_getMs2Hmix-Ms(), TC_getSrc(), TC_getSrcCons(), and TCDND_getHspecMs().

5.2.1.5 int TC_getMI2CpMixMI (double * scal, int Nvars, double * cpmix)

Computes mixture specific heat at constant pressure based on temperature and species mole fractions.

Parameters

<i>scal</i>	: array of N _{spec} + 1 doubles (T, X ₁ , X ₂ , ..., X _{N_{spec}}), temperature T [K], mole fractions X []
<i>Nvars</i>	: no. of variables = N _{spec} + 1

Returns

cpmix : pointer to mixture specific heat at constant pressure [J/(kmol.K)]

References TC_errorMSG(), TC_getCpSpecMI(), TC_Nspec_, and TC_Nvars_.

Referenced by TCDND_getMI2CpMixMI().

5.2.1.6 int TC_getMI2HmixMI (double * scal, int Nvars, double * hmix)

Computes mixture molar enthalpy based on temperature and species mole fractions.

Parameters

<i>scal</i>	: array of N _{spec} + 1 doubles (T, X ₁ , X ₂ , ..., X _{N_{spec}}), temperature T [K], mole fractions X []
<i>Nvars</i>	: no. of variables = N _{spec} + 1

Returns

hmix : pointer to mixture molar enthalpy [J/kmol]

References TC_errorMSG(), TC_getHspecMI(), TC_Nspec_, and TC_Nvars_.

Referenced by TCDND_getMI2HmixMI().

5.2.1.7 int TC_getMs2CpMixMs (double * scal, int Nvars, double * cpmix)

Computes mixture specific heat at constant pressure based on temperature and species mass fractions.

Parameters

<i>scal</i>	: array of N _{spec} + 1 doubles (T, Y ₁ , Y ₂ , ..., Y _{N_{spec}}), temperature T [K], mass fractions Y []
<i>Nvars</i>	: no. of variables = N _{spec} + 1

Returns

cpmix : pointer to mixture specific heat at constant pressure [J/(kg.K)]

References TC_errorMSG(), TC_getCpSpecMs(), TC_Nspec_, and TC_Nvars_.

Referenced by TC_getJacRPTYNanl(), TC_getJacRPTYNnum(), TC_getMs2CvMixMs(), TC_getSrc(), TC_getSrcCons(), and TCDND_getMs2CpMixMs().

5.2.1.8 int TC_getMs2CvMixMs (double * scal, int Nvars, double * cvmix)

Computes mixture specific heat at constant volume based on temperature and species mass fractions.

Parameters

<i>scal</i>	: array of Nspec +1 doubles (T, Y ₁ , Y ₂ , ..., Y _{Nspec}), temperature T [K], mass fractions Y []
<i>Nvars</i>	: no. of variables = Nspec +1

Returns

cvmix : pointer to mixture specific heat at constant volume [J/(kg.K)]

References TC_errorMSG(), TC_getMs2CpMixMs(), TC_Nspec_, TC_Nvars_, and TC_sMass_.

Referenced by TCDND_getMs2CvMixMs().

5.2.1.9 int TC_getMs2HmixMs (double * scal, int Nvars, double * hmix)

Computes mixture specific enthalpy based on temperature and species mass fractions.

Parameters

<i>scal</i>	: array of Nspec +1 doubles (T, Y ₁ , Y ₂ , ..., Y _{Nspec}), temperature T [K], mass fractions Y []
<i>Nvars</i>	: no. of variables = Nspec +1

Returns

hmix : pointer to mixture specific enthalpy [J/kg]

References TC_errorMSG(), TC_getHspecMs(), TC_Nspec_, and TC_Nvars_.

Referenced by TCDND_getMs2HmixMs().

5.2.1.10 int TCDND_getCpSpecMI (double *t*, int *Nspec*, double * *cpu*)

Computes species heat capacities at constant pressure based on temperature. Input temperature is normalized, output heat capacities are also normalized before exit.

Parameters

<i>t</i>	: temperature T [K]
<i>Nspec</i>	: no. of species

Returns

cpu : array with species heat capacities at constant pressure [J/(kmol.K)]

References TC_errorMSG(), TC_getCpSpecMI(), and TC_Nspec_.

5.2.1.11 int TCDND_getCpSpecMs (double *t*, int *Nspec*, double * *cpu*)

Computes species specific heat at constant pressure based on temperature. Input temperature is normalized, output specific heats are also normalized before exit.

Parameters

<i>t</i>	: temperature T [K]
<i>Nspec</i>	: no. of species

Returns

cpu : array with species specific heats at constant pressure [J/(kg.K)]

References TC_errorMSG(), TC_getCpSpecMs(), and TC_Nspec_.

5.2.1.12 int TCDND_getHspecMI (double *t*, int *Nspec*, double * *hi*)

Computes species molar enthalpies based on temperature. Input temperature is normalized, output enthalpies are also normalized before exit.

Parameters

<i>t</i>	: temperature T [K]
<i>Nspec</i>	: no. of species

Returns

hi : array with species molar enthalpies [J/kmol]

References TC_errorMSG(), TC_getHspecMI(), and TC_Nspec_.

5.2.1.13 int TCDND_getHspecMs (double t, int Nspec, double * hi)

Computes species specific enthalpies based on temperature. Input temperature is normalized, output enthalpies are also normalized before exit.

Parameters

t	: temperature T [K]
Nspec	: no. of species

Returns

hi : array with species specific enthalpies [J/kg]

References TC_errorMSG(), TC_getHspecMs(), and TC_Nspec_.

5.2.1.14 int TCDND_getMI2CpMixMI (double * scal, int Nvars, double * cpmix)

Computes mixture heat capacity at constant pressure based on temperature and species mole fractions. Input temperature is normalized, output specific heat is also normalized before exit.

Parameters

scal	: array of Nspec +1 doubles (T, X ₁ , X ₂ , ..., X _{Nspec}), temperature T [K], mole fractions X []
Nvars	: no. of variables = Nspec +1

Returns

cpmix : pointer to mixture heat capacity at constant pressure [J/(kmol.K)]

References TC_errorMSG(), TC_getMI2CpMixMI(), and TC_Nvars_.

5.2.1.15 int TCDND_getMI2HmixMI (double * scal, int Nvars, double * hmix)

Computes mixture molar enthalpy based on temperature and species mole fractions. Input temperature is normalized, output enthalpy is normalized before exit.

Parameters

<i>scal</i>	: array of $N_{\text{spec}} + 1$ doubles ($T, X_1, X_2, \dots, X_{N_{\text{spec}}}$), temperature T [K], mole fractions X []
<i>Nvars</i>	: no. of variables = $N_{\text{spec}} + 1$

Returns

hmix : pointer to mixture molar enthalpy [J/kmol]

References TC_errorMSG(), TC_getMI2HmixMI(), and TC_Nvars_.

5.2.1.16 int TCDND_getMs2CpMixMs (double * *scal*, int *Nvars*, double * *cpmix*)

Computes mixture specific heat at constant pressure based on temperature and species mass fractions. Input temperature is normalized, output specific heat is also normalized before exit.

Parameters

<i>scal</i>	: array of $N_{\text{spec}} + 1$ doubles ($T, Y_1, Y_2, \dots, Y_{N_{\text{spec}}}$), temperature T [K], mass fractions Y []
<i>Nvars</i>	: no. of variables = $N_{\text{spec}} + 1$

Returns

cpmix : pointer to mixture specific heat at constant pressure [J/(kg.K)]

References TC_errorMSG(), TC_getMs2CpMixMs(), and TC_Nvars_.

5.2.1.17 int TCDND_getMs2CvMixMs (double * *scal*, int *Nvars*, double * *cvmix*)

Computes mixture specific heat at constant volume based on temperature and species mass fractions. Input temperature is normalized, output specific heat is also normalized before exit.

Parameters

<i>scal</i>	: array of $N_{\text{spec}} + 1$ doubles ($T, Y_1, Y_2, \dots, Y_{N_{\text{spec}}}$), temperature T [K], mass fractions Y []
<i>Nvars</i>	: no. of variables = $N_{\text{spec}} + 1$

Returns

cpmix : pointer to mixture specific heat at constant volume [J/(kg.K)]

References TC_errorMSG(), TC_getMs2CvMixMs(), and TC_Nvars_.

5.2.1.18 int TCDND_getMs2HmixMs (double * scal, int Nvars, double * hmix)

Computes mixture specific enthalpy based on temperature and species mass fractions. Input temperature is normalized, output enthalpy is normalized before exit.

Parameters

<i>scal</i>	: array of N _{spec} +1 doubles (T, Y ₁ , Y ₂ , ..., Y _{N_{spec}}), temperature T [K], mass fractions Y []
<i>Nvars</i>	: no. of variables = N _{spec} +1

Returns

hmix : pointer to mixture specific enthalpy [J/kg]

References TC_errorMSG(), TC_getMs2HmixMs(), and TC_Nvars_.

5.3 Initialization

Functions

- int `TC_initChem` (char *mechfile, char *thermofile, int tab, double delT)

Initialize library.
- void `TC_setRefVal` (double rhoref, double pref, double Tref, double Wref, double Daref, double omgref, double cpref, double href, double timref)

Set reference values to the library.
- void `TC_setNonDim` ()

Set library to function in non-dimensional mode.
- void `TC_setDim` ()

Set library to function in dimensional mode (default)
- void `TC_setThermoPres` (double pressure)

Send thermodynamic pressure to the library.
- int `TC_makeSpace` ()

Allocate internal work arrays for the library. Should not be called by external functions.
- int `TC_createTables` (double delT)

Create tables for temperature dependent terms. Should not be called by external functions.
- int `TC_kmodint_` (char *mechfile, int *lmech, char *thermofile, int *lthrm)

Kinetic model interpretor.

5.3.1 Function Documentation

5.3.1.1 int TC_createTables (double delT)

Create tables for temperature dependent terms. Should not be called by external functions.

Parameters

<code>delT</code>	: temperature step size [K]
-------------------	-----------------------------

References `TC_Nreac_`, and `TC_Nspec_`.

Referenced by `TC_initChem()`.

5.3.1.2 int TC_initChem (char * mechfile, char * thermofile, int tab, double delT)

Initialize library.

Parameters

<i>mechfile</i>	: name of file containing kinetic model in chemkin format
<i>thermofile</i>	: name of file containing coefficients for NASA polynomials
<i>tab</i>	: flag to tabulate temperature dependent terms (tab=1 -> create tables)
<i>delT</i>	: temperature step size [K] for the tables; used only tab=1

References ATMPA, CALJO, fastIntPow(), LENGTHOFELEMENTNAME, LENGTHOFSPECNAME, RUNIV, TC_createTables(), TC_electrIndx_, TC_elemcount_, TC_eMass_, TC_eNames_, TC_errorINI(), TC_kmodint_(), TC_makeSpace(), TC_maxOrdPar_, TC_maxSpecInReac_, TC_maxTbInReac_, TC_nArhPar_, TC_nCpCoef_, TC_Nelem_, TC_nExciReac_, TC_nFallPar_, TC_nFallReac_, TC_nFit1Par_, TC_nFit1Reac_, TC_nHvReac_, TC_nIonEspec_, TC_nIonSpec_, TC_nJanPar_, TC_nJanReac_, TC_nLtPar_, TC_nLtReac_, TC_nMomeReac_, TC_nNASAinter_, TC_nOrdReac_, TC_nNreac_, TC_nRealNuReac_, TC_nRevReac_, TC_nRltReac_, TC_nSpec_, TC_nTdepReac_, TC_nThbReac_, TC_Nvars_, TC_Nvjac_, TC_nXsmiReac_, TC_sCharge_, TC_setThermoPres(), TC_sMass_, TC_sNames_, TC_sPhase_, and TC_sTfit_.

5.3.1.3 int TC_kmodint_(char * *mechfile*, int * *lmech*, char * *thermofile*, int * *lthrm*)

Kinetic model interpretor.

Parameters

<i>mechfile</i>	: name of file containing kinetic model in chemkin format
<i>lmech</i>	: length of mechfile character string
<i>thermofile</i>	: name of file containing coefficients for NASA polynomials
<i>lthrm</i>	: length of thermofile character string

References checkunits(), cleancharstring(), elimleads(), errormsg(), extractWordLeft(), getelements(), getreactions(), getspecies(), getthermo(), MIN, out_formatted(), out_mathem(), out_unformatted(), rescalereac(), setelementmass(), setperiodictable(), verifyreac(), and wordtoupper().

Referenced by TC_initChem().

5.3.1.4 void TC_setRefVal (double *rhoref*, double *pref*, double *Tref*, double *Wref*, double *Daref*, double *omgref*, double *cpref*, double *href*, double *timref*)

Set reference values to the library.

Parameters

<i>rhoref</i>	: density [kg/m^3]
<i>pref</i>	: pressure [N/m^2]
<i>Tref</i>	: temperature [K]

<i>Wref</i>	: molecular weight [$kg/kmol$]
<i>Daref</i>	: Damkohler number []
<i>omgref</i>	: molar reaction rate [$kmol/(m^3 \cdot s)$]
<i>cpref</i>	: specific heat at constant pressure [$J/(kg \cdot K)$]
<i>href</i>	: specific enthalpy [J/kg]
<i>timref</i>	: time [s]

5.3.1.5 void TC_setThermoPres (double *pressure*)

Send thermodynamic pressure to the library.

Parameters

<i>pressure</i>	: thermodynamic pressure [N/m^2]
-----------------	--------------------------------------

Referenced by TC_initChem().

5.4 Jacobians

Functions

- int [TCDND_getJacTYNm1anl](#) (double *scal, int Nspec, double *jac)
Computes analytical Jacobian (dimensional/non-dimensional) for the system $(T, Y_1, Y_2, \dots, Y_{N-1})$ based on temperature T and species mass fractions Y 's.
- int [TC_getJacTYNm1anl](#) (double *scal, int Nspec, double *jac)
Computes analytical Jacobian for the system $(T, Y_1, Y_2, \dots, Y_{N-1})$ based on T and Y 's.
- int [TCDND_getJacTYNanl](#) (double *scal, int Nspec, double *jac)
Computes analytical Jacobian (dimensional/non-dimensional) for the system $(T, Y_1, Y_2, \dots, Y_N)$ based on T and Y 's.
- int [TC_getJacTYNanl](#) (double *scal, int Nspec, double *jac)
Computes analytical Jacobian for the system $(T, Y_1, Y_2, \dots, Y_N)$ based on T and Y 's.
- int [TCDND_getJacTYNm1](#) (double *scal, int Nspec, double *jac, unsigned int useJacAnl)
Computes (analytical or numerical) Jacobian (dimensional/non-dimensional) for the system $(T, Y_1, Y_2, \dots, Y_{N-1})$ based on temperature T and species mass fractions Y 's.
- int [TC_getJacTYNm1](#) (double *scal, int Nspec, double *jac, unsigned int useJacAnl)
Computes (analytical or numerical) Jacobian for the system $(T, Y_1, Y_2, \dots, Y_{N-1})$ based on T and Y 's.
- int [TCDND_getJacTYN](#) (double *scal, int Nspec, double *jac, unsigned int useJacAnl)
Computes (analytical or numerical) Jacobian for the system $(T, Y_1, Y_2, \dots, Y_N)$ based on T and Y 's.
- int [TC_getJacTYN](#) (double *scal, int Nspec, double *jac, unsigned int useJacAnl)
Computes (analytical or numerical) Jacobian for the system $(T, Y_1, Y_2, \dots, Y_N)$ based on T and Y 's.
- int [TC_getJacRPTYN](#) (double *scal, int Nspec, double *jac, unsigned int useJacAnl)
Computes (analytical) Jacobian for the system $(\rho, P, T, Y_1, Y_2, \dots, Y_N)$ based on T and Y 's.
- int [TC_getJacRPTYNanl](#) (double *scal, int Nspec, double *jac)
Computes analytical Jacobian for the system $(\rho, P, T, Y_1, Y_2, \dots, Y_N)$ based on T and Y 's.
- int [TC_getJacRPTYNnum](#) (double *scal, int Nspec, double *jac)
Computes numerical Jacobian for the system $(\rho, P, T, Y_1, Y_2, \dots, Y_N)$ based on T and Y 's.

5.4.1 Function Documentation

5.4.1.1 `int TC_getJacRPTYN (double * scal, int Nspec, double * jac, unsigned int useJacAnl)`

Computes (analytical) Jacobian for the system $(\rho, P, T, Y_1, Y_2, \dots, Y_N)$ based on T and Y's.

Parameters

<code>scal</code>	: array of $N_{spec} + 1$ doubles $(T, Y_1, Y_2, \dots, Y_{N_{spec}})$ temperature T [K], mass fractions Y []
<code>Nspec</code>	: no. of species N_{spec}
<code>useJacAnl</code>	: flag for Jacobian type (1-analytical, other values-numerical)

Returns

`jac` : Jacobian array

References `TC_errorMSG()`, `TC_getJacRPTYNanl()`, `TC_getJacRPTYNnum()`, and `TC_Nspec_`.

5.4.1.2 `int TC_getJacRPTYNanl (double * scal, int Nspec, double * jac)`

Computes analytical Jacobian for the system $(\rho, P, T, Y_1, Y_2, \dots, Y_N)$ based on T and Y's.

Parameters

<code>scal</code>	: array of $N_{spec} + 1$ doubles $(T, Y_1, Y_2, \dots, Y_{N_{spec}})$ temperature T [K], mass fractions Y []
<code>Nspec</code>	: no. of species N_{spec}

Returns

`jac` : Jacobian array

References `TC_errorMSG()`, `TC_getHspecMs()`, `TC_getMs2Cc()`, `TC_getMs2CpMixMs()`, `TC_getRhoMixMs()`, `TC_maxOrdPar_`, `TC_maxSpecInReac_`, `TC_nOrdReac_`, `TC_Nreac_`, `TC_nRealNuReac_`, `TC_Nspec_`, `TC_Nvars_`, `TC_Nvjac_`, and `TC_s-Mass_`.

Referenced by `TC_getJacRPTYN()`, `TC_getJacTYN()`, `TC_getJacTYNanl()`, `TC_getJacTYNm1()`, and `TC_getJacTYNm1anl()`.

5.4.1.3 int TC_getJacRPTYNnum (double * scal, int Nspec, double * jac)

Computes numerical Jacobian for the system $(\rho, P, T, Y_1, Y_2, \dots, Y_N)$ based on T and Y's.

Parameters

<i>scal</i>	: array of $N_{spec} + 1$ doubles $(T, Y_1, Y_2, \dots, Y_{N_{spec}})$ temperature T [K], mass fractions Y []
<i>Nspec</i>	: no. of species N_{spec}

Returns

jac : Jacobian array

References TC_errorMSG(), TC_getCpSpecMs(), TC_getHspecMs(), TC_getMs2Cc(), TC_getMs2CpMixMs(), TC_getRhoMixMs(), TC_getTXC2RRms(), TC_Nspec_, TC_Nvjac_, and TC_sMass_.

Referenced by TC_getJacRPTYN(), TC_getJacTYN(), and TC_getJacTYNm1().

5.4.1.4 int TC_getJacTYN (double * scal, int Nspec, double * jac, unsigned int useJacAnl)

Computes (analytical or numerical) Jacobian for the system $(T, Y_1, Y_2, \dots, Y_N)$ based on T and Y's.

Parameters

<i>scal</i>	: array of $N_{spec} + 1$ doubles $(T, Y_1, Y_2, \dots, Y_{N_{spec}})$ temperature T [K], mass fractions Y []
<i>Nspec</i>	: no. of species N_{spec}
<i>useJacAnl</i>	: flag for Jacobian type (1-analytical, other values-numerical)

Returns

jac : Jacobian array

References TC_errorMSG(), TC_getJacRPTYNanl(), TC_getJacRPTYNnum(), TC_getMs2Wmix(), TC_getRhoMixMs(), TC_Nspec_, TC_Nvars_, TC_Nvjac_, and TC_sMass_.

Referenced by TCDND_getJacTYN().

5.4.1.5 int TC_getJacTYNanl (double * scal, int Nspec, double * jac)

Computes analytical Jacobian for the system $(T, Y_1, Y_2, \dots, Y_N)$ based on T and Y's.

Parameters

<i>scal</i>	: array of $N_{spec} + 1$ doubles ($T, Y_1, Y_2, \dots, Y_{N_{spec}}$) temperature T [K], mass fractions Y []
<i>Nspec</i>	: no. of species N_{spec}

Returns

jac : Jacobian array

References TC_errorMSG(), TC_getJacRPTYNanl(), TC_getMs2Wmix(), TC_getRhoMixMs(), TC_Nspec_, TC_Nvars_, TC_Nvjac_, and TC_sMass_.

Referenced by TCDND_getJacTYNanl().

5.4.1.6 int TC_getJacTYNm1 (double * *scal*, int *Nspec*, double * *jac*, unsigned int *useJacAnl*)

Computes (analytical or numerical) Jacobian for the system ($T, Y_1, Y_2, \dots, Y_{N-1}$) based on T and Y's.

Parameters

<i>scal</i>	: array of $N_{spec} + 1$ doubles ($T, Y_1, Y_2, \dots, Y_{N_{spec}}$) temperature T [K], mass fractions Y []
<i>Nspec</i>	: no. of species N_{spec}
<i>useJacAnl</i>	: flag for Jacobian type (1-analytical, other values-numerical)

Returns

jac : Jacobian array

References TC_errorMSG(), TC_getJacRPTYNanl(), TC_getJacRPTYNnum(), TC_getMs2Wmix(), TC_getRhoMixMs(), TC_Nspec_, TC_Nvars_, TC_Nvjac_, and TC_sMass_.

Referenced by TCDND_getJacTYNm1().

5.4.1.7 int TC_getJacTYNm1anl (double * *scal*, int *Nspec*, double * *jac*)

Computes analytical Jacobian for the system ($T, Y_1, Y_2, \dots, Y_{N-1}$) based on T and Y's.

Parameters

<i>scal</i>	: array of $N_{spec} + 1$ doubles ($T, Y_1, Y_2, \dots, Y_{N_{spec}}$) temperature T [K], mass fractions Y []
<i>Nspec</i>	: no. of species N_{spec}

Returns

jac : Jacobian array

References TC_errorMSG(), TC_getJacRPTYNanI(), TC_getMs2Wmix(), TC_getRhoMixMs(), TC_Nspec_, TC_Nvars_, TC_Nvjac_, and TC_sMass_.

Referenced by TCDND_getJacTYNm1anI().

5.4.1.8 int TCDND_getJacTYN (double * scal, int Nspec, double * jac, unsigned int useJacAnI)

Computes (analytical or numerical) Jacobian for the system $(T, Y_1, Y_2, \dots, Y_N)$ based on T and Y's.

Parameters

scal	: array of $N_{spec} + 1$ doubles $(T, Y_1, Y_2, \dots, Y_{N_{spec}})$ temperature T [K], mass fractions Y []
Nspec	: no. of species N_{spec}
useJacAnI	: flag for Jacobian type (1-analytical, other values-numerical)

Returns

jac : Jacobian array

References TC_errorMSG(), TC_getJacTYN(), and TC_Nspec_.

5.4.1.9 int TCDND_getJacTYNanI (double * scal, int Nspec, double * jac)

Computes analytical Jacobian (dimensional/non-dimensional) for the system $(T, Y_1, Y_2, \dots, Y_N)$ based on T and Y's.

Parameters

scal	: array of $N_{spec} + 1$ doubles $(T, Y_1, Y_2, \dots, Y_{N_{spec}})$ temperature T [K], mass fractions Y []
Nspec	: no. of species N_{spec}

Returns

jac : Jacobian array

References TC_errorMSG(), TC_getJacTYNanI(), and TC_Nspec_.

5.4.1.10 `int TCDND_getJacTYNm1 (double * scal, int Nspec, double * jac, unsigned int useJacAnl)`

Computes (analytical or numerical) Jacobian (dimensional/non-dimensional) for the system $(T, Y_1, Y_2, \dots, Y_{N-1})$ based on temperature T and species mass fractions Y's.

Parameters

<i>scal</i>	: array of $N_{spec} + 1$ doubles $(T, Y_1, Y_2, \dots, Y_{N_{spec}})$ temperature T [K], mass fractions Y []
<i>Nspec</i>	: no. of species N_{spec}
<i>useJacAnl</i>	: flag for Jacobian type (1-analytical, other values-numerical)

Returns

jac : Jacobian array

References TC_errorMSG(), TC_getJacTYNm1(), and TC_Nspec_.

5.4.1.11 `int TCDND_getJacTYNm1anl (double * scal, int Nspec, double * jac)`

Computes analytical Jacobian (dimensional/non-dimensional) for the system $(T, Y_1, Y_2, \dots, Y_{N-1})$ based on temperature T and species mass fractions Y's.

Parameters

<i>scal</i>	: array of $N_{spec} + 1$ doubles $(T, Y_1, Y_2, \dots, Y_{N_{spec}})$ temperature T [K], mass fractions Y []
<i>Nspec</i>	: no. of species N_{spec}

Returns

jac : Jacobian array

References TC_errorMSG(), TC_getJacTYNm1anl(), and TC_Nspec_.

5.5 Source terms

Functions

- int [TCDND_getSrc](#) (double *scal, int Nvars, double *omega)

Returns dimensional/non-dimensional source term for

$$\frac{\partial T}{\partial t} = \omega_0, \frac{\partial Y_i}{\partial t} = \omega_i,$$

based on temperature T and species mass fractions Y 's.

- int [TC_getSrc](#) (double *scal, int Nvars, double *omega)

Returns source term for

$$\frac{\partial T}{\partial t} = \omega_0, \frac{\partial Y_i}{\partial t} = \omega_i,$$

based on temperature T and species mass fractions Y 's.

- int [TCDND_getSrcCons](#) (double *scal, int Nvars, double *omega)

Returns source term (dimensional/non-dimensional) for

$$\frac{\partial \rho}{\partial t} = \omega_0, \rho \frac{\partial Y_i}{\partial t} = \omega_i,$$

based on ρ and Y 's.

- int [TC_getSrcCons](#) (double *scal, int Nvars, double *omega)

Returns source term for

$$\frac{\partial \rho}{\partial t} = \omega_0, \rho \frac{\partial Y_i}{\partial t} = \omega_i,$$

based on ρ and Y 's.

5.5.1 Function Documentation

5.5.1.1 int TC_getSrc (double * scal, int Nvars, double * omega)

Returns source term for

$$\frac{\partial T}{\partial t} = \omega_0, \frac{\partial Y_i}{\partial t} = \omega_i,$$

based on temperature T and species mass fractions Y 's.

Parameters

<i>scal</i>	: array of Nspec+1 doubles ($T, Y_1, Y_2, \dots, Y_{N_{\text{spec}}}$) temperature T [K], mass fractions Y []
<i>Nvars</i>	: no. of variables $N_{\text{vars}} = N_{\text{spec}} + 1$

Returns

omega : array of $N_{\text{spec}} + 1$ source terms for temperature and species mass fractions: omega[0] : [(K/s)], omega[1... N_{spec}] : [(1/s)]

References TC_errorMSG(), TC_getHspecMs(), TC_getMs2CpMixMs(), TC_getRhoMixMs(), TC_getTY2RRms(), TC_Nspec_, and TC_Nvars_.

Referenced by TCDND_getSrc().

5.5.1.2 int TC_getSrcCons (double * scal, int Nvars, double * omega)

Returns source term for

$$\frac{\partial \rho}{\partial t} = \omega_0, \rho \frac{\partial Y_i}{\partial t} = \omega_i,$$

based on ρ and Y's.

Parameters

scal	: array of $N_{\text{spec}}+1$ doubles ($\rho, Y_1, Y_2, \dots, Y_N$) density [kg/m^3], mass fractions Y []
Nvars	: no. of variables = $N_{\text{spec}} + 1$

Returns

omega : array of $N_{\text{spec}} + 1$ source terms for density and species mf conservative formulation: omega[0] : [$\text{kg}/(\text{m}^3 \cdot \text{s})$], omega[1... N_{spec}] : [$\text{kg}/(\text{m}^3 \cdot \text{s})$]

References TC_errorMSG(), TC_getHspecMs(), TC_getMs2CpMixMs(), TC_getMs2Wmix(), TC_getTmixMs(), TC_getTY2RRml(), TC_Nspec_, TC_Nvars_, and TC_sMass_.

Referenced by TCDND_getSrcCons().

5.5.1.3 int TCDND_getSrc (double * scal, int Nvars, double * omega)

Returns dimensional/non-dimensional source term for

$$\frac{\partial T}{\partial t} = \omega_0, \frac{\partial Y_i}{\partial t} = \omega_i,$$

based on temperature T and species mass fractions Y's.

Parameters

scal	: array of $N_{\text{spec}}+1$ doubles ($T, Y_1, Y_2, \dots, Y_{N_{\text{spec}}}$) temperature T [K], mass fractions Y []
Nvars	: no. of variables $N_{\text{vars}} = N_{\text{spec}} + 1$

Returns

omega : array of N_{spec} +1 source terms (possibly normalized) for temperature and species mass fractions equations: omega[0] : [(K/s)], omega[1...N_{spec}] : [(1/s)]

References TC_errorMSG(), TC_getSrc(), TC_Nspec_, and TC_Nvars_.

5.5.1.4 int TCDND_getSrcCons (double * scal, int Nvars, double * omega)

Returns source term (dimensional/non-dimensional) for

$$\frac{\partial \rho}{\partial t} = \omega_0, \rho \frac{\partial Y_i}{\partial t} = \omega_i,$$

based on ρ and Y's.

Parameters

<i>scal</i>	: array of N _{spec} +1 doubles ($\rho, Y_1, Y_2, \dots, Y_N$) density [kg/m^3], mass fractions Y []
<i>Nvars</i>	: no. of variables = N _{spec} +1

Returns

omega : array of N_{spec} +1 source terms for density and species mf conservative formulation: omega[0] : [$kg/(m^3 \cdot s)$], omega[1...N_{spec}] : [$kg/(m^3 \cdot s)$]

References TC_errorMSG(), TC_getSrcCons(), TC_Nspec_, and TC_Nvars_.

5.6 Max. no. of parameters

Variables

- static int `TC_maxSpecInReac_`
Maximum number of species in a reaction.
- static int `TC_maxTbInReac_`
Max # of third-body efficiencies in a reaction.
- static int `TC_nNASAinter_`
of temperature regions for thermo fits
- static int `TC_nCpCoef_`
of polynomial coefficients for thermo fits
- static int `TC_nArhPar_`
of Arrhenius parameters
- static int `TC_nLtPar_`
of parameters for Landau-Teller reactions
- static int `TC_nFallPar_`
of parameters for pressure-dependent reactions
- static int `TC_nJanPar_`
of parameters for Jannev-Langer fits (JAN)
- static int `TC_maxOrdPar_`
of parameters for arbitrary order reactions
- static int `TC_nFit1Par_`
of parameters for FIT1 fits

5.7 No. of reactions

Variables

- static int [TC_Nreac_](#)
of reactions
- static int [TC_nRevReac_](#)
of reactions with REV given
- static int [TC_nFallReac_](#)
of pressure-dependent reactions
- static int [TC_nThbReac_](#)
of reactions using third-body efficiencies
- static int [TC_nLtReac_](#)
of Landau-Teller reactions
- static int [TC_nRltReac_](#)
of Landau-Teller reactions with RLT given
- static int [TC_nHvReac_](#)
of reactions with HV
- static int [TC_nJanReac_](#)
of reactions with JAN fits
- static int [TC_nFit1Reac_](#)
of reactions with FIT1 fits
- static int [TC_nExciReac_](#)
of reactions with EXCI
- static int [TC_nMomeReac_](#)
of reactions with MOME
- static int [TC_nXsmiReac_](#)
of reactions with XSMI
- static int [TC_nTdepReac_](#)
of reactions with TDEP
- static int [TC_nRealNuReac_](#)
of reactions with non-int stoichiometric coefficients
- static int [TC_nOrdReac_](#)
of reactions with arbitrary order

5.8 No. of species, variables, etc.

Variables

- static int `TC_Nvars_`
of variables = no. of species + 1
- static int `TC_Nvjac_`
of lines/cols in the Jacobian = no. of species + 3
- static int `TC_Nelem_`
of chemical elements
- static int `TC_Nspec_`
of species
- static int `TC_nIonSpec_`
of ion species
- static int `TC_electrIndx_`
Index of the electron species.
- static int `TC_nIonEspec_`
of ion species excluding the electron species

Chapter 6

Class Documentation

6.1 element Struct Reference

Element data.

```
#include <TC_kmodint.h>
```

Public Attributes

- char **name** [LENGTHOFLEMNAME]
- int **hasmass**
- double **mass**

6.1.1 Detailed Description

Element data.

The documentation for this struct was generated from the following file:

- TC_kmodint.h

6.2 elemtable Struct Reference

Entry in the table of periodic elements.

```
#include <TC_kmodint.h>
```

Public Attributes

- char **name** [LENGTHOFELEMENTNAME]
- double **mass**

6.2.1 Detailed Description

Entry in the table of periodic elements.

The documentation for this struct was generated from the following file:

- TC_kmodint.h

6.3 reaction Struct Reference

Reaction data.

```
#include <TC_kmodint.h>
```

Public Attributes

- int **isdup**
- int **isreal**
- int **isrev**
- int **isfall**
- int **specfall**
- int **isthrdb**
- int **nthbdb**
- int **iswl**
- int **isbal**
- int **iscomp**
- int **inreac**
- int **inprod**
- int **ismome**
- int **isxsmi**
- int **isford**
- int **isrord**
- int **islowset**
- int **ishighset**
- int **istroeset**
- int **issriset**
- int **isrevset**

- int **isltset**
- int **isrltset**
- int **ishvset**
- int **istdepset**
- int **isexciset**
- int **isjanset**
- int **isfit1set**
- int **spec** [2 *NSPECREACMAX]
- int **nuki** [2 *NSPECREACMAX]
- double **rnuki** [2 *NSPECREACMAX]
- double **arhenfor** [3]
- double **arhenrev** [3]
- int **ithrdb** [NTHRDBMAX]
- double **rthrdb** [NTHRDBMAX]
- double **fallpar** [8]
- double **ltpar** [2]
- double **rltpar** [2]
- double **hvpar**
- char **aunits** [4]
- char **eunits** [4]
- int **tdeppar**
- double **excipar**
- double **optfit** [9]
- int **arbspec** [4 *NSPECREACMAX]
- double **arbnuki** [4 *NSPECREACMAX]

6.3.1 Detailed Description

Reaction data.

The documentation for this struct was generated from the following file:

- TC_kmodint.h

6.4 species Struct Reference

Species data.

```
#include <TC_kmodint.h>
```

Public Attributes

- char **name** [LENGTHOFSPECNAME]
- int **hasthermo**
- int **hasmass**
- double **mass**
- int **charge**
- int **phase**
- int **numofelem**
- int **elemindx** [NUMBEROFELEMINSPEC]
- int **elemcontent** [NUMBEROFELEMINSPEC]
- double **nasapoltemp** [3]
- double **nasapolcoefs** [14]

6.4.1 Detailed Description

Species data.

The documentation for this struct was generated from the following file:

- TC_kmodint.h

Chapter 7

File Documentation

7.1 copyright.h File Reference

Copyright 2011 Sandia Corporation. Under the terms of Contract DE-AC04-94AL85000 with Sandia Corporation, the U.S. Government retains certain rights in this software.

7.1.1 Detailed Description

Copyright 2011 Sandia Corporation. Under the terms of Contract DE-AC04-94AL85000 with Sandia Corporation, the U.S. Government retains certain rights in this software. The software is released subject the terms of the GNU Lesser General Public License (<http://www.gnu.org/copyleft/lesser.html>) Please read the license information before downloading the software.

7.2 TC_chg.c File Reference

Functions for changing Arrhenius rate factors.

Functions

- int [TC_chgArhenFor](#) (int ireac, int ipos, double newval)
Change parameters for forward rate constants.
- int [TC_chgArhenForBack](#) (int ireac, int ipos)
Reverse changes for forward rate constants' parameters.
- int [TC_chgArhenRev](#) (int ireac, int ipos, double newval)

Change parameters for reverse rate constants.

- int [TC_chgArhenRevBack](#) (int ireac, int ipos)

Reverse changes for reverse rate constants' parameters.

7.2.1 Detailed Description

Functions for changing Arrhenius rate factors.

7.2.2 Function Documentation

7.2.2.1 int TC_chgArhenFor (int ireac, int ipos, double newval)

Change parameters for forward rate constants.

Parameters

<i>ireac</i>	: reaction index
<i>ipos</i>	: index of parameter to be changed (0) pre-exponential factor (1) temperature exponent, (2) activation energy
<i>newval</i>	: new parameter value

References TC_Nreac_.

7.2.2.2 int TC_chgArhenForBack (int ireac, int ipos)

Reverse changes for forward rate constants' parameters.

Parameters

<i>ireac</i>	: reaction index
<i>ipos</i>	: index of parameter to be changed (0) pre-exponential factor (1) temperature exponent, (2) activation energy

References TC_Nreac_.

7.2.2.3 int TC_chgArhenRev (int ireac, int ipos, double newval)

Change parameters for reverse rate constants.

Parameters

<i>ireac</i>	: reaction index
--------------	------------------

<i>ipos</i>	: index of parameter to be changed (0) pre-exponential factor (1) temperature exponent, (2) activation energy
<i>newval</i>	: new parameter value

References TC_nRevReac_.

7.2.2.4 int TC_chgArhenRevBack (int ireac, int ipos)

Reverse changes for reverse rate constants' parameters.

Parameters

<i>ireac</i>	: reaction index
<i>ipos</i>	: index of parameter to be changed (0) pre-exponential factor (1) temperature exponent, (2) activation energy

References TC_nRevReac_.

7.3 TC_defs.h File Reference

Definitions of variables names used by the library.

```
#include "copyright.h" #include <stdio.h> #include "TC_
params.h"
```

Defines

- #define **TMAX** 3500.0
- #define **TMIN** 290.0
- #define **DTMID** 10.0

Functions

- int [TC_kmodint_](#) (char *mechfile, int *lmech, char *thermofile, int *lthrm)
Kinetic model interpretor.
- int [TC_makeSpace](#) ()
Allocate internal work arrays for the library. Should not be called by external functions.
- int [TC_createTables](#) (double delT)
Create tables for temperature dependent terms. Should not be called by external functions.

- void **TC_errorMSG** (int msgID, char const *func, int var1, int var2)
Outputs error messages for the library, then exits the execution.
- void **TC_errorINI** (FILE *errfile, char const *msg)
Outputs error messages generated by TC_initChem.
- static double **fastIntPow** (double val, int exponent)
- int **TC_getRopsLocal** (double *scal)
- int **TC_getReacRates** (double *scal, int Nvars, double *omega)
Returns molar reaction rates, $\dot{\omega}_i$, based on T and molar concentrations XC's (semi-private function)
- int **TC_getgk** (double t1, double t_1, double tln)
- int **TC_getgkFcn** (double t1, double t_1, double tln)
- int **TC_getgkTab** (double t1)
- int **TC_getgkp** (double t1, double t_1, double tln)
- int **TC_getgkpFcn** (double t1, double t_1, double tln)
- int **TC_getgkpTab** (double t1)
- double **TC_getSumNuGk** (int i, double *gkLoc)
- double **TC_getSumRealNuGk** (int i, int ir, double *gkLoc)
- int **TC_get3rdBdyConc** (double *concX, double *concM)
- int **TC_getkForRev** (double t1, double t_1, double tln)
- int **TC_getkForRevFcn** (double t_1, double tln)
- int **TC_getkForRevTab** (double t1)
- int **TC_getkForRevP** (double t1, double t_1)
- int **TC_getkForRevPFcn** (double t_1)
- int **TC_getkForRevPTab** (double t1)
- int **TC_getRateofProg** (double *concX)
- int **TC_getRateofProgDer** (double *concX, int ireac, int ispec, double *qfr)
- int **TC_getCrnd** (double t1, double t_1, double tln, double *concX, double *concM)
- int **TC_getCrndDer** (int ireac, int *itbdy, int *ipfal, double t1, double t_1, double tln, double *concX, double *concM)
- int **TC_getCpSpecMsFcn** (double t, double *cpi)
- int **TC_getCpSpecMsTab** (double t1, double *cpi)
- int **TC_getCpSpecMIFcn** (double t, double *cpi)
- int **TC_getCpSpecMITab** (double t1, double *cpi)
- int **TC_getCpSpecMs1Fcn** (double t, int i, double *cpi)
- int **TC_getCpSpecMI1Fcn** (double t, int i, double *cpi)
- int **TC_getCpMixMsP** (double *scal, int Nvars, double *cpmix)
- int **TC_getCpSpecMsP** (double t, int Nspec, double *cpi)
- int **TC_getCpSpecMsPFcn** (double t, double *cpi)
- int **TC_getCpSpecMsPtab** (double t1, double *cpi)
- int **TC_getCpSpecMs1PFcn** (double t, int i, double *cpi)
- int **TC_getHspecMsFcn** (double t, double *hi)
- int **TC_getHspecMsTab** (double t1, double *hi)
- int **TC_getHspecMIFcn** (double t, double *hi)
- int **TC_getHspecMITab** (double t1, double *hi)

Variables

- static int [TC_maxSpecInReac_](#)
Maximum number of species in a reaction.
- static int [TC_maxTbInReac_](#)
Max # of third-body efficiencies in a reaction.
- static int [TC_nNASAinter_](#)
of temperature regions for thermo fits
- static int [TC_nCpCoef_](#)
of polynomial coefficients for thermo fits
- static int [TC_nArhPar_](#)
of Arrhenius parameters
- static int [TC_nLtPar_](#)
of parameters for Landau-Teller reactions
- static int [TC_nFallPar_](#)
of parameters for pressure-dependent reactions
- static int [TC_nJanPar_](#)
of parameters for Jannev-Langer fits (JAN)
- static int [TC_maxOrdPar_](#)
of parameters for arbitrary order reactions
- static int [TC_nFit1Par_](#)
of parameters for FIT1 fits
- static int [TC_Nvars_](#)
of variables = no. of species + 1
- static int [TC_Nvjac_](#)
of lines/cols in the Jacobian = no. of species + 3
- static int [TC_Nelem_](#)
of chemical elements
- static int [TC_Nspec_](#)
of species
- static int [TC_nIonSpec_](#)
of ion species
- static int [TC_electrIndx_](#)
Index of the electron species.
- static int [TC_nIonEspec_](#)
of ion species excluding the electron species
- static int [TC_Nreac_](#)
of reactions
- static int [TC_nRevReac_](#)
of reactions with REV given

- static int [TC_nFallReac_](#)
of pressure-dependent reactions
- static int [TC_nThbReac_](#)
of reactions using third-body efficiencies
- static int [TC_nLtReac_](#)
of Landau-Teller reactions
- static int [TC_nRltReac_](#)
of Landau-Teller reactions with RLT given
- static int [TC_nHvReac_](#)
of reactions with HV
- static int [TC_nJanReac_](#)
of reactions with JAN fits
- static int [TC_nFit1Reac_](#)
of reactions with FIT1 fits
- static int [TC_nExciReac_](#)
of reactions with EXCI
- static int [TC_nMomeReac_](#)
of reactions with MOME
- static int [TC_nXsmiReac_](#)
of reactions with XSML
- static int [TC_nTdepReac_](#)
of reactions with TDEP
- static int [TC_nRealNuReac_](#)
of reactions with non-int stoichiometric coefficients
- static int [TC_nOrdReac_](#)
of reactions with arbitrary order
- static char * [TC_sNames_](#)
*species names, name of species i stored at LENGTHOFSPECNAME*i*
- static char * [TC_eNames_](#)
*species names, name of species i stored at LENGTHOFELEMENTNAME*i*
- static double * [TC_sMass_](#)
array of species molar masses
- static double * [TC_eMass_](#)
array of element molar masses
- static int * [TC_elemcount_](#)
*no. of atoms of element j in each species i at (i*TC_Nelem_+j)*
- static int * [TC_sCharge_](#)
species electrical charges
- static int * [TC_sTfit_](#)

no. of temperature fits for thermodynamic properties

- static int * [TC_sPhase_](#)
species phase id
- static double * [TC_Tlo_](#)
- static double * [TC_Tmi_](#)
- static double * [TC_Thi_](#)
- static double * [TC_cppol_](#)
- static int * [TC_sNion_](#)
- static int * [TC_isRev_](#)
- static int * [TC_reacNrp_](#)
- static int * [TC_reacNreac_](#)
- static int * [TC_reacNprod_](#)
- static int * [TC_reacScoef_](#)
- static int * [TC_reacNuki_](#)
- static int * [TC_reacSidx_](#)
- static double * [TC_reacNukiDbl_](#)
- static int * [TC_reacRnu_](#)
- static double * [TC_reacRealNuki_](#)
- static int * [TC_reacRev_](#)
- static double * [TC_reacArhenFor_](#)
- static double * [TC_reacArhenRev_](#)
- static int * [TC_reacPfal_](#)
- static int * [TC_reacPtype_](#)
- static int * [TC_reacPlphi_](#)
- static int * [TC_reacPspec_](#)
- static double * [TC_reacPpar_](#)
- static int * [TC_reactbdy_](#)
- static int * [TC_reactbno_](#)
- static int * [TC_specTbdldx_](#)
- static double * [TC_specTbdEff_](#)
- static int * [TC_reacAOrd_](#)
- static int * [TC_specAOidx_](#)
- static double * [TC_specAOval_](#)
- static int * [TC_reachvldx_](#)
- static double * [TC_reachvpar_](#)
- static double * [TC_kfor](#)
- static double * [TC_krev](#)
- static double * [TC_kforP](#)
- static double * [TC_krevP](#)
- static double * [TC_ropFor](#)
- static double * [TC_ropRev](#)
- static double * [TC_rop](#)

- static double * **TC_cpks**
- static double * **TC_hks**
- static double * **TC_gk**
- static double * **TC_gkp**
- static double * **TC_PrDer**
- static double * **TC_Crnd**
- static double * **TC_CrndDer**
- static double * **TC_dFfac**
- static double * **TC_omg**
- static double * **TC_omgP**
- static double * **TC_jacFull**
- static double * **TC_Xconc**
- static double * **TC_scalIn**
- static double * **TC_Mconc**
- static int * **TC_sigNu**
- static int * **TC_NuIJ**
- static double * **TC_sigRealNu**
- static double * **TC_RealNuIJ**
- static double * **qfr**
- static double **TC_reltol**
- static double **TC_abstol**
- static double **TC_rhref_**
- static double **TC_pref_**
- static double **TC_Tref_**
- static double **TC_Wref_**
- static double **TC_Daref_**
- static double **TC_omgref_**
- static double **TC_cpref_**
- static double **TC_href_**
- static double **TC_timref_**
- static int **TC_isInit_**
- static int **TC_tab_**
- static int **TC_nonDim_**
- static int **TC_RVset_**
- static int **TC_Ntab_**
- static double **TC_deIT_**
- static double **TC_odeIT_**
- static double * **TC_cptab**
- static double * **TC_cpPtab**
- static double * **TC_htab**
- static double * **TC_gktab**
- static double * **TC_gkPtab**
- static double * **TC_kfortab**

- static double * **TC_krevtab**
- static double * **TC_kforPtab**
- static double * **TC_krevPtab**
- static double **TC_pressure_**
- static double **TC_prescgs_**
- static double **TC_Runiv_**
- static double **TC_Rcal_**
- static double **TC_Rcgs_**
- static int **TC_ArhenForChg_**
- static int **TC_ArhenRevChg_**
- static double * **TC_reacArhenForSave_**
- static double * **TC_reacArhenRevSave_**
- static double * **TC_kc_coeff**
- static int **TC_initRemoved**
- static int **TC_NreacBackup_**
- static int **TC_NrevBackup_**
- static int **TC_NfalBackup_**
- static int **TC_NthbBackup_**
- static int **TC_nRealNuReacBackup_**
- static int **TC_nOrdReacBackup_**
- static int * **TC_isRevBackup_**
- static int * **TC_reacNrpBackup_**
- static int * **TC_reacNreacBackup_**
- static int * **TC_reacNprodBackup_**
- static int * **TC_reacNukiBackup_**
- static int * **TC_reacSidxBackup_**
- static int * **TC_reacScoefBackup_**
- static double * **TC_reacArhenForBackup_**
- static double * **TC_reacArhenRevBackup_**
- static double * **TC_reacNukiDbIBackup_**
- static int * **TC_reacPfalBackup_**
- static int * **TC_reacPtypeBackup_**
- static int * **TC_reacPlohiBackup_**
- static int * **TC_reacPspecBackup_**
- static int * **TC_reacTbdyBackup_**
- static int * **TC_reacTbnoBackup_**
- static int * **TC_specTbdIdxBackup_**
- static double * **TC_reacPparBackup_**
- static double * **TC_specTbdEffBackup_**
- static int * **TC_reacRnuBackup_**
- static int * **TC_reacAOrdBackup_**
- static int * **TC_specAOidxBackup_**
- static double * **TC_reacRealNukiBackup_**

- static double * **TC_sigRealNuBackup**
- static double * **TC_RealNuJBackup**
- static double * **TC_specAOvalBackup_**
- static double * **TC_kc_coeffBackup**
- static int * **TC_sigNuBackup**
- static int * **TC_NuJBackup**

7.3.1 Detailed Description

Definitions of variables names used by the library. #

7.3.2 Function Documentation

7.3.2.1 void TC_errorINI (FILE * *errfile*, char const * *msg*)

Outputs error messages generated by TC_initChem.

Parameters

<i>errfile</i>	: file pointer for output
<i>msg</i>	: error message

Referenced by TC_initChem().

7.3.2.2 void TC_errorMSG (int *msgID*, char const * *func*, int *var1*, int *var2*)

Outputs error messages for the library, then exits the execution.

Parameters

<i>msgID</i>	: message ID
<i>func</i>	: name of function calling this error function
<i>var1</i>	: value #1 to be printed in the message
<i>var2</i>	: value #2 to be printed in the message

Referenced by TC_getCpSpecMI(), TC_getCpSpecMs(), TC_getHspecMI(), TC_getHspecMs(), TC_getJacRPTYN(), TC_getJacRPTYNanl(), TC_getJacRPTYNnum(), TC_getJacTYN(), TC_getJacTYNanl(), TC_getJacTYNm1(), TC_getJacTYNm1anl(), TC_getMI2CpMixMI(), TC_getMI2HmixMI(), TC_getMs2CpMixMs(), TC_getMs2CvMixMs(), TC_getMs2HmixMs(), TC_getRfrb(), TC_getRhoMixMI(), TC_getRhoMixMs(), TC_getRops(), TC_getSrc(), TC_getSrcCons(), TC_getStoiCoef(), TC_getStoiCoefReac(), TC_getTmixMI(), TC_getTmixMs(), TC_getTXC2RRml(), TC_getTXC2RRms(), TC_getTY2RRml(), TC_getTY2RRms(), TCDND_getCpSpecMI(), TCDND_getCp-

SpecMs(), TCDND_getHspecMI(), TCDND_getHspecMs(), TCDND_getJacTYN(), TCDND_getJacTYNanl(), TCDND_getJacTYNm1(), TCDND_getJacTYNm1anl(), TCDND_getMI2CpMixMI(), TCDND_getMI2HmixMI(), TCDND_getMs2CpMixMs(), TCDND_getMs2CvMixMs(), TCDND_getMs2HmixMs(), TCDND_getRhoMixMI(), TCDND_getRhoMixMs(), TCDND_getSrc(), TCDND_getSrcCons(), TCDND_getTmixMI(), TCDND_getTmixMs(), TCDND_getTXC2RRml(), TCDND_getTXC2RRms(), TCDND_getTY2RRml(), and TCDND_getTY2RRms().

7.3.2.3 int TC_getReacRates (double * scal, int Nvars, double * omega)

Returns molar reaction rates, $\dot{\omega}_i$, based on T and molar concentrations XC's (semi-private function)

Parameters

scal	: array of $N_{spec} + 1$ doubles (($T, XC_1, XC_2, \dots, XC_N$): temperature T [K], molar concentrations XC [$kmol/m^3$])
Nvars	: no. of variables $N_{vars} = N_{spec} + 1$

Returns

omega : array of N_{spec} molar reaction rates $\dot{\omega}_i$ [$kmol/(m^3 \cdot s)$]

References TC_maxSpecInReac_, TC_Nreac_, TC_nRealNuReac_, and TC_Nspec_.

Referenced by TC_getTXC2RRml(), TC_getTXC2RRms(), TC_getTY2RRml(), and TC_getTY2RRms().

7.4 TC_init.c File Reference

Initialize chemical library.

Functions

- int [TC_initChem](#) (char *mechfile, char *thermofile, int tab, double delT)
Initialize library.
- void [TC_setRefVal](#) (double rhoref, double pref, double Tref, double Wref, double Daref, double omgref, double cpref, double href, double timref)
Set reference values to the library.
- void [TC_setNonDim](#) ()
Set library to function in non-dimensional mode.
- void [TC_setDim](#) ()

Set library to function in dimensional mode (default)

- void [TC_setThermoPres](#) (double pressure)
Send thermodynamic pressure to the library.
- int [TC_makeSpace](#) ()
Allocate internal work arrays for the library. Should not be called by external functions.
- int [TC_createTables](#) (double delT)
Create tables for temperature dependent terms. Should not be called by external functions.
- void [TC_errorMSG](#) (int msgID, char const *func, int var1, int var2)
Outputs error messages for the library, then exits the execution.
- void [TC_errorINI](#) (FILE *errfile, char const *msg)
Outputs error messages generated by TC_initChem.

7.4.1 Detailed Description

Initialize chemical library.

7.4.2 Function Documentation

7.4.2.1 void TC_errorINI (FILE * errfile, char const * msg)

Outputs error messages generated by TC_initChem.

Parameters

<i>errfile</i>	: file pointer for output
<i>msg</i>	: error message

Referenced by TC_initChem().

7.4.2.2 void TC_errorMSG (int msgID, char const * func, int var1, int var2)

Outputs error messages for the library, then exits the execution.

Parameters

<i>msgID</i>	: message ID
<i>func</i>	: name of function calling this error function
<i>var1</i>	: value #1 to be printed in the message
<i>var2</i>	: value #2 to be printed in the message

Referenced by TC_getCpSpecMI(), TC_getCpSpecMs(), TC_getHspecMI(), TC_getHspecMs(), TC_getJacRPTYN(), TC_getJacRPTYNanl(), TC_getJacRPTYNnum(), TC_getJacTYN(), TC_getJacTYNanl(), TC_getJacTYNm1(), TC_getJacTYNm1anl(), TC_getMI2CpMixMI(), TC_getMI2HmixMI(), TC_getMs2CpMixMs(), TC_getMs2CvMixMs(), TC_getMs2HmixMs(), TC_getRfrb(), TC_getRhoMixMI(), TC_getRhoMixMs(), TC_getRops(), TC_getSrc(), TC_getSrcCons(), TC_getStoiCoef(), TC_getStoiCoefReac(), TC_getTmixMI(), TC_getTmixMs(), TC_getTxC2RRml(), TC_getTxC2RRms(), TC_getTY2RRml(), TC_getTY2RRms(), TCDND_getCpSpecMI(), TCDND_getCpSpecMs(), TCDND_getHspecMI(), TCDND_getHspecMs(), TCDND_getJacTYN(), TCDND_getJacTYNanl(), TCDND_getJacTYNm1(), TCDND_getJacTYNm1anl(), TCDND_getMI2CpMixMI(), TCDND_getMI2HmixMI(), TCDND_getMs2CpMixMs(), TCDND_getMs2CvMixMs(), TCDND_getMs2HmixMs(), TCDND_getRhoMixMI(), TCDND_getRhoMixMs(), TCDND_getSrc(), TCDND_getSrcCons(), TCDND_getTmixMI(), TCDND_getTmixMs(), TCDND_getTxC2RRml(), TCDND_getTxC2RRms(), TCDND_getTY2RRml(), and TCDND_getTY2RRms().

7.5 TC_interface.h File Reference

Header file to be included in user's code. Contains function definitions.

```
#include "copyright.h"
```

Defines

- #define **TCSMALL** 1.e-30

Functions

- int [TC_chgArhenFor](#) (int ireac, int ipos, double newval)
Change parameters for forward rate constants.
- int [TC_chgArhenRev](#) (int ireac, int ipos, double newval)
Change parameters for reverse rate constants.
- int [TC_chgArhenForBack](#) (int ireac, int ipos)
Reverse changes for forward rate constants' parameters.
- int [TC_chgArhenRevBack](#) (int ireac, int ipos)
Reverse changes for reverse rate constants' parameters.
- int [TC_getArhenRev](#) (int ireac, int ipos, double *val)
Return current value of the Arrhenius parameters for reverse rate constants. Return -1 if no data available, otherwise return 0 and store value in val.
- int [TC_getArhenFor](#) (int ireac, int ipos, double *val)

Return current value of the Arrhenius parameters for forward rate constants. Return -1 if no data available, otherwise return 0 and store value in val.

- void [TC_reset](#) ()

Frees all memory and sets variables to 0 so that TC_initChem can be called again without a memory leak. Not designed for use with tables.
- void [TC_removeReaction](#) (int *reacArr, int numRemoveReacs, int revOnly)

Removes a reaction from the mechanism. Not designed for use with tables.
- void [TC_restoreReactions](#) ()

Restores reaction mechanism to original state. Any changes from TC_chgArhenFor and TC_chgArhenRev are also reset.
- int [TC_initChem](#) (char *mechfile, char *thermofile, int tab, double delT)

Initialize library.
- void [TC_setRefVal](#) (double rhoref, double pref, double Tref, double Wref, double Daref, double omgref, double cpref, double href, double timref)

Set reference values to the library.
- void [TC_setNonDim](#) ()

Set library to function in non-dimensional mode.
- void [TC_setDim](#) ()

Set library to function in dimensional mode (default)
- void [TC_setThermoPres](#) (double pressure)

Send thermodynamic pressure to the library.
- int [TC_getNspec](#) ()

Returns no. of species N_{spec}
- int [TC_getNelem](#) ()

Returns no. of elements N_{elem}
- int [TC_getNvars](#) ()

Returns no. of variables ($N_{spec} + 1$)
- double [TC_getThermoPres](#) ()
- int [TC_getSnames](#) (int Nspec, char *snames)

Returns species names.
- int [TC_getSpos](#) (const char *sname, const int slen)

Returns position a species in the list of species.
- int [TC_getSmass](#) (int Nspec, double *Wi)

Returns species molar weights.
- int [TC_getSnameLen](#) ()

Returns length of species names.
- int [TC_getNreac](#) ()

Returns number of reactions N_{reac} .
- int [TC_getStoiCoef](#) (int Nspec, int Nreac, double *stoicoef)

Returns stoichiometric coefficients' matrix. The stoichiometric coefficient for species "j" in reaction "i" is stored at position $i \cdot N_{spec} + j$. It assumes that stoicoef was dimensioned to at least $N_{react} \cdot N_{spec}$.

- int [TC_getStoiCoefReac](#) (int Nspec, int Nreact, int ireac, int idx, double *stoicoef)

Returns stoichiometric coefficients' array for reaction 'ireac' for either reactants (idx=0) or products (idx=1) The stoichiometric coefficient for species "j" in reaction "ireac" is stored at position j. It assumes that stoicoef was dimensioned to at least N_{spec} .

- int [TCDND_getTY2RRml](#) (double *scal, int Nvars, double *omega)

Returns non-dimensional molar reaction rates, $\dot{\omega}_i \cdot t_{ref} \frac{W_{ref}}{\rho_{ref}}$, based on T and Y's.

- int [TC_getTY2RRml](#) (double *scal, int Nvars, double *omega)

Returns molar reaction rates, $\dot{\omega}_i$, based on T and Y's.

- int [TCDND_getTY2RRms](#) (double *scal, int Nvars, double *omega)

Returns non-dimensional mass reaction rates based on T and Y's.

- int [TC_getTY2RRms](#) (double *scal, int Nvars, double *omega)

Returns mass reaction rates based on T and Y's.

- int [TCDND_getTxC2RRml](#) (double *scal, int Nvars, double *omega)

Returns non-dimensional molar reaction rates based on temperature T and molar concentrations XC.

- int [TC_getTxC2RRml](#) (double *scal, int Nvars, double *omega)

Returns molar reaction rates based on temperature T and molar concentrations XC.

- int [TCDND_getTxC2RRms](#) (double *scal, int Nvars, double *omega)

Returns non-dimensional mass reaction rates based on T and molar concentrations.

- int [TC_getTxC2RRms](#) (double *scal, int Nvars, double *omega)

Returns mass reaction rates based on T and molar concentrations.

- int [TC_getRops](#) (double *scal, int Nvars, double *datarop)

Returns rate-of-progress variables based on temperature T and species mass fractions Y's.

- int [TC_getRfrb](#) (double *scal, int Nvars, double *datarop)

Returns forward and reverse rate-of-progress variables based on T and Y's.

- int [TCDND_getRhoMixMs](#) (double *scal, int Nvars, double *rhomix)

Computes density based on temperature and species mass fractions using the equation of state. Input temperature is normalized, output density also normalized before exit.

- int [TC_getRhoMixMs](#) (double *scal, int Nvars, double *rhomix)

Computes density based on temperature and species mass fractions using the equation of state.

- int [TCDND_getRhoMixMI](#) (double *scal, int Nvars, double *rhomix)

Computes density based on temperature and species mole fractions using the equation of state. Input temperature is normalized, output density also normalized before exit.

- int [TC_getRhoMixMI](#) (double *scal, int Nvars, double *rhomix)

Computes density based on temperature and species mole fractions using the equation of state.

- int [TCDND_getTmixMs](#) (double *scal, int Nvars, double *Tmix)
Computes temperature based on density and species mass fractions using the equation of state. Input density is normalized, output temperature also normalized before exit.
- int [TC_getTmixMs](#) (double *scal, int Nvars, double *Tmix)
Computes temperature based on density and species mass fractions using the equation of state.
- int [TCDND_getTmixMI](#) (double *scal, int Nvars, double *Tmix)
Computes temperature based on density and species mole fractions using the equation of state. Input density is normalized, output temperature also normalized before exit.
- int [TC_getTmixMI](#) (double *scal, int Nvars, double *Tmix)
Computes temperature based on density and species mole fractions using the equation of state.
- int [TCDND_getMs2CpMixMs](#) (double *scal, int Nvars, double *cpmix)
Computes mixture specific heat at constant pressure based on temperature and species mass fractions. Input temperature is normalized, output specific heat is also normalized before exit.
- int [TC_getMs2CpMixMs](#) (double *scal, int Nvars, double *cpmix)
Computes mixture specific heat at constant pressure based on temperature and species mass fractions.
- int [TCDND_getMs2CvMixMs](#) (double *scal, int Nvars, double *cvmix)
Computes mixture specific heat at constant volume based on temperature and species mass fractions. Input temperature is normalized, output specific heat is also normalized before exit.
- int [TC_getMs2CvMixMs](#) (double *scal, int Nvars, double *cvmix)
Computes mixture specific heat at constant volume based on temperature and species mass fractions.
- int [TCDND_getMI2CpMixMI](#) (double *scal, int Nvars, double *cpmix)
Computes mixture heat capacity at constant pressure based on temperature and species mole fractions. Input temperature is normalized, output specific heat is also normalized before exit.
- int [TC_getMI2CpMixMI](#) (double *scal, int Nvars, double *cpmix)
Computes mixture specific heat at constant pressure based on temperature and species mole fractions.
- int [TCDND_getCpSpecMs](#) (double t, int Nspec, double *cpi)
Computes species specific heat at constant pressure based on temperature. Input temperature is normalized, output specific heats are also normalized before exit.
- int [TC_getCpSpecMs](#) (double t, int Nspec, double *cpi)
Computes species specific heat at constant pressure based on temperature.
- int [TCDND_getCpSpecMI](#) (double t, int Nspec, double *cpi)

Computes species heat capacities at constant pressure based on temperature. Input temperature is normalized, output heat capacities are also normalized before exit.

- int [TC_getCpSpecMI](#) (double t, int Nspec, double *cpi)

Computes species heat capacities at constant pressure based on temperature.

- int [TCDND_getMs2HmixMs](#) (double *scal, int Nvars, double *hmix)

Computes mixture specific enthalpy based on temperature and species mass fractions. Input temperature is normalized, output enthalpy is normalized before exit.

- int [TC_getMs2HmixMs](#) (double *scal, int Nvars, double *hmix)

Computes mixture specific enthalpy based on temperature and species mass fractions.

- int [TCDND_getMI2HmixMI](#) (double *scal, int Nvars, double *hmix)

Computes mixture molar enthalpy based on temperature and species mole fractions. Input temperature is normalized, output enthalpy is normalized before exit.

- int [TC_getMI2HmixMI](#) (double *scal, int Nvars, double *hmix)

Computes mixture molar enthalpy based on temperature and species mole fractions.

- int [TCDND_getHspecMs](#) (double t, int Nspec, double *hi)

Computes species specific enthalpies based on temperature. Input temperature is normalized, output enthalpies are also normalized before exit.

- int [TC_getHspecMs](#) (double t, int Nspec, double *hi)

Computes species specific enthalpies based on temperature.

- int [TCDND_getHspecMI](#) (double t, int Nspec, double *hi)

Computes species molar enthalpies based on temperature. Input temperature is normalized, output enthalpies are also normalized before exit.

- int [TC_getHspecMI](#) (double t, int Nspec, double *hi)

Computes species molar enthalpies based on temperature.

- int [TCDND_getMs2Cc](#) (double *scal, int Nvars, double *concX)

Computes molar concentrations based on temperature and species mass fractions. Input temperature is normalized, output concentrations are also normalized before exit.

$$\overline{[X_k]} = [X_k] \cdot \frac{W_{ref}}{\rho_{ref}} = Y_k \cdot \frac{\rho}{W_k} \cdot \frac{W_{ref}}{\rho_{ref}}$$

- int [TC_getMs2Cc](#) (double *scal, int Nvars, double *concX)

Computes molar concentrations based on temperature and species mass fractions.

$$[X_k] = Y_k \cdot \rho / W_k$$

- int [TCDND_getMI2Ms](#) (double *Xspec, int Nspec, double *Yspec)

Transforms mole fractions to mass fractions (same as [TC_getMI2Ms\(\)](#)).

$$Y_k = X_k \cdot W_k / W_{mix}$$

- int [TC_getMI2Ms](#) (double *Xspec, int Nspec, double *Yspec)

Transforms mole fractions to mass fractions.

$$Y_k = X_k \cdot W_k / W_{mix}$$

- int [TCDND_getMs2MI](#) (double *Yspec, int Nspec, double *Xspec)

Transforms mass fractions to mole fractions (same as [TC_getMs2MI\(\)](#))

$$X_k = Y_k \cdot W_{mix} / W_k$$

- int [TC_getMs2MI](#) (double *Yspec, int Nspec, double *Xspec)

Transforms mass fractions to mole fractions.

$$X_k = Y_k \cdot W_{mix} / W_k$$

- int [TCDND_getMs2Wmix](#) (double *Yspec, int Nspec, double *Wmix)

Computes mixture molecular weight based on species mass fractions. The molecular weight ([kg/kmol]=[g/mol]) is normalized before output.

$$\bar{W}_{mix} = \frac{W_{mix}}{W_{ref}} = \frac{1}{W_{ref}} \left(\sum_{k=1}^{N_{spec}} Y_k / W_k \right)^{-1}$$

- int [TC_getMs2Wmix](#) (double *Yspec, int Nspec, double *Wmix)

Computes mixture molecular weight based on species mass fractions.

$$W_{mix} = \left(\sum_{k=1}^{N_{spec}} Y_k / W_k \right)^{-1}$$

- int [TCDND_getMI2Wmix](#) (double *Xspec, int Nspec, double *Wmix)

Computes mixture molecular weight based on species mole fractions. The molecular weight ([kg/kmol]=[g/mol]) is normalized before output.

$$\bar{W}_{mix} = \frac{W_{mix}}{W_{ref}} = \frac{1}{W_{ref}} \sum_{k=1}^{N_{spec}} X_k W_k$$

- int [TC_getMI2Wmix](#) (double *Xspec, int Nspec, double *Wmix)

Computes mixture molecular weight based on species mole fractions.

$$W_{mix} = \sum_{k=1}^{N_{spec}} X_k W_k$$

- int [TCDND_getSrc](#) (double *scal, int Nvars, double *omega)

Returns dimensional/non-dimensional source term for

$$\frac{\partial T}{\partial t} = \omega_0, \frac{\partial Y_i}{\partial t} = \omega_i,$$

based on temperature T and species mass fractions Y 's.

- int [TC_getSrc](#) (double *scal, int Nvars, double *omega)

Returns source term for

$$\frac{\partial T}{\partial t} = \omega_0, \frac{\partial Y_i}{\partial t} = \omega_i,$$

based on temperature T and species mass fractions Y 's.

- int [TCDND_getSrcCons](#) (double *scal, int Nvars, double *omega)

Returns source term (dimensional/non-dimensional) for

$$\frac{\partial \rho}{\partial t} = \omega_0, \rho \frac{\partial Y_i}{\partial t} = \omega_i,$$

based on ρ and Y 's.

- int [TC_getSrcCons](#) (double *scal, int Nvars, double *omega)

Returns source term for

$$\frac{\partial \rho}{\partial t} = \omega_0, \rho \frac{\partial Y_i}{\partial t} = \omega_i,$$

based on ρ and Y 's.

- int [TCDND_getJacTYNm1anl](#) (double *scal, int Nspec, double *jac)

Computes analytical Jacobian (dimensional/non-dimensional) for the system $(T, Y_1, Y_2, \dots, Y_{N-1})$ based on temperature T and species mass fractions Y 's.

- int [TC_getJacTYNm1anl](#) (double *scal, int Nspec, double *jac)

Computes analytical Jacobian for the system $(T, Y_1, Y_2, \dots, Y_{N-1})$ based on T and Y 's.

- int [TCDND_getJacTYNm1](#) (double *scal, int Nspec, double *jac, unsigned int useJacAnl)

Computes (analytical or numerical) Jacobian (dimensional/non-dimensional) for the system $(T, Y_1, Y_2, \dots, Y_{N-1})$ based on temperature T and species mass fractions Y 's.

- int [TC_getJacTYNm1](#) (double *scal, int Nspec, double *jac, unsigned int useJacAnl)

Computes (analytical or numerical) Jacobian for the system $(T, Y_1, Y_2, \dots, Y_{N-1})$ based on T and Y 's.

- int [TCDND_getJacTYNanl](#) (double *scal, int Nspec, double *jac)

Computes analytical Jacobian (dimensional/non-dimensional) for the system $(T, Y_1, Y_2, \dots, Y_N)$ based on T and Y 's.

- int [TC_getJacTYNanl](#) (double *scal, int Nspec, double *jac)

Computes analytical Jacobian for the system $(T, Y_1, Y_2, \dots, Y_N)$ based on T and Y 's.

- int [TCDND_getJacTYN](#) (double *scal, int Nspec, double *jac, unsigned int useJacAnl)

Computes (analytical or numerical) Jacobian for the system $(T, Y_1, Y_2, \dots, Y_N)$ based on T and Y 's.

- int [TC_getJacTYN](#) (double *scal, int Nspec, double *jac, unsigned int useJacAnl)

Computes (analytical or numerical) Jacobian for the system $(T, Y_1, Y_2, \dots, Y_N)$ based on T and Y 's.

- int [TC_getJacRPTYN](#) (double *scal, int Nspec, double *jac, unsigned int useJac-Anl)

Computes (analytical) Jacobian for the system $(\rho, P, T, Y_1, Y_2, \dots, Y_N)$ based on T and Y 's.

- int [TC_getJacRPTYNanl](#) (double *scal, int Nspec, double *jac)

Computes analytical Jacobian for the system $(\rho, P, T, Y_1, Y_2, \dots, Y_N)$ based on T and Y 's.

- int [TC_getJacRPTYNnum](#) (double *scal, int Nspec, double *jac)

Computes numerical Jacobian for the system $(\rho, P, T, Y_1, Y_2, \dots, Y_N)$ based on T and Y 's.

7.5.1 Detailed Description

Header file to be included in user's code. Contains function definitions.

7.5.2 Function Documentation

7.5.2.1 int TC_chgArhenFor (int ireac, int ipos, double newval)

Change parameters for forward rate constants.

Parameters

<i>ireac</i>	: reaction index
<i>ipos</i>	: index of parameter to be changed (0) pre-exponential factor (1) temperature exponent, (2) activation energy
<i>newval</i>	: new parameter value

References TC_Nreac_.

7.5.2.2 int TC_chgArhenForBack (int ireac, int ipos)

Reverse changes for forward rate constants' parameters.

Parameters

<i>ireac</i>	: reaction index
<i>ipos</i>	: index of parameter to be changed (0) pre-exponential factor (1) temperature exponent, (2) activation energy

References TC_Nreac_.

7.5.2.3 int TC_chgArhenRev (int *ireac*, int *ipos*, double *newval*)

Change parameters for reverse rate constants.

Parameters

<i>ireac</i>	: reaction index
<i>ipos</i>	: index of parameter to be changed (0) pre-exponential factor (1) temperature exponent, (2) activation energy
<i>newval</i>	: new parameter value

References TC_nRevReac_.

7.5.2.4 int TC_chgArhenRevBack (int *ireac*, int *ipos*)

Reverse changes for reverse rate constants' parameters.

Parameters

<i>ireac</i>	: reaction index
<i>ipos</i>	: index of parameter to be changed (0) pre-exponential factor (1) temperature exponent, (2) activation energy

References TC_nRevReac_.

7.5.2.5 int TC_getArhenFor (int *ireac*, int *ipos*, double * *val*)

Return current value of the Arrhenius parameters for forward rate constants. Return -1 if no data available, otherwise return 0 and store value in *val*.

Parameters

<i>ireac</i>	: reaction index
<i>ipos</i>	: index of Arrhenius parameter (0) pre-exponential factor (1) temperature exponent, (2) activation energy
* <i>val</i>	: value of Arrhenius parameter

References TC_Nreac_.

7.5.2.6 int TC_getArhenRev (int *ireac*, int *ipos*, double * *val*)

Return current value of the Arrhenius parameters for reverse rate constants. Return -1 if no data available, otherwise return 0 and store value in *val*.

Parameters

<i>ireac</i>	: reaction index
<i>ipos</i>	: index of of Arrhenius parameter (0) pre-exponential factor (1) temperature exponent, (2) activation energy
<i>*val</i>	: value of Arrhenius parameter

References TC_nRevReac_.

7.5.2.7 int TC_getMI2Ms (double * *Xspec*, int *Nspec*, double * *Yspec*)

Transforms mole fractions to mass fractions.

$$Y_k = X_k \cdot W_k / W_{mix}$$

Parameters

<i>Xspec</i>	: array of <i>Nspec</i> mole fractions
<i>Nspec</i>	: no. of species

Returns

Yspec : array of *Nspec* mass fractions

References TC_getMI2Wmix(), TC_Nspec_, and TC_sMass_.

Referenced by TCDND_getMI2Ms().

7.5.2.8 int TC_getMI2Wmix (double * *Xspec*, int *Nspec*, double * *Wmix*)

Computes mixture molecular weight based on species mole fractions.

$$W_{mix} = \sum_{k=1}^{N_{spec}} X_k W_k$$

Parameters

<i>Xspec</i>	: array of <i>Nspec</i> mole fractions
<i>Nspec</i>	: no. of species

Returns

Wmix : pointer to mixture molecular weight [kg/kmol]=[g/mol]

References TC_Nspec_, and TC_sMass_.

Referenced by TC_getMI2Ms().

7.5.2.9 int TC_getMs2Cc (double * scal, int Nvars, double * concX)

Computes molar concentrations based on temperature and species mass fractions.

$$[X_k] = Y_k \cdot \rho / W_k$$

Parameters

<i>scal</i>	: array of Nspec +1 doubles (T, Y ₁ , Y ₂ , ..., Y _{Nspec}), temperature T [K], mass fractions Y []
<i>Nvars</i>	: no. of variables = Nspec +1

Returns

concX : array of doubles containing species molar concentrations [kmol/m³]

References TC_getRhoMixMs(), TC_Nspec_, TC_Nvars_, and TC_sMass_.

Referenced by TC_getJacRPTYNanl(), TC_getJacRPTYNnum(), TC_getRfrb(), TC_getRops(), TC_getTY2RRml(), TC_getTY2RRms(), and TCDND_getMs2Cc().

7.5.2.10 int TC_getMs2MI (double * Yspec, int Nspec, double * Xspec)

Transforms mass fractions to mole fractions.

$$X_k = Y_k \cdot W_{mix} / W_k$$

Parameters

<i>Yspec</i>	: array of Nspec mass fractions
<i>Nspec</i>	: no. of species

Returns

Xspec : array of Nspec mole fractions

References TC_getMs2Wmix(), TC_Nspec_, and TC_sMass_.

Referenced by TCDND_getMs2MI().

7.5.2.11 int TC_getMs2Wmix (double * Yspec, int Nspec, double * Wmix)

Computes mixture molecular weight based on species mass fractions.

$$W_{mix} = \left(\sum_{k=1}^{N_{spec}} Y_k / W_k \right)^{-1}$$

Parameters

<i>Yspec</i>	: array of Nspec mass fractions
<i>Nspec</i>	: no. of species

Returns

Wmix : pointer to mixture molecular weight [kg/kmol]=[g/mol]

References TC_Nspec_, and TC_sMass_.

Referenced by TC_getJacTYN(), TC_getJacTYNanl(), TC_getJacTYNm1(), TC_getJacTYNm1anl(), TC_getMs2MI(), TC_getSrcCons(), TCDND_getMI2Wmix(), and TCDND_getMs2Wmix().

7.5.2.12 int TC_getRfrb (double * scal, int Nvars, double * datarop)

Returns forward and reverse rate-of-progress variables based on T and Y's.

Parameters

<i>scal</i>	: array of Nspec+1 doubles (T,Y_1,Y_2,...,Y_N) temperature T [K], mass fractions Y []
<i>Nvars</i>	: no. of variables = Nspec + 1

Returns

datarop : array of Nreac forward rate-of-progress variables and Nreac reverse rate-of-progress variables [kmol/(m3.s)]

References TC_errorMSG(), TC_getMs2Cc(), TC_Nreac_, TC_Nspec_, and TC_Nvars_.

7.5.2.13 int TC_getRops (double * scal, int Nvars, double * datarop)

Returns rate-of-progress variables based on temperature T and species mass fractions Y's.

Parameters

<i>scal</i>	: array of N _{reac} +1 doubles (T,Y ₁ ,Y ₂ ,...,Y _N) temperature T [K], mass fractions Y []
<i>Nvars</i>	: no. of variables = N _{spec} +1

Returns

datarop : array of N_{reac} rate-of-progress variables [kmol/(m3.s)]

References TC_errorMSG(), TC_getMs2Cc(), TC_Nreac_, TC_Nspec_, and TC_Nvars_.

7.5.2.14 int TC_getSmass (int Nspec, double * Wi)

Returns species molar weights.

Parameters

<i>Nspec</i>	: no. of species
--------------	------------------

Returns

Wi : array of species molar weights [kg/kmol]=[g/mol]

References TC_Nspec_, and TC_sMass_.

7.5.2.15 int TC_getSnames (int Nspec, char * snames)

Returns species names.

Parameters

<i>Nspec</i>	: no. of species
--------------	------------------

Returns

snames: array of characters containing species names, each name is LENGTHOF-SPECNAME characters

References LENGTHOF-SPECNAME, TC_Nspec_, and TC_sNames_.

7.5.2.16 int TC_getSpos (const char * sname, const int slen)

Returns position a species in the list of species.

Parameters

<i>sname</i>	: string containing the name of the species
<i>slen</i>	: length of species "sname" name

Returns

position of species sname in the list of species, 0...(Nspec -1)

References LENGTHOF-SPECNAME, TC_Nspec_, and TC_sNames_.

7.5.2.17 int TC_getStoiCoef (int Nspec, int Nreac, double * stoicoef)

Returns stoichiometric coefficients' matrix. The stoichiometric coefficient for species "j" in reaction "i" is stored at position $i \cdot N_{spec} + j$. It assumes that stoicoef was dimensioned to at least $N_{reac} \cdot N_{spec}$.

Parameters

<i>Nspec</i>	: no. of species
<i>Nreac</i>	: no. of reactions

Returns

stoicoef : array of stoichiometric coefficients

References TC_errorMSG(), TC_maxSpecInReac_, TC_Nreac_, TC_nRealNuReac_, and TC_Nspec_.

7.5.2.18 int TC_getStoiCoefReac (int Nspec, int Nreac, int ireac, int idx, double * stoicoef)

Returns stoichiometric coefficients' array for reaction 'ireac' for either reactants (idx=0) or products (idx=1) The stoichiometric coefficient for species "j" in reaction "ireac" is

stored at position j . It assumes that stoicoef was dimensioned to at least N_{spec} .

Parameters

<i>Nspec</i>	: no. of species
<i>Nreac</i>	: no. of reactions
<i>ireac</i>	: reaction index
<i>idx</i>	: 0-reactants, 1-products

Returns

stoicoef : array of stoichiometric coefficients

References TC_errorMSG(), TC_maxSpecInReac_, TC_Nreac_, TC_nRealNuReac_, and TC_Nspec_.

7.5.2.19 int TC_getTxC2RRml (double * scal, int Nvars, double * omega)

Returns molar reaction rates based on temperature T and molar concentrations XC.

Parameters

<i>scal</i>	: array of $N_{spec} + 1$ doubles (T, XC_1, XC_2, ..., XC_N) temperature T [K], molar concentrations XC [kmol/m3]
<i>Nvars</i>	: no. of variables = $N_{spec} + 1$

Returns

omega : array of N_{spec} (molar) reaction rates [kmol/(m3.s)]

References TC_errorMSG(), TC_getReacRates(), TC_Nspec_, and TC_Nvars_.

Referenced by TCDND_getTxC2RRml().

7.5.2.20 int TC_getTxC2RRms (double * scal, int Nvars, double * omega)

Returns mass reaction rates based on T and molar concentrations.

Parameters

<i>scal</i>	: array of $N_{spec} + 1$ doubles (T, XC_1, XC_2, ..., XC_N) temperature T [K], molar concentrations XC [kmol/m3]
<i>Nvars</i>	: no. of variables = $N_{spec} + 1$

Returns

omega : array of N_{spec} (mass) reaction rates [kg/(m³.s)]

References TC_errorMSG(), TC_getReacRates(), TC_Nspec_, TC_Nvars_, and TC_s-Mass_.

Referenced by TC_getJacRPTYNnum(), and TCDND_getTxC2RRms().

7.5.2.21 int TC_getTY2RRml (double * scal, int Nvars, double * omega)

Returns molar reaction rates, $\dot{\omega}_i$, based on T and Y's.

Parameters

scal	: array of $N_{spec} + 1$ doubles (T, Y_1, Y_2, \dots, Y_N): temperature T [K], mass fractions Y []
Nvars	: no. of variables $N_{vars} = N_{spec} + 1$

Returns

omega : array of N_{spec} molar reaction rates $\dot{\omega}_i$ [$kmol/(m^3 \cdot s)$]

References TC_errorMSG(), TC_getMs2Cc(), TC_getReacRates(), TC_Nspec_, and TC_Nvars_.

Referenced by TC_getSrcCons(), and TCDND_getTY2RRml().

7.5.2.22 int TC_getTY2RRms (double * scal, int Nvars, double * omega)

Returns mass reaction rates based on T and Y's.

Parameters

scal	: array of $N_{spec} + 1$ doubles (T, Y_1, Y_2, \dots, Y_N) temperature T [K], mass fractions Y []
Nvars	: no. of variables = $N_{spec} + 1$

Returns

omega : array of N_{spec} mass reaction rates [kg/(m³.s)]

References TC_errorMSG(), TC_getMs2Cc(), TC_getReacRates(), TC_Nspec_, TC_Nvars_, and TC_sMass_.

Referenced by TC_getSrc(), and TCDND_getTY2RRms().

7.5.2.23 void TC_removeReaction (int * reacArr, int numRemoveReacs, int revOnly)

Removes a reaction from the mechanism. Not designed for use with tables.

Parameters

<i>reacArr</i>	: 0-based reaction indices in ascending order.
<i>num-Remove-Reacs</i>	: length of rearArr array.
<i>revOnly</i>	: set to 1 to remove reverse only, 0 to remove forward and reverse.

References TC_maxOrdPar_, TC_maxSpecInReac_, TC_maxTbInReac_, TC_nFallPar_, TC_nFallReac_, TC_nOrdReac_, TC_nreac_, TC_nRealNuReac_, TC_nRevReac_, TC_Nspec_, TC_nThbReac_, and TC_removeReaction().

Referenced by TC_removeReaction().

7.5.2.24 int TCDND_getMI2Ms (double * Xspec, int Nspec, double * Yspec)

Transforms mole fractions to mass fractions (same as TC_getMI2Ms()).

$$Y_k = X_k \cdot W_k / W_{mix}$$

Parameters

<i>Xspec</i>	: array of Nspec mole fractions
<i>Nspec</i>	: no. of species

Returns

Yspec : array of Nspec mass fractions

References TC_getMI2Ms(), and TC_Nspec_.

7.5.2.25 int TCDND_getMI2Wmix (double * Xspec, int Nspec, double * Wmix)

Computes mixture molecular weight based on species mole fractions. The molecular weight ([kg/kmol]=[g/mol]) is normalized before output.

$$\bar{W}_{mix} = \frac{W_{mix}}{W_{ref}} = \frac{1}{W_{ref}} \sum_{k=1}^{N_{spec}} X_k W_k$$

Parameters

<i>Xspec</i>	: array of <i>Nspec</i> mole fractions
<i>Nspec</i>	: no. of species

Returns

Wmix : pointer to normalized mixture molecular weight

References TC_getMs2Wmix(), and TC_Nspec_.

7.5.2.26 int TCDND_getMs2Cc (double * scal, int Nvars, double * concX)

Computes molar concentrations based on temperature and species mass fractions. - Input temperature is normalized, output concentrations are also normalized before exit.

$$\overline{[X_k]} = [X_k] \cdot \frac{W_{ref}}{\rho_{ref}} = Y_k \cdot \frac{\rho}{W_k} \cdot \frac{W_{ref}}{\rho_{ref}}$$

Parameters

<i>scal</i>	: array of <i>Nspec</i> +1 doubles (T, Y ₁ , Y ₂ , ..., Y _{Nspec}), temperature T [K], mass fractions Y []
<i>Nvars</i>	: no. of variables = <i>Nspec</i> +1

Returns

concX : array of doubles containing species molar concentrations [kmol/m³]

References TC_getMs2Cc(), TC_Nspec_, and TC_Nvars_.

7.5.2.27 int TCDND_getMs2MI (double * Yspec, int Nspec, double * Xspec)

Transforms mass fractions to mole fractions (same as TC_getMs2MI())

$$X_k = Y_k \cdot W_{mix} / W_k$$

Parameters

<i>Yspec</i>	: array of <i>Nspec</i> mass fractions
<i>Nspec</i>	: no. of species

Returns

Xspec : array of Nspec mole fractions

References TC_getMs2Ml(), and TC_Nspec_.

7.5.2.28 int TCDND_getMs2Wmix (double * Yspec, int Nspec, double * Wmix)

Computes mixture molecular weight based on species mass fractions. The molecular weight ([kg/kmol]=[g/mol]) is normalized before output.

$$\bar{W}_{mix} = \frac{W_{mix}}{W_{ref}} = \frac{1}{W_{ref}} \left(\sum_{k=1}^{N_{spec}} Y_k / W_k \right)^{-1}$$

Parameters

<i>Yspec</i>	: array of Nspec mass fractions
<i>Nspec</i>	: no. of species

Returns

Wmix : pointer to normalized mixture molecular weight

References TC_getMs2Wmix(), and TC_Nspec_.

7.5.2.29 int TCDND_getTxC2RRml (double * scal, int Nvars, double * omega)

Returns non-dimensional molar reaction rates based on temperature T and molar concentrations XC.

Parameters

<i>scal</i>	: array of Nspec + 1 doubles (T, XC_1, XC_2, ..., XC_N) temperature T [K], molar concentrations XC [kmol/m3] (but non-dimensional)
<i>Nvars</i>	: no. of variables = Nspec + 1

Returns

omega : array of Nspec (molar) reaction rates [kmol/(m3.s)] (but non-dimensional)

References TC_errorMSG(), TC_getTxC2RRml(), TC_Nspec_, and TC_Nvars_.

7.5.2.30 int TCDND_getTxC2RRms (double * scal, int Nvars, double * omega)

Returns non-dimensional mass reaction rates based on T and molar concentrations.

Parameters

<i>scal</i>	: array of N _{spec} +1 doubles (T, XC_1, XC_2, ..., XC_N) temperature T [K], molar concentrations XC [kmol/m ³] (but non-dimensional)
<i>Nvars</i>	: no. of variables = N _{spec} +1

Returns

omega : array of N_{spec} (mass) reaction rates [kg/(m³.s)] (but non-dimensional)

References TC_errorMSG(), TC_getTxC2RRms(), TC_Nspec_, and TC_Nvars_.

7.5.2.31 int TCDND_getTY2RRml (double * scal, int Nvars, double * omega)

Returns non-dimensional molar reaction rates, $\dot{\omega}_i \cdot t_{ref} \frac{W_{ref}}{\rho_{ref}}$, based on T and Y's.

Parameters

<i>scal</i>	: array of N _{spec} +1 doubles (T, Y_1, Y_2, ..., Y_N) temperature T [K], mass fractions Y []
<i>Nvars</i>	: no. of variables = N _{spec} +1

Returns

omega : array of N_{spec} molar reaction rates [kmol/(m³.s)] (but non-dimensional)

References TC_errorMSG(), TC_getTY2RRml(), TC_Nspec_, and TC_Nvars_.

7.5.2.32 int TCDND_getTY2RRms (double * scal, int Nvars, double * omega)

Returns non-dimensional mass reaction rates based on T and Y's.

Parameters

<i>scal</i>	: array of N _{spec} +1 doubles (T, Y_1, Y_2, ..., Y_N) temperature T [K], mass fractions Y []
<i>Nvars</i>	: no. of variables = N _{spec} +1

Returns

omega : array of Nspec mass reaction rates [kg/(m3.s)] (but non-dimensional)

References TC_errorMSG(), TC_getTY2RRms(), TC_Nspec_, and TC_Nvars_.

7.6 TC_kmodint.c File Reference

Collection of functions used to parse kinetic models from files.

```
#include <stdio.h> #include <stdlib.h> #include <math.-  
h> #include <string.h> #include <ctype.h> #include <assert.-  
h> #include "TC_params.h" #include "TC_kmodint.h"
```

Functions

- int **TC_kmodint_** (char *mechfile, int *lmech, char *thermofile, int *lthrm)
Kinetic model interpreter.
- void **setperiodictable** (elementable *periodictable, int *Natoms, int iflag)
Read periodic table.
- void **checkelemindex** (char *elemname, element **listelem, int *Nelem, int *ipos)
Returns index of an element in the list of elements:
- int **getelements** (char *linein, char *singleword, element **listelemaddr, int *Nelem, int *Nelemmax, int *iread, int *ierror)
Interprets a character string containing element names and possible their mass.
- void **resetelemdata** (element *currentelem)
Reset data for an element.
- int **setelementmass** (element *listelem, int *Nelem, elementable *periodictable, int *Natoms, int *ierror)
Set the mass for all entries in the list of elements based on the values found in the periodic table.
- int **getspecies** (char *linein, char *singleword, species **listspecaddr, int *Nspec, int *Nspecmax, int *iread, int *ierror)
Interprets a character string containing species names.
- void **resetspecdata** (species *currentspec)
Reset data for a species.
- int **setspecmass** (element *listelem, int *Nelem, species *listspec, int *Nspec, int *ierror)
Set the mass for all entries in the list of elements based on the values found in the periodic table.
- void **checkspecindex** (char *specname, species *listspec, int *Nspec, int *ipos)

Returns position of a species in the list of species The index goes from 0 to (Nspec-1); if the species is not found the value of Nspec is returned.

- int **checkthermo** (**species** *listspec, int *Nspec)

Returns 1 if all species have thermodynamic properties set, 0 otherwise.
- int **getthermo** (char *linein, char *singleword, FILE *mechin, FILE *thermoin, **element** *listelem, int *Nelem, **species** *listspec, int *Nspec, double *Tglobal, int *ithermo, int *iread, int *ierror)

Reads thermodynamic properties (NASA polynomials) from the mechanism input file or from a separate file.
- void **resetreacdata** (**reaction** *currentreac, char *aunits, char *eunits)

Resets the current entry in the list of reactions.
- void **checkunits** (char *linein, char *singleword, char *aunits, char *eunits)

Sets units for the pre-exponential factor and for the activation energy.
- int **getreacline** (char *linein, char *singleword, **species** *listspec, int *Nspec, **reaction** *listreac, int *Nreac, int *ierror)

Interprets a character string containing reaction description (equation + forward - Arrhenius parameters)
- int **getreacauxl** (char *linein, char *singleword, **species** *listspec, int *Nspec, **reaction** *listreac, int *Nreac, int *ierror)

Interprets a character string containing reaction description (auxiliary information)
- int **getreactions** (char *linein, char *singleword, **species** *listspec, int *Nspec, **reaction** *listreac, int *Nreac, char *aunits, char *eunits, int *ierror)

Interprets a character string containing reaction description.
- int **verifyreac** (**element** *listelem, int *Nelem, **species** *listspec, int *Nspec, **reaction** *listreac, int *Nreac, int *ierror)

Verifies corectness and completness for all reactions in the list.
- int **rescalereac** (**reaction** *listreac, int *Nreac)

Verifies corectness and completness for all reactions in the list.
- int **out_formatted** (**element** *listelem, int *Nelem, **species** *listspec, int *Nspec, **reaction** *listreac, int *Nreac, char *aunits, char *eunits, FILE *fileascii)

Outputs reaction data to ascii file.
- int **out_unformatted** (**element** *listelem, int *Nelem, **species** *listspec, int *Nspec, **reaction** *listreac, int *Nreac, char *aunits, char *eunits, FILE *filelist, int *ierror)

Outputs reaction data to an unformatted ascii file.
- void **errormsg** (int ierror)

Outputs error messages.
- int **elimleads** (char *linein)

Checks if a string of characters contains leading spaces Then shifts the string left over the leading spaces, and marks the remaining space at the right with null characters.
- int **elimends** (char *linein)

Checks if a string of characters contains trailing spaces Marks all those positions with null characters.

- int [elimspaces](#) (char *linein)
Eliminates space characters from a string.
- int [elimcomm](#) (char *linein)
Eliminate comments: advances through a line of characters, determines the first occurrence of "!" (if any), then nulls out all the positions downstream (including the "!")
- int [tab2space](#) (char *linein)
Replaces all horizontal tab, vertical tab, line feed, and carriage return characters in a line with spaces.
- int [extractWordLeft](#) (char *linein, char *oneword)
Extracts the left most word from a character string.
- int [extractWordLeftauxline](#) (char *linein, char *oneword, char *twoword, int *inum, int *ierror)
Extracts two strings from a character strings.
- int [extractWordLeftNoslash](#) (char *linein, char *oneword)
The only difference between this method and "extractWordLeft" is the absence of "/" as delimiter.
- int [extractWordRight](#) (char *linein, char *oneword)
Extracts last word from a character strings: (1) assumes the word is separated from the rest of the string by at least a space; (2) the corresponding positions in the initial string are filled with null characters.
- int [extractdouble](#) (char *wordval, double *dvalues, int *inum, int *ierror)
Extracts a double number from a character string: (1) the number is assumed to be the left most word in the string; (2) words are separated by spaces.
- void [wordtupper](#) (char *linein, char *oneword, int Npos)
Converts all letters in a character string to uppercase.
- void [cleancharstring](#) (char *linein, int *len1)
Performs various operations on a character strings (see explanations for individual methods)
- int [charfixespc](#) (char *singleword, int *len1)
- int [checkstrnum](#) (char *singleword, int *len1, int *ierror)
Checks if all components of a character string are valid number characters (as described by the f or e formats)
- int [findnonnum](#) (char *specname, int *ipos)
Identifies the first position in the character string that does not correspond to a positive f format. (in other words, finds the first position that is not a digit or a decimal point)
- int [kmodsum](#) ([element](#) *listelem, int *Nelem, [species](#) *listspec, int *Nspec, [reaction](#) *listreac, int *Nreac, int *nlonEspec, int *electrIdx, int *nlonSpec, int *maxSpecInReac, int *maxTbInReac, int *maxOrdPar, int *nFallPar, int *maxTpRange, int *nLtReac, int *nRltReac, int *nFallReac, int *nThbReac, int *nRevReac, int *nHvReac, int *nTdepReac, int *nJanReac, int *nFit1Reac, int *nExciReac, int *nMomeReac, int *nXsmiReac, int *nRealNuReac, int *nOrdReac, int *nNASAinter, int *nCpCoef, int *nNASAfit, int *nArhPar, int *nLtPar, int *nJanPar, int *nFit1Par)

kinetic model summary

- int `out_mathem` (`element` *listelem, int *Nelem, `species` *listspec, int *Nspec, `reaction` *listreac, int *Nreac, char *aunits, char *eunits)

Outputs reaction data to ascii file.

7.6.1 Detailed Description

Collection of functions used to parse kinetic models from files. TC_kmodint - utility used to parse kinetic models -----

Usage: `TC_kmodint_(char *mechfile,int *lmech,char *thermofile,int *lthrm)`

- `mechfile`: file containing the kinetic model
- `lmech`: length of the character string above (introduced to enable passing of character strings from Fortran to C)
- `thermofile`: file containing thermodynamic properties (NASA polynomials)
- `lmech`: length of the character string thermofile (introduced to enable passing of character strings from Fortran to C)

Output:

- `kmod.out` - ascii file containing kinetic model info formatted for visual inspection
- `kmod.list` - ascii file containing unformatted data for `tchem`

Brief description of kinetic model input format (For a detailed description of the kinetic model format and keywords see: Robert J. Kee, Fran M. Rupley, Ellen Meeks, and James A. Miller "CHEMKIN-III:A Fortran Chemical Kinetics Package for the Analysis of Gas-phase Chemical and Plasma Kinetics", Sandia Report, SAND96-8216, (1996))

Elements :

- Number of elements is "unlimited"
- Elements not present in file `periodictable.dat` must be followed by their atomic weight, e.g. `N+ /14.0010/`
- Element names are one or two characters long
- Any duplicate listing of an element is ignored

Species :

- Number of species is "unlimited"

- Species need to be formed only of elements declared in the list of elements
- Species names are "LENGTHOFSPECNAME" characters long
- Any duplicate listing of an species is ignored
- A species can contain at most "NUMBEROFELEMINSPEC" distinct elements

Thermodynamic data :

- Data can be provided in the kinetic model file and/or the thermodynamic file;
- Currently, only NASA polynomials are accepted; two (2) temperature intervals
- Data needs to be provided for all species

Reactions :

- Number of reactions is "unlimited"
- Pre-exponential factor units MOLES or MOLECULES; default is MOLES
- Activation energy units: CAL/MOLE, KCAL/MOLE, JOULES/MOLE, KJOULES/MOLE, KELVINS, eVOLTS ; default is CAL/MOLE. Units are converted to KELVINS if necessary; conversion factors are based on NIST data as of July 2007
- maximum number of reactants or products is "NSPECREACMAX"
- reactants and products are separated by "<=>" or "=" (reversible reactions) or "=>" (irreversible reactions)
- species are separated by "+"
- three Arrhenius parameters should be given for each reaction in the order : pre-exponential factor, temperature exponent, activation energy
- reaction lines that are too long can be split on several lines using the character "&" at the end of each line

Auxiliary reaction info :

- Auxiliary data needs to be provided immediately following the reaction to which it corresponds to
- Any keywords except DUPLICATE, MOME, and XSMI need to be followed by numerical values enclosed between "/"
- Duplicate reactions

DUPLICATE

- third-body efficiencies for reactions containing "+M" (not "(+M)") as a reactant and/or product:
speciesname /value/ the maximum number of third-body efficiencies is given by "NTHRDBMAX"
- pressure-dependent reaction are signaled by the inclusion of "(+M)" as a reactant and/or product or by the inclusion of a particular species, e.g. "(+H2)" as a reactant and/or product. Some of the following parameters are required to describe the pressure dependency:
LOW /value1 value2 value3/
HIGH /value1 value2 value3/
TROE /value1 value2 value3 value4/ (if value4 is omitted then the corresponding term is omitted in the corresponding Troe formulation)
SRI /value1 value2 value3 value4 value5/ (if value4 and value5 are omitted then value4=1.0, value5=0.0)
- Landau-Teller reactions
LT /value1 value2/ for the forward rate
RLT /value1 value2/ for the reverse rate. If REV is given then RLT is mandatory; if not then RLT is optional
- Additional rate fit expressions:
JAN /value1 value2 ... value9/
FIT1 /value1 value2 value3 value4/
- Radiation wavelength for reactions containing HV as a reactant and/or product
HV /value1/
- Reaction rate dependence on a particular species temperature
TDEP /specname/
– Energy loss parameter
EXCI /value1/
- Plasma (Ion) momentum-transfer collision frequency
MOME (XSMI)
- Reverse reaction Arrhenius parameters
REV /value1 value2 value3/
- Change reaction order parameters
FORD /specname value1/ (for forward rate)
RORD /specname value1/ (for reverse rate)

- Reaction units for reactions with units different than most of the other reactions
UNITS /unit1 unit2/
(the number of keywords between "/" can be one if only one set of units is changed or two if both pre-exponential factor and activation energy are to be modified)

7.6.2 Function Documentation

7.6.2.1 void checkelemindex (char * *elemname*, element * *listelem*, int * *Nelem*, int * *ipos*)

Returns index of an element in the list of elements:

- the index goes from 0 to (Nelem-1);
- the value of Nelem is returned if the element is not found

Referenced by getthermo().

7.6.2.2 int extractWordLeft (char * *linein*, char * *oneword*)

Extracts the left most word from a character string.

- this function assumes words are separated by spaces or "/"
- the initial character string is shifted to the left starting with the separation character
- the corresponding positions left at the right are filled with null characters

References elimleads().

Referenced by getelements(), getspecies(), getthermo(), and TC_kmodint_().

7.6.2.3 int extractWordLeftauxline (char * *linein*, char * *oneword*, char * *twoword*, int * *inum*, int * *ierror*)

Extracts two strings from a character strings.

- the first string starts from the first position until a space or a "/"
- the second string is starts after the first slash and ends at the second "/"
 - the initial character string is shifted to the left starting with the first character after the second "/"

- the corresponding positions left at the right are filled with null characters

References `elimleads()`.

Referenced by `getreacauxl()`.

7.6.2.4 `int getreactions (char * linein, char * singleword, species * listspec, int * Nspec, reaction * listreac, int * Nreac, char * aunits, char * eunits, int * ierror)`

Interprets a character string containing reaction description.

- decides if the character string describes a reaction or the auxiliary information associated with one

References `getreacauxl()`, `getreacline()`, and `resetreacdata()`.

Referenced by `TC_kmodint_()`.

7.6.2.5 `void setperiodictable (elemtable * periodictable, int * Natoms, int iflag)`

Read periodic table.

- First line contains two integer values: the total number of elements and the number of elements listed on each of the following lines
- The following lines (an even number) contain lists of elements names (two characters separated by one or more spaces, and elemental masses (separated by spaces):
 1. S1 S2 ... S(Nline)
 2. M1 M2 ... M(Nline)
 3. S(Nline+1) S(Nline+2)
 4. M(Nline+1) M(Nline+2)

Referenced by `TC_kmodint_()`.

7.6.2.6 `int tab2space (char * linein)`

Replaces all horizontal tab, vertical tab, line feed, and carriage return characters in a line with spaces.

- ASCII code for a horizontal TAB is 9

- ASCII code for a vertical TAB is 11
- ASCII code for a SPACE is 32
- ASCII code for line feed (new line) is 10
- ASCII code for carriage return is 15

Referenced by cleancharstring().

7.7 TC_mlms.c File Reference

Mass fractions - Mole fractions - Molar concentrations - Molecular weight.

Functions

- int [TCNDND_getMs2Cc](#) (double *scal, int Nvars, double *concX)
Computes molar concentrations based on temperature and species mass fractions. Input temperature is normalized, output concentrations are also normalized before exit.

$$\overline{[X_k]} = [X_k] \cdot \frac{W_{ref}}{\rho_{ref}} = Y_k \cdot \frac{\rho}{W_k} \cdot \frac{W_{ref}}{\rho_{ref}}$$

- int [TC_getMs2Cc](#) (double *scal, int Nvars, double *concX)
Computes molar concentrations based on temperature and species mass fractions.

$$[X_k] = Y_k \cdot \rho / W_k$$

- int [TCNDND_getMI2Ms](#) (double *Xspec, int Nspec, double *Yspec)
Transforms mole fractions to mass fractions (same as [TC_getMI2Ms\(\)](#)).

$$Y_k = X_k \cdot W_k / W_{mix}$$

- int [TC_getMI2Ms](#) (double *Xspec, int Nspec, double *Yspec)
Transforms mole fractions to mass fractions.

$$Y_k = X_k \cdot W_k / W_{mix}$$

- int [TCNDND_getMs2MI](#) (double *Yspec, int Nspec, double *Xspec)
Transforms mass fractions to mole fractions (same as [TC_getMs2MI\(\)](#)).

$$X_k = Y_k \cdot W_{mix} / W_k$$

- int [TC_getMs2MI](#) (double *Yspec, int Nspec, double *Xspec)

Transforms mass fractions to mole fractions.

$$X_k = Y_k \cdot W_{mix} / W_k$$

- int [TCDND_getMs2Wmix](#) (double *Yspec, int Nspec, double *Wmix)

Computes mixture molecular weight based on species mass fractions. The molecular weight ([kg/kmol]=[g/mol]) is normalized before output.

$$\bar{W}_{mix} = \frac{W_{mix}}{W_{ref}} = \frac{1}{W_{ref}} \left(\sum_{k=1}^{N_{spec}} Y_k / W_k \right)^{-1}$$

- int [TC_getMs2Wmix](#) (double *Yspec, int Nspec, double *Wmix)

Computes mixture molecular weight based on species mass fractions.

$$W_{mix} = \left(\sum_{k=1}^{N_{spec}} Y_k / W_k \right)^{-1}$$

- int [TCDND_getMI2Wmix](#) (double *Xspec, int Nspec, double *Wmix)

Computes mixture molecular weight based on species mole fractions. The molecular weight ([kg/kmol]=[g/mol]) is normalized before output.

$$\bar{W}_{mix} = \frac{W_{mix}}{W_{ref}} = \frac{1}{W_{ref}} \sum_{k=1}^{N_{spec}} X_k W_k$$

- int [TC_getMI2Wmix](#) (double *Xspec, int Nspec, double *Wmix)

Computes mixture molecular weight based on species mole fractions.

$$W_{mix} = \sum_{k=1}^{N_{spec}} X_k W_k$$

7.7.1 Detailed Description

Mass fractions - Mole fractions - Molar concentrations - Molecular weight.

7.7.2 Function Documentation

7.7.2.1 int TC_getMI2Ms (double * Xspec, int Nspec, double * Yspec)

Transforms mole fractions to mass fractions.

$$Y_k = X_k \cdot W_k / W_{mix}$$

Parameters

<i>Xspec</i>	: array of N_{spec} mole fractions
<i>Nspec</i>	: no. of species

Returns

Yspec : array of N_{spec} mass fractions

References TC_getMI2Wmix(), TC_Nspec_, and TC_sMass_.

Referenced by TCDND_getMI2Ms().

7.7.2.2 int TC_getMI2Wmix (double * *Xspec*, int *Nspec*, double * *Wmix*)

Computes mixture molecular weight based on species mole fractions.

$$W_{\text{mix}} = \sum_{k=1}^{N_{\text{spec}}} X_k W_k$$

Parameters

<i>Xspec</i>	: array of N_{spec} mole fractions
<i>Nspec</i>	: no. of species

Returns

Wmix : pointer to mixture molecular weight [kg/kmol]=[g/mol]

References TC_Nspec_, and TC_sMass_.

Referenced by TC_getMI2Ms().

7.7.2.3 int TC_getMs2Cc (double * *scal*, int *Nvars*, double * *concX*)

Computes molar concentrations based on temperature and species mass fractions.

$$[X_k] = Y_k \cdot \rho / W_k$$

Parameters

<i>scal</i>	: array of $N_{\text{spec}} + 1$ doubles ($T, Y_1, Y_2, \dots, Y_{N_{\text{spec}}}$), temperature T [K], mass fractions Y []
<i>Nvars</i>	: no. of variables = $N_{\text{spec}} + 1$

Returns

concX : array of doubles containing species molar concentrations [kmol/m³]

References TC_getRhoMixMs(), TC_Nspec_, TC_Nvars_, and TC_sMass_.

Referenced by TC_getJacRPTYNanl(), TC_getJacRPTYNnum(), TC_getRfrb(), TC_getRops(), TC_getTY2RRml(), TC_getTY2RRms(), and TCDND_getMs2Cc().

7.7.2.4 int TC_getMs2MI (double * Yspec, int Nspec, double * Xspec)

Transforms mass fractions to mole fractions.

$$X_k = Y_k \cdot W_{mix} / W_k$$

Parameters

<i>Yspec</i>	: array of N _{spec} mass fractions
<i>Nspec</i>	: no. of species

Returns

Xspec : array of N_{spec} mole fractions

References TC_getMs2Wmix(), TC_Nspec_, and TC_sMass_.

Referenced by TCDND_getMs2MI().

7.7.2.5 int TC_getMs2Wmix (double * Yspec, int Nspec, double * Wmix)

Computes mixture molecular weight based on species mass fractions.

$$W_{mix} = \left(\sum_{k=1}^{N_{spec}} Y_k / W_k \right)^{-1}$$

Parameters

<i>Yspec</i>	: array of N _{spec} mass fractions
<i>Nspec</i>	: no. of species

Returns

Wmix : pointer to mixture molecular weight [kg/kmol]=[g/mol]

References TC_Nspec_, and TC_sMass_.

Referenced by TC_getJacTYN(), TC_getJacTYNanl(), TC_getJacTYNm1(), TC_getJacTYNm1anl(), TC_getMs2MI(), TC_getSrcCons(), TCDND_getMI2Wmix(), and TC-DND_getMs2Wmix().

7.7.2.6 int TCDND_getMI2Ms (double * Xspec, int Nspec, double * Yspec)

Transforms mole fractions to mass fractions (same as TC_getMI2Ms()).

$$Y_k = X_k \cdot W_k / W_{mix}$$

Parameters

Xspec	: array of Nspec mole fractions
Nspec	: no. of species

Returns

Yspec : array of Nspec mass fractions

References TC_getMI2Ms(), and TC_Nspec_.

7.7.2.7 int TCDND_getMI2Wmix (double * Xspec, int Nspec, double * Wmix)

Computes mixture molecular weight based on species mole fractions. The molecular weight ([kg/kmol]=[g/mol]) is normalized before output.

$$\bar{W}_{mix} = \frac{W_{mix}}{W_{ref}} = \frac{1}{W_{ref}} \sum_{k=1}^{N_{spec}} X_k W_k$$

Parameters

Xspec	: array of Nspec mole fractions
Nspec	: no. of species

Returns

Wmix : pointer to normalized mixture molecular weight

References TC_getMs2Wmix(), and TC_Nspec_.

7.7.2.8 int TCDND_getMs2Cc (double * scal, int Nvars, double * concX)

Computes molar concentrations based on temperature and species mass fractions. - Input temperature is normalized, output concentrations are also normalized before exit.

$$\overline{[X_k]} = [X_k] \cdot \frac{W_{ref}}{\rho_{ref}} = Y_k \cdot \frac{\rho}{W_k} \cdot \frac{W_{ref}}{\rho_{ref}}$$

Parameters

<i>scal</i>	: array of Nspec +1 doubles (T, Y ₁ , Y ₂ , ..., Y _{Nspec}), temperature T [K], mass fractions Y []
<i>Nvars</i>	: no. of variables = Nspec +1

Returns

concX : array of doubles containing species molar concentrations [kmol/m³]

References TC_getMs2Cc(), TC_Nspec_, and TC_Nvars_.

7.7.2.9 int TCDND_getMs2MI (double * Yspec, int Nspec, double * Xspec)

Transforms mass fractions to mole fractions (same as [TC_getMs2MI\(\)](#))

$$X_k = Y_k \cdot W_{mix} / W_k$$

Parameters

<i>Yspec</i>	: array of Nspec mass fractions
<i>Nspec</i>	: no. of species

Returns

Xspec : array of Nspec mole fractions

References TC_getMs2MI(), and TC_Nspec_.

7.7.2.10 int TCDND_getMs2Wmix (double * Yspec, int Nspec, double * Wmix)

Computes mixture molecular weight based on species mass fractions. The molecular weight ([kg/kmol]=[g/mol]) is normalized before output.

$$\bar{W}_{mix} = \frac{W_{mix}}{W_{ref}} = \frac{1}{W_{ref}} \left(\sum_{k=1}^{N_{spec}} Y_k / W_k \right)^{-1}$$

Parameters

<i>Yspec</i>	: array of Nspec mass fractions
<i>Nspec</i>	: no. of species

Returns

Wmix : pointer to normalized mixture molecular weight

References TC_getMs2Wmix(), and TC_Nspec_.

7.8 TC_params.h File Reference

Definitions of parameters and constants.

```
#include "copyright.h"
```

Defines

- #define MAX(A, B) (((A) > (B)) ? (A) : (B))
- #define MIN(A, B) (((A) < (B)) ? (A) : (B))
- #define LENGTHOFELEMENTNAME 3
- #define LENGTHOFSPECNAME 18
- #define NUMBEROFELEMENTINSPEC 5
- #define NTHRDBMAX 10
- #define NSPECREACMAX 6
- #define REACBALANCE 1.e-4
- #define RUNIV 8.314472
- #define NAVOG 6.02214179E23
- #define ATMPA 101325.0
- #define CALJO 4.184
- #define KBOLT 1.3806504E-23
- #define EVOLT 1.60217653E-19

7.8.1 Detailed Description

Definitions of parameters and constants.

7.8.2 Define Documentation

7.8.2.1 #define ATMPA 101325.0

Standard atmospheric pressure [*Pa*]

Referenced by TC_initChem().

7.8.2.2 #define CALJO 4.184

Conversion from calories to Joule

Referenced by rescalereac(), and TC_initChem().

7.8.2.3 #define EVOLT 1.60217653E-19

electron volt (eV) unit [*J*]

Referenced by rescalereac().

7.8.2.4 #define KBOLT 1.3806504E-23

Boltzmann's constant (k_B) [JK^{-1}]

Referenced by rescalereac().

7.8.2.5 #define LENGTHOFELEMENTNAME 3

Maximum number of characters for element names

Referenced by TC_initChem().

7.8.2.6 #define LENGTHOFSPECNAME 18

Maximum number of characters for species names

Referenced by getreacLine(), getspecies(), TC_getSnameLen(), TC_getSnames(), TC_getSpos(), and TC_initChem().

7.8.2.7 `#define MAX(A, B) (((A) > (B)) ? (A) : (B))`

Maximum of two expressions

Referenced by `kmodsum()`.

7.8.2.8 `#define MIN(A, B) (((A) < (B)) ? (A) : (B))`

Minimum of two expressions

Referenced by `TC_kmodint_()`.

7.8.2.9 `#define NAVOG 6.02214179E23`

Avogadro's number

Referenced by `rescalereac()`.

7.8.2.10 `#define NSPECREACMAX 6`

Maximum number of reactant or product species in a reaction

Referenced by `getreacauxl()`, `getreacline()`, `out_formatted()`, `out_mathem()`, `out_unformatted()`, `rescalereac()`, `resetreacdata()`, and `verifyreac()`.

7.8.2.11 `#define NTHRDBMAX 10`

Maximum number of third body efficiencies

Referenced by `getreacauxl()`, and `resetreacdata()`.

7.8.2.12 `#define NUMBEROFELEMINSPEC 5`

Maximum number of (different) elements that compose a species

Referenced by `resetspecdata()`.

7.8.2.13 `#define REACBALANCE 1.e-4`

Threshold for checking reaction balance with real stoichiometric coefficients

Referenced by `out_formatted()`, `out_unformatted()`, and `verifyreac()`.

7.8.2.14 #define RUNIV 8.314472

Universal gas constant $J/(mol \cdot K)$

Referenced by rescalereac(), and TC_initChem().

7.9 TC_rr.c File Reference

Reaction rate functions.

Functions

- int [TC_getNreac](#) ()
Returns number of reactions N_{reac} .
- int [TC_getStoiCoef](#) (int Nspec, int Nreac, double *stoicoef)
Returns stoichiometric coefficients' matrix. The stoichiometric coefficient for species "j" in reaction "i" is stored at position $i \cdot N_{spec} + j$. It assumes that stoicoef was dimensioned to at least $N_{reac} \cdot N_{spec}$.
- int [TC_getStoiCoefReac](#) (int Nspec, int Nreac, int ireac, int idx, double *stoicoef)
Returns stoichiometric coefficients' array for reaction "ireac" for either reactants ($idx=0$) or products ($idx=1$) The stoichiometric coefficient for species "j" in reaction "ireac" is stored at position j . It assumes that stoicoef was dimensioned to at least N_{spec} .
- int [TC_getArhenFor](#) (int ireac, int ipos, double *val)
Return current value of the Arrhenius parameters for forward rate constants. Return -1 if no data available, otherwise return 0 and store value in val.
- int [TC_getArhenRev](#) (int ireac, int ipos, double *val)
Return current value of the Arrhenius parameters for reverse rate constants. Return -1 if no data available, otherwise return 0 and store value in val.
- int [TCDND_getTY2RRml](#) (double *scal, int Nvars, double *omega)
Returns non-dimensional molar reaction rates, $\dot{\omega}_i \cdot t_{ref} \frac{W_{ref}}{\rho_{ref}}$, based on T and Y's.
- int [TC_getTY2RRml](#) (double *scal, int Nvars, double *omega)
Returns molar reaction rates, $\dot{\omega}_i$, based on T and Y's.
- int [TCDND_getTY2RRms](#) (double *scal, int Nvars, double *omega)
Returns non-dimensional mass reaction rates based on T and Y's.
- int [TC_getTY2RRms](#) (double *scal, int Nvars, double *omega)
Returns mass reaction rates based on T and Y's.
- int [TCDND_getTXC2RRml](#) (double *scal, int Nvars, double *omega)
Returns non-dimensional molar reaction rates based on temperature T and molar concentrations XC.
- int [TC_getTXC2RRml](#) (double *scal, int Nvars, double *omega)

Returns molar reaction rates based on temperature T and molar concentrations XC .

- int [TCDND_getTxC2RRms](#) (double *scal, int Nvars, double *omega)

Returns non-dimensional mass reaction rates based on T and molar concentrations.

- int [TC_getTxC2RRms](#) (double *scal, int Nvars, double *omega)

Returns mass reaction rates based on T and molar concentrations.

- int [TC_getRops](#) (double *scal, int Nvars, double *datarop)

Returns rate-of-progress variables based on temperature T and species mass fractions Y 's.

- int [TC_getRfrb](#) (double *scal, int Nvars, double *dataRfrb)

Returns forward and reverse rate-of-progress variables based on T and Y 's.

- int [TC_getRopsLocal](#) (double *scal)
- int [TC_getReacRates](#) (double *scal, int Nvars, double *omega)

Returns molar reaction rates, $\dot{\omega}_i$, based on T and molar concentrations XC 's (semi-private function)

- int [TC_getgk](#) (double t1, double t_1, double tln)
- int [TC_getgkFcn](#) (double t1, double t_1, double tln)
- int [TC_getgkTab](#) (double t1)
- int [TC_getgkp](#) (double t1, double t_1, double tln)
- int [TC_getgkpFcn](#) (double t1, double t_1, double tln)
- int [TC_getgkpTab](#) (double t1)
- double [TC_getSumNuGk](#) (int i, double *gkLoc)
- double [TC_getSumRealNuGk](#) (int i, int ir, double *gkLoc)
- int [TC_get3rdBdyConc](#) (double *concX, double *concM)
- int [TC_getkForRev](#) (double t1, double t_1, double tln)
- int [TC_getkForRevFcn](#) (double t_1, double tln)
- int [TC_getkForRevTab](#) (double t1)
- int [TC_getkForRevP](#) (double t1, double t_1)
- int [TC_getkForRevPFcn](#) (double t_1)
- int [TC_getkForRevPTab](#) (double t1)
- int [TC_getRateofProg](#) (double *concX)
- int [TC_getRateofProgDer](#) (double *concX, int ireac, int ispec, double *qfr)
- int [TC_getCrnd](#) (double t1, double t_1, double tln, double *concX, double *concM)
- int [TC_getCrndDer](#) (int ireac, int *itbdy, int *ipfal, double t1, double t_1, double tln, double *concX, double *concM)

7.9.1 Detailed Description

Reaction rate functions.

7.9.2 Function Documentation

7.9.2.1 int TC_getArhenFor (int *ireac*, int *ipos*, double * *val*)

Return current value of the Arrhenius parameters for forward rate constants. Return -1 if no data available, otherwise return 0 and store value in *val*.

Parameters

<i>ireac</i>	: reaction index
<i>ipos</i>	: index of Arrhenius parameter (0) pre-exponential factor (1) temperature exponent, (2) activation energy
* <i>val</i>	: value of Arrhenius parameter

References TC_Nreac_.

7.9.2.2 int TC_getArhenRev (int *ireac*, int *ipos*, double * *val*)

Return current value of the Arrhenius parameters for reverse rate constants. Return -1 if no data available, otherwise return 0 and store value in *val*.

Parameters

<i>ireac</i>	: reaction index
<i>ipos</i>	: index of of Arrhenius parameter (0) pre-exponential factor (1) temperature exponent, (2) activation energy
* <i>val</i>	: value of Arrhenius parameter

References TC_nRevReac_.

7.9.2.3 int TC_getReacRates (double * *scal*, int *Nvars*, double * *omega*)

Returns molar reaction rates, $\dot{\omega}_i$, based on T and molar concentrations XC's (semi-private function)

Parameters

<i>scal</i>	: array of $N_{spec} + 1$ doubles ((<i>T</i> , <i>XC</i> ₁ , <i>XC</i> ₂ , ..., <i>XC</i> _{<i>N</i>}): temperature T [K], molar concentrations XC [<i>kmol/m</i> ³])
<i>Nvars</i>	: no. of variables $N_{vars} = N_{spec} + 1$

Returns

omega : array of N_{spec} molar reaction rates $\dot{\omega}_i$ [$kmol/(m^3 \cdot s)$]

References TC_maxSpecInReac_, TC_Nreac_, TC_nRealNuReac_, and TC_Nspec_.

Referenced by TC_getTXC2RRml(), TC_getTXC2RRms(), TC_getTY2RRml(), and TC_getTY2RRms().

7.9.2.4 int TC_getRfrb (double * scal, int Nvars, double * dataRfrb)

Returns forward and reverse rate-of-progress variables based on T and Y's.

Parameters

scal	: array of Nspec+1 doubles (T,Y_1,Y_2,...,Y_N) temperature T [K], mass fractions Y []
Nvars	: no. of variables = Nspec +1

Returns

datarop : array of Nreac forward rate-of-progress variables and Nreac reverse rate-of-progress variables [kmol/(m3.s)]

References TC_errorMSG(), TC_getMs2Cc(), TC_Nreac_, TC_Nspec_, and TC_Nvars_.

7.9.2.5 int TC_getRops (double * scal, int Nvars, double * datarop)

Returns rate-of-progress variables based on temperature T and species mass fractions Y's.

Parameters

scal	: array of Nreac +1 doubles (T,Y_1,Y_2,...,Y_N) temperature T [K], mass fractions Y []
Nvars	: no. of variables = Nspec +1

Returns

datarop : array of Nreac rate-of-progress variables [kmol/(m3.s)]

References TC_errorMSG(), TC_getMs2Cc(), TC_Nreac_, TC_Nspec_, and TC_Nvars_.

7.9.2.6 int TC_getStoiCoef (int *Nspec*, int *Nreac*, double * *stoicoef*)

Returns stoichiometric coefficients' matrix. The stoichiometric coefficient for species "j" in reaction "i" is stored at position $i \cdot N_{spec} + j$. It assumes that stoicoef was dimensioned to at least $N_{reac} \cdot N_{spec}$.

Parameters

<i>Nspec</i>	: no. of species
<i>Nreac</i>	: no. of reactions

Returns

stoicoef : array of stoichiometric coefficients

References TC_errorMSG(), TC_maxSpecInReac_, TC_Nreac_, TC_nRealNuReac_, and TC_Nspec_.

7.9.2.7 int TC_getStoiCoefReac (int *Nspec*, int *Nreac*, int *ireac*, int *idx*, double * *stoicoef*)

Returns stoichiometric coefficients' array for reaction 'ireac' for either reactants (idx=0) or products (idx=1) The stoichiometric coefficient for species "j" in reaction "ireac" is stored at position j . It assumes that stoicoef was dimensioned to at least N_{spec} .

Parameters

<i>Nspec</i>	: no. of species
<i>Nreac</i>	: no. of reactions
<i>ireac</i>	: reaction index
<i>idx</i>	: 0-reactants, 1-products

Returns

stoicoef : array of stoichiometric coefficients

References TC_errorMSG(), TC_maxSpecInReac_, TC_Nreac_, TC_nRealNuReac_, and TC_Nspec_.

7.9.2.8 int TC_getTXC2RRml (double * *scal*, int *Nvars*, double * *omega*)

Returns molar reaction rates based on temperature T and molar concentrations XC.

Parameters

<i>scal</i>	: array of $N_{spec} + 1$ doubles (T,XC_1,XC_2,...,XC_N) temperature T [K], molar concentrations XC [kmol/m3]
<i>Nvars</i>	: no. of variables = $N_{spec} + 1$

Returns

ω : array of N_{spec} (molar) reaction rates [kmol/(m3.s)]

References TC_errorMSG(), TC_getReacRates(), TC_Nspec_, and TC_Nvars_.

Referenced by TCDND_getTXC2RRml().

7.9.2.9 int TC_getTXC2RRms (double * *scal*, int *Nvars*, double * *omega*)

Returns mass reaction rates based on T and molar concentrations.

Parameters

<i>scal</i>	: array of $N_{spec} + 1$ doubles (T,XC_1,XC_2,...,XC_N) temperature T [K], molar concentrations XC [kmol/m3]
<i>Nvars</i>	: no. of variables = $N_{spec} + 1$

Returns

ω : array of N_{spec} (mass) reaction rates [kg/(m3.s)]

References TC_errorMSG(), TC_getReacRates(), TC_Nspec_, TC_Nvars_, and TC_s-Mass_.

Referenced by TC_getJacRPTYNnum(), and TCDND_getTXC2RRms().

7.9.2.10 int TC_getTY2RRml (double * *scal*, int *Nvars*, double * *omega*)

Returns molar reaction rates, $\dot{\omega}_i$, based on T and Y's.

Parameters

<i>scal</i>	: array of $N_{spec} + 1$ doubles (T, Y_1, Y_2, \dots, Y_N): temperature T [K], mass fractions Y []
<i>Nvars</i>	: no. of variables $N_{vars} = N_{spec} + 1$

Returns

omega : array of N_{spec} molar reaction rates $\dot{\omega}_i$ [$kmol/(m^3 \cdot s)$]

References TC_errorMSG(), TC_getMs2Cc(), TC_getReacRates(), TC_Nspec_, and TC_Nvars_.

Referenced by TC_getSrcCons(), and TCDND_getTY2RRml().

7.9.2.11 int TC_getTY2RRms (double * scal, int Nvars, double * omega)

Returns mass reaction rates based on T and Y's.

Parameters

<i>scal</i>	: array of $N_{spec} + 1$ doubles (T,Y_1,Y_2,...,Y_N) temperature T [K], mass fractions Y []
<i>Nvars</i>	: no. of variables = $N_{spec} + 1$

Returns

omega : array of N_{spec} mass reaction rates [kg/(m3.s)]

References TC_errorMSG(), TC_getMs2Cc(), TC_getReacRates(), TC_Nspec_, TC_Nvars_, and TC_sMass_.

Referenced by TC_getSrc(), and TCDND_getTY2RRms().

7.9.2.12 int TCDND_getTXC2RRml (double * scal, int Nvars, double * omega)

Returns non-dimensional molar reaction rates based on temperature T and molar concentrations XC.

Parameters

<i>scal</i>	: array of $N_{spec} + 1$ doubles (T,XC_1,XC_2,...,XC_N) temperature T [K], molar concentrations XC [kmol/m3] (but non-dimensional)
<i>Nvars</i>	: no. of variables = $N_{spec} + 1$

Returns

omega : array of N_{spec} (molar) reaction rates [kmol/(m3.s)] (but non-dimensional)

References TC_errorMSG(), TC_getTXC2RRml(), TC_Nspec_, and TC_Nvars_.

7.9.2.13 int TCDND_getTXC2RRms (double * scal, int Nvars, double * omega)

Returns non-dimensional mass reaction rates based on T and molar concentrations.

Parameters

scal	: array of N _{spec} +1 doubles (T, XC_1, XC_2, ..., XC_N) temperature T [K], molar concentrations XC [kmol/m ³] (but non-dimensional)
Nvars	: no. of variables = N _{spec} +1

Returns

omega : array of N_{spec} (mass) reaction rates [kg/(m³.s)] (but non-dimensional)

References TC_errorMSG(), TC_getTXC2RRms(), TC_Nspec_, and TC_Nvars_.

7.9.2.14 int TCDND_getTY2RRml (double * scal, int Nvars, double * omega)

Returns non-dimensional molar reaction rates, $\dot{\omega}_i \cdot t_{ref} \frac{W_{ref}}{\rho_{ref}}$, based on T and Y's.

Parameters

scal	: array of N _{spec} +1 doubles (T, Y_1, Y_2, ..., Y_N) temperature T [K], mass fractions Y []
Nvars	: no. of variables = N _{spec} +1

Returns

omega : array of N_{spec} molar reaction rates [kmol/(m³.s)] (but non-dimensional)

References TC_errorMSG(), TC_getTY2RRml(), TC_Nspec_, and TC_Nvars_.

7.9.2.15 int TCDND_getTY2RRms (double * scal, int Nvars, double * omega)

Returns non-dimensional mass reaction rates based on T and Y's.

Parameters

scal	: array of N _{spec} +1 doubles (T, Y_1, Y_2, ..., Y_N) temperature T [K], mass fractions Y []
Nvars	: no. of variables = N _{spec} +1

Returns

omega : array of Nspec mass reaction rates [kg/(m³.s)] (but non-dimensional)

References TC_errorMSG(), TC_getTY2RRms(), TC_Nspec_, and TC_Nvars_.

7.10 TC_spec.c File Reference

Species info.

Functions

- int [TC_getNspec](#) ()
Returns no. of species Nspec
- int [TC_getNelem](#) ()
Returns no. of elements Nelem
- int [TC_getNvars](#) ()
Returns no. of variables (Nspec +1)
- int [TC_getSnames](#) (int Nspec, char *snames)
Returns species names.
- int [TC_getSnameLen](#) ()
Returns length of species names.
- int [TC_getSpos](#) (const char *sname, const int slen)
Returns position a species in the list of species.
- int [TC_getSmass](#) (int Nspec, double *Wi)
Returns species molar weights.

7.10.1 Detailed Description

Species info.

7.10.2 Function Documentation**7.10.2.1 int TC_getSmass (int Nspec, double * Wi)**

Returns species molar weights.

Parameters

<i>Nspec</i>	: no. of species
--------------	------------------

Returns

W_i : array of species molar weights [kg/kmol]=[g/mol]

References TC_Nspec_, and TC_sMass_.

7.10.2.2 int TC_getSnames (int Nspec, char * snames)

Returns species names.

Parameters

<i>Nspec</i>	: no. of species
--------------	------------------

Returns

snames: array of characters containing species names, each name is LENGTHOFSPECNAME characters

References LENGTHOFSPECNAME, TC_Nspec_, and TC_sNames_.

7.10.2.3 int TC_getSpos (const char * sname, const int slen)

Returns position a species in the list of species.

Parameters

<i>sname</i>	: string containing the name of the species
<i>slen</i>	: length of species "sname" name

Returns

position of species sname in the list of species, 0...(Nspec -1)

References LENGTHOFSPECNAME, TC_Nspec_, and TC_sNames_.

7.11 TC_src.c File Reference

Source term and Jacobian functions.

Functions

- int TCDND_getSrc (double *scal, int Nvars, double *omega)

Returns dimensional/non-dimensional source term for

$$\frac{\partial T}{\partial t} = \omega_0, \frac{\partial Y_i}{\partial t} = \omega_i,$$

based on temperature T and species mass fractions Y 's.

- int [TC_getSrc](#) (double *scal, int Nvars, double *omega)

Returns source term for

$$\frac{\partial T}{\partial t} = \omega_0, \frac{\partial Y_i}{\partial t} = \omega_i,$$

based on temperature T and species mass fractions Y 's.

- int [TCDND_getSrcCons](#) (double *scal, int Nvars, double *omega)

Returns source term (dimensional/non-dimensional) for

$$\frac{\partial \rho}{\partial t} = \omega_0, \rho \frac{\partial Y_i}{\partial t} = \omega_i,$$

based on ρ and Y 's.

- int [TC_getSrcCons](#) (double *scal, int Nvars, double *omega)

Returns source term for

$$\frac{\partial \rho}{\partial t} = \omega_0, \rho \frac{\partial Y_i}{\partial t} = \omega_i,$$

based on ρ and Y 's.

- int [TCDND_getJacTYNm1anl](#) (double *scal, int Nspec, double *jac)

Computes analytical Jacobian (dimensional/non-dimensional) for the system $(T, Y_1, Y_2, \dots, Y_{N-1})$ based on temperature T and species mass fractions Y 's.

- int [TC_getJacTYNm1anl](#) (double *scal, int Nspec, double *jac)

Computes analytical Jacobian for the system $(T, Y_1, Y_2, \dots, Y_{N-1})$ based on T and Y 's.

- int [TCDND_getJacTYNanl](#) (double *scal, int Nspec, double *jac)

Computes analytical Jacobian (dimensional/non-dimensional) for the system $(T, Y_1, Y_2, \dots, Y_N)$ based on T and Y 's.

- int [TC_getJacTYNanl](#) (double *scal, int Nspec, double *jac)

Computes analytical Jacobian for the system $(T, Y_1, Y_2, \dots, Y_N)$ based on T and Y 's.

- int [TCDND_getJacTYNm1](#) (double *scal, int Nspec, double *jac, unsigned int useJacAnl)

Computes (analytical or numerical) Jacobian (dimensional/non-dimensional) for the system $(T, Y_1, Y_2, \dots, Y_{N-1})$ based on temperature T and species mass fractions Y 's.

- int [TC_getJacTYNm1](#) (double *scal, int Nspec, double *jac, unsigned int useJacAnl)

Computes (analytical or numerical) Jacobian for the system $(T, Y_1, Y_2, \dots, Y_{N-1})$ based on T and Y 's.

- int [TCDND_getJacTYN](#) (double *scal, int Nspec, double *jac, unsigned int useJacAnl)

Computes (analytical or numerical) Jacobian for the system $(T, Y_1, Y_2, \dots, Y_N)$ based on T and Y 's.

- int [TC_getJacTYN](#) (double *scal, int Nspec, double *jac, unsigned int useJacAnl)

Computes (analytical or numerical) Jacobian for the system $(T, Y_1, Y_2, \dots, Y_N)$ based on T and Y 's.

- int [TC_getJacRPTYN](#) (double *scal, int Nspec, double *jac, unsigned int useJac-Anl)

Computes (analytical) Jacobian for the system $(\rho, P, T, Y_1, Y_2, \dots, Y_N)$ based on T and Y 's.

- int [TC_getJacRPTYNanl](#) (double *scal, int Nspec, double *jac)

Computes analytical Jacobian for the system $(\rho, P, T, Y_1, Y_2, \dots, Y_N)$ based on T and Y 's.

- int [TC_getJacRPTYNnum](#) (double *scal, int Nspec, double *jac)

Computes numerical Jacobian for the system $(\rho, P, T, Y_1, Y_2, \dots, Y_N)$ based on T and Y 's.

7.11.1 Detailed Description

Source term and Jacobian functions.

7.12 TC_thermo.c File Reference

Equation of state and thermodynamic functions.

Functions

- int [TCDND_getRhoMixMs](#) (double *scal, int Nvars, double *rhomix)

Computes density based on temperature and species mass fractions using the equation of state. Input temperature is normalized, output density also normalized before exit.

- int [TC_getRhoMixMs](#) (double *scal, int Nvars, double *rhomix)

Computes density based on temperature and species mass fractions using the equation of state.

- int [TCDND_getRhoMixMI](#) (double *scal, int Nvars, double *rhomix)

Computes density based on temperature and species mole fractions using the equation of state. Input temperature is normalized, output density also normalized before exit.

- int [TC_getRhoMixMI](#) (double *scal, int Nvars, double *rhomix)

Computes density based on temperature and species mole fractions using the equation of state.

- int [TCDND_getTmixMs](#) (double *scal, int Nvars, double *Tmix)

Computes temperature based on density and species mass fractions using the equation of state. Input density is normalized, output temperature also normalized before exit.

- int [TC_getTmixMs](#) (double *scal, int Nvars, double *Tmix)
Computes temperature based on density and species mass fractions using the equation of state.
- int [TCDND_getTmixMI](#) (double *scal, int Nvars, double *Tmix)
Computes temperature based on density and species mole fractions using the equation of state. Input density is normalized, output temperature also normalized before exit.
- int [TC_getTmixMI](#) (double *scal, int Nvars, double *Tmix)
Computes temperature based on density and species mole fractions using the equation of state.
- int [TCDND_getMs2CpMixMs](#) (double *scal, int Nvars, double *cpmix)
Computes mixture specific heat at constant pressure based on temperature and species mass fractions. Input temperature is normalized, output specific heat is also normalized before exit.
- int [TC_getMs2CpMixMs](#) (double *scal, int Nvars, double *cpmix)
Computes mixture specific heat at constant pressure based on temperature and species mass fractions.
- int [TCDND_getMs2CvMixMs](#) (double *scal, int Nvars, double *cvmix)
Computes mixture specific heat at constant volume based on temperature and species mass fractions. Input temperature is normalized, output specific heat is also normalized before exit.
- int [TC_getMs2CvMixMs](#) (double *scal, int Nvars, double *cvmix)
Computes mixture specific heat at constant volume based on temperature and species mass fractions.
- int [TCDND_getMI2CpMixMI](#) (double *scal, int Nvars, double *cpmix)
Computes mixture heat capacity at constant pressure based on temperature and species mole fractions. Input temperature is normalized, output specific heat is also normalized before exit.
- int [TC_getMI2CpMixMI](#) (double *scal, int Nvars, double *cpmix)
Computes mixture specific heat at constant pressure based on temperature and species mole fractions.
- int [TCDND_getCpSpecMs](#) (double t, int Nspec, double *cpi)
Computes species specific heat at constant pressure based on temperature. Input temperature is normalized, output specific heats are also normalized before exit.
- int [TC_getCpSpecMs](#) (double t, int Nspec, double *cpi)
Computes species specific heat at constant pressure based on temperature.
- int [TCDND_getCpSpecMI](#) (double t, int Nspec, double *cpi)
Computes species heat capacities at constant pressure based on temperature. Input temperature is normalized, output heat capacities are also normalized before exit.
- int [TC_getCpSpecMI](#) (double t, int Nspec, double *cpi)
Computes species heat capacities at constant pressure based on temperature.
- int [TCDND_getMs2HmixMs](#) (double *scal, int Nvars, double *hmix)

Computes mixture specific enthalpy based on temperature and species mass fractions. Input temperature is normalized, output enthalpy is normalized before exit.

- int [TC_getMs2HmixMs](#) (double *scal, int Nvars, double *hmix)

Computes mixture specific enthalpy based on temperature and species mass fractions.

- int [TCDND_getMI2HmixMI](#) (double *scal, int Nvars, double *hmix)

Computes mixture molar enthalpy based on temperature and species mole fractions. Input temperature is normalized, output enthalpy is normalized before exit.

- int [TC_getMI2HmixMI](#) (double *scal, int Nvars, double *hmix)

Computes mixture molar enthalpy based on temperature and species mole fractions.

- int [TCDND_getHspecMs](#) (double t, int Nspec, double *hi)

Computes species specific enthalpies based on temperature. Input temperature is normalized, output enthalpies are also normalized before exit.

- int [TC_getHspecMs](#) (double t, int Nspec, double *hi)

Computes species specific enthalpies based on temperature.

- int [TCDND_getHspecMI](#) (double t, int Nspec, double *hi)

Computes species molar enthalpies based on temperature. Input temperature is normalized, output enthalpies are also normalized before exit.

- int [TC_getHspecMI](#) (double t, int Nspec, double *hi)

Computes species molar enthalpies based on temperature.

- int [TC_getCpSpecMsFcn](#) (double t, double *cpi)
- int [TC_getCpSpecMs1Fcn](#) (double t, int i, double *cpi)
- int [TC_getCpSpecMIFcn](#) (double t, double *cpi)
- int [TC_getCpSpecMI1Fcn](#) (double t, int i, double *cpi)
- int [TC_getCpSpecMsTab](#) (double t1, double *cpi)
- int [TC_getCpSpecMITab](#) (double t1, double *cpi)
- int [TC_getCpMixMsP](#) (double *scal, int Nvars, double *cpmix)
- int [TC_getCpSpecMsP](#) (double t, int Nspec, double *cpi)
- int [TC_getCpSpecMsPFcn](#) (double t, double *cpi)
- int [TC_getCpSpecMs1PFcn](#) (double t, int i, double *cpi)
- int [TC_getCpSpecMsPtab](#) (double t1, double *cpi)
- int [TC_getHspecMsFcn](#) (double t, double *hi)
- int [TC_getHspecMIFcn](#) (double t, double *hi)
- int [TC_getHspecMsTab](#) (double t1, double *hi)
- int [TC_getHspecMITab](#) (double t1, double *hi)

7.12.1 Detailed Description

Equation of state and thermodynamic functions.

7.13 TC_utils.c File Reference

Various utilities used by other functions.

Functions

- static double `fastIntPow` (double val, int exponent)

Much faster version of pow for small integers (considering that reactions are generally no more than 3rd order). The C version of pow requires a double exponent; pow is the slowest part of the code when used instead of fastIntPow.

7.13.1 Detailed Description

Various utilities used by other functions.

Index

- ATMPA
 - TC_params.h, [90](#)
- CALJO
 - TC_params.h, [90](#)
- EVOLT
 - TC_params.h, [90](#)
- Equation of state, [9](#)
 - TCDND_getRhoMixMI, [11](#)
 - TCDND_getRhoMixMs, [12](#)
 - TCDND_getTmixMI, [12](#)
 - TCDND_getTmixMs, [13](#)
 - TC_getRhoMixMI, [10](#)
 - TC_getRhoMixMs, [10](#)
 - TC_getTmixMI, [11](#)
 - TC_getTmixMs, [11](#)
- Initialization, [23](#)
 - TC_createTables, [23](#)
 - TC_initChem, [23](#)
 - TC_kmodint_, [24](#)
 - TC_setRefVal, [24](#)
 - TC_setThermoPres, [25](#)
- Jacobians, [26](#)
 - TCDND_getJacTYN, [30](#)
 - TCDND_getJacTYNanI, [30](#)
 - TCDND_getJacTYNm1, [30](#)
 - TCDND_getJacTYNm1anI, [31](#)
 - TC_getJacRPTYN, [27](#)
 - TC_getJacRPTYNanI, [27](#)
 - TC_getJacRPTYNnum, [27](#)
 - TC_getJacTYN, [28](#)
 - TC_getJacTYNanI, [28](#)
 - TC_getJacTYNm1, [29](#)
 - TC_getJacTYNm1anI, [29](#)
- KBOLT
 - TC_params.h, [90](#)
- LENGTHOFELEMENTNAME
 - TC_params.h, [90](#)
- LENGTHOFSPECNAME
 - TC_params.h, [90](#)
- MAX
 - TC_params.h, [90](#)
- MIN
 - TC_params.h, [91](#)
- Max. no. of parameters, [35](#)
- NAVOG
 - TC_params.h, [91](#)
- NSPECREACMAX
 - TC_params.h, [91](#)
- NTHRDBMAX
 - TC_params.h, [91](#)
- NUMBEROFELEMENTINSPEC
 - TC_params.h, [91](#)
- No. of reactions, [36](#)
- No. of species, variables, etc., [37](#)
- REACBALANCE
 - TC_params.h, [91](#)
- RUNIV
 - TC_params.h, [91](#)
- Source terms, [32](#)
 - TCDND_getSrc, [33](#)
 - TCDND_getSrcCons, [34](#)
 - TC_getSrc, [32](#)
 - TC_getSrcCons, [33](#)
- TCDND_getCpSpecMI
 - Thermodynamic properties, [18](#)
- TCDND_getCpSpecMs
 - Thermodynamic properties, [19](#)
- TCDND_getHspecMI
 - Thermodynamic properties, [19](#)
- TCDND_getHspecMs
 - Thermodynamic properties, [20](#)
- TCDND_getJacTYN
 - Jacobians, [30](#)
- TCDND_getJacTYNanI

- Jacobians, [30](#)
- TCDND_getJacTYNm1
 - Jacobians, [30](#)
- TCDND_getJacTYNm1anl
 - Jacobians, [31](#)
- TCDND_getMl2CpMixMl
 - Thermodynamic properties, [20](#)
- TCDND_getMl2HmixMl
 - Thermodynamic properties, [20](#)
- TCDND_getMl2Ms
 - TC_interface.h, [71](#)
 - TC_mlms.c, [87](#)
- TCDND_getMl2Wmix
 - TC_interface.h, [71](#)
 - TC_mlms.c, [87](#)
- TCDND_getMs2Cc
 - TC_interface.h, [72](#)
 - TC_mlms.c, [88](#)
- TCDND_getMs2CpMixMs
 - Thermodynamic properties, [21](#)
- TCDND_getMs2CvMixMs
 - Thermodynamic properties, [21](#)
- TCDND_getMs2HmixMs
 - Thermodynamic properties, [22](#)
- TCDND_getMs2Ml
 - TC_interface.h, [72](#)
 - TC_mlms.c, [88](#)
- TCDND_getMs2Wmix
 - TC_interface.h, [73](#)
 - TC_mlms.c, [88](#)
- TCDND_getRhoMixMl
 - Equation of state, [11](#)
- TCDND_getRhoMixMs
 - Equation of state, [12](#)
- TCDND_getSrc
 - Source terms, [33](#)
- TCDND_getSrcCons
 - Source terms, [34](#)
- TCDND_getTxC2RRml
 - TC_interface.h, [73](#)
 - TC_rr.c, [98](#)
- TCDND_getTxC2RRms
 - TC_interface.h, [73](#)
 - TC_rr.c, [98](#)
- TCDND_getTY2RRml
 - TC_interface.h, [74](#)
- TC_rr.c, [99](#)
- TCDND_getTY2RRms
 - TC_interface.h, [74](#)
 - TC_rr.c, [99](#)
- TCDND_getTmixMl
 - Equation of state, [12](#)
- TCDND_getTmixMs
 - Equation of state, [13](#)
- TC_chg.c, [43](#)
 - TC_chgArhenFor, [44](#)
 - TC_chgArhenForBack, [44](#)
 - TC_chgArhenRev, [44](#)
 - TC_chgArhenRevBack, [45](#)
- TC_chgArhenFor
 - TC_chg.c, [44](#)
 - TC_interface.h, [62](#)
- TC_chgArhenForBack
 - TC_chg.c, [44](#)
 - TC_interface.h, [62](#)
- TC_chgArhenRev
 - TC_chg.c, [44](#)
 - TC_interface.h, [62](#)
- TC_chgArhenRevBack
 - TC_chg.c, [45](#)
 - TC_interface.h, [63](#)
- TC_createTables
 - Initialization, [23](#)
- TC_defs.h, [45](#)
 - TC_errorINI, [52](#)
 - TC_errorMSG, [52](#)
 - TC_getReacRates, [53](#)
- TC_errorINI
 - TC_defs.h, [52](#)
 - TC_init.c, [54](#)
- TC_errorMSG
 - TC_defs.h, [52](#)
 - TC_init.c, [54](#)
- TC_getArhenFor
 - TC_interface.h, [63](#)
 - TC_rr.c, [94](#)
- TC_getArhenRev
 - TC_interface.h, [63](#)
 - TC_rr.c, [94](#)
- TC_getCpSpecMl
 - Thermodynamic properties, [15](#)
- TC_getCpSpecMs

- Thermodynamic properties, 15
- TC_getHspecMI
 - Thermodynamic properties, 16
- TC_getHspecMs
 - Thermodynamic properties, 16
- TC_getJacRPTYN
 - Jacobians, 27
- TC_getJacRPTYNanl
 - Jacobians, 27
- TC_getJacRPTYNnum
 - Jacobians, 27
- TC_getJacTYN
 - Jacobians, 28
- TC_getJacTYNanl
 - Jacobians, 28
- TC_getJacTYNm1
 - Jacobians, 29
- TC_getJacTYNm1anl
 - Jacobians, 29
- TC_getMI2CpMixMI
 - Thermodynamic properties, 16
- TC_getMI2HmixMI
 - Thermodynamic properties, 17
- TC_getMI2Ms
 - TC_interface.h, 64
 - TC_mlms.c, 84
- TC_getMI2Wmix
 - TC_interface.h, 64
 - TC_mlms.c, 85
- TC_getMs2Cc
 - TC_interface.h, 65
 - TC_mlms.c, 85
- TC_getMs2CpMixMs
 - Thermodynamic properties, 17
- TC_getMs2CvMixMs
 - Thermodynamic properties, 18
- TC_getMs2HmixMs
 - Thermodynamic properties, 18
- TC_getMs2MI
 - TC_interface.h, 65
 - TC_mlms.c, 86
- TC_getMs2Wmix
 - TC_interface.h, 66
 - TC_mlms.c, 86
- TC_getReacRates
 - TC_defs.h, 53
- TC_rr.c, 94
- TC_getRfrb
 - TC_interface.h, 66
 - TC_rr.c, 95
- TC_getRhoMixMI
 - Equation of state, 10
- TC_getRhoMixMs
 - Equation of state, 10
- TC_getRops
 - TC_interface.h, 67
 - TC_rr.c, 95
- TC_getSmass
 - TC_interface.h, 67
 - TC_spec.c, 100
- TC_getSnames
 - TC_interface.h, 67
 - TC_spec.c, 101
- TC_getSpos
 - TC_interface.h, 68
 - TC_spec.c, 101
- TC_getSrc
 - Source terms, 32
- TC_getSrcCons
 - Source terms, 33
- TC_getStoiCoef
 - TC_interface.h, 68
 - TC_rr.c, 95
- TC_getStoiCoefReac
 - TC_interface.h, 68
 - TC_rr.c, 96
- TC_getTXC2RRml
 - TC_interface.h, 69
 - TC_rr.c, 96
- TC_getTXC2RRms
 - TC_interface.h, 69
 - TC_rr.c, 97
- TC_getTY2RRml
 - TC_interface.h, 70
 - TC_rr.c, 97
- TC_getTY2RRms
 - TC_interface.h, 70
 - TC_rr.c, 98
- TC_getTmixMI
 - Equation of state, 11
- TC_getTmixMs
 - Equation of state, 11

- TC_init.c, 53
 - TC_errorINI, 54
 - TC_errorMSG, 54
- TC_initChem
 - Initialization, 23
- TC_interface.h, 55
 - TCDND_getMI2Ms, 71
 - TCDND_getMI2Wmix, 71
 - TCDND_getMs2Cc, 72
 - TCDND_getMs2MI, 72
 - TCDND_getMs2Wmix, 73
 - TCDND_getTXC2RRml, 73
 - TCDND_getTXC2RRms, 73
 - TCDND_getTY2RRml, 74
 - TCDND_getTY2RRms, 74
 - TC_chgArhenFor, 62
 - TC_chgArhenForBack, 62
 - TC_chgArhenRev, 62
 - TC_chgArhenRevBack, 63
 - TC_getArhenFor, 63
 - TC_getArhenRev, 63
 - TC_getMI2Ms, 64
 - TC_getMI2Wmix, 64
 - TC_getMs2Cc, 65
 - TC_getMs2MI, 65
 - TC_getMs2Wmix, 66
 - TC_getRfrb, 66
 - TC_getRops, 67
 - TC_getSmass, 67
 - TC_getSnames, 67
 - TC_getSpos, 68
 - TC_getStoiCoef, 68
 - TC_getStoiCoefReac, 68
 - TC_getTXC2RRml, 69
 - TC_getTXC2RRms, 69
 - TC_getTY2RRml, 70
 - TC_getTY2RRms, 70
 - TC_removeReaction, 70
- TC_kmodint.c, 75
 - checkelemint, 81
 - extractWordLeft, 81
 - extractWordLeftauxline, 81
 - getreactions, 82
 - setperiodictable, 82
 - tab2space, 82
- TC_kmodint_
 - Initialization, 24
- TC_mlms.c, 83
 - TCDND_getMI2Ms, 87
 - TCDND_getMI2Wmix, 87
 - TCDND_getMs2Cc, 88
 - TCDND_getMs2MI, 88
 - TCDND_getMs2Wmix, 88
 - TC_getMI2Ms, 84
 - TC_getMI2Wmix, 85
 - TC_getMs2Cc, 85
 - TC_getMs2MI, 86
 - TC_getMs2Wmix, 86
- TC_params.h, 89
 - ATMPA, 90
 - CALJO, 90
 - EVOLT, 90
 - KBOLT, 90
 - LENGTHOFELEMNAME, 90
 - LENGTHOFSPECNAME, 90
 - MAX, 90
 - MIN, 91
 - NAVOG, 91
 - NSPECREACMAX, 91
 - NTHRDBMAX, 91
 - NUMBEROFELEMINSPEC, 91
 - REACBALANCE, 91
 - RUNIV, 91
- TC_removeReaction
 - TC_interface.h, 70
- TC_rr.c, 92
 - TCDND_getTXC2RRml, 98
 - TCDND_getTXC2RRms, 98
 - TCDND_getTY2RRml, 99
 - TCDND_getTY2RRms, 99
 - TC_getArhenFor, 94
 - TC_getArhenRev, 94
 - TC_getReacRates, 94
 - TC_getRfrb, 95
 - TC_getRops, 95
 - TC_getStoiCoef, 95
 - TC_getStoiCoefReac, 96
 - TC_getTXC2RRml, 96
 - TC_getTXC2RRms, 97
 - TC_getTY2RRml, 97
 - TC_getTY2RRms, 98
- TC_setRefVal

- Initialization, 24
- TC_setThermoPres
 - Initialization, 25
- TC_spec.c, 100
 - TC_getSmass, 100
 - TC_getSnames, 101
 - TC_getSpos, 101
- TC_src.c, 101
- TC_thermo.c, 103
- TC_utils.c, 106
- Thermodynamic properties, 14
 - TCDND_getCpSpecMI, 18
 - TCDND_getCpSpecMs, 19
 - TCDND_getHspecMI, 19
 - TCDND_getHspecMs, 20
 - TCDND_getMI2CpMixMI, 20
 - TCDND_getMI2HmixMI, 20
 - TCDND_getMs2CpMixMs, 21
 - TCDND_getMs2CvMixMs, 21
 - TCDND_getMs2HmixMs, 22
 - TC_getCpSpecMI, 15
 - TC_getCpSpecMs, 15
 - TC_getHspecMI, 16
 - TC_getHspecMs, 16
 - TC_getMI2CpMixMI, 16
 - TC_getMI2HmixMI, 17
 - TC_getMs2CpMixMs, 17
 - TC_getMs2CvMixMs, 18
 - TC_getMs2HmixMs, 18
- checkelemenlist
 - TC_kmodint.c, 81
- copyright.h, 43
- element, 39
- elemtable, 39
- extractWordLeft
 - TC_kmodint.c, 81
- extractWordLeftauxline
 - TC_kmodint.c, 81
- getreactions
 - TC_kmodint.c, 82
- reaction, 40
- setperiodictable
 - TC_kmodint.c, 82
- species, 41
- tab2space
 - TC_kmodint.c, 82