Software Development Kits Documentation Set

Version 2016b

Thermo-Calc Software AB or Thermo-Calc Software, Inc..

Norra Stationsgatan 93, SE-113 64 Stockholm, Sweden

+46 8 545 959 30

documentation@thermocalc.com

www.thermocalc.com

Contents

TQ-Interface 1

Introduction to the TQ-Interface 2

Introduction to the TQ-Interface 3

The Subroutines and Functions 4

Installing and Using the TQ-Interface 6

Installing the TQ-Interface and Examples 7

Using this Guide 9

Programming Languages 9

FORTRAN 10

C-Interface 10

Basic Concepts 11

Naming Components, Phases and Constituents 11

Naming Conventions 11

About Adaptive Interpolation Schemes 13

Initialization Subroutines 14

Units for TQSSU and TQGSU 14

Legal Input/Output Options for TQSIO and TQGIO 15

TQINI3 15

TQINI 16

TQSIO 17

TQGIO 17

TQRFIL 18

TQSSU 19

TQGSU 20

TQSAME 21

TQGVER 21

System Data Manipulation Subroutines 23

Legal Component Status 24

Legal Phase Status 24

TQGNC 24

TQSCOM 25

TQGCOM 26

TQGSCI 27

TQGNP 27

TQGPN 28

TQGPI 29

TQGPCN 29

TQGPCI 30

TQGCCF 31

TQGPCS 32

TQGNPC 32

TQCSSC 33

TQGSSC 34

TQCSP 35

TQGSP 36

TQSETR 36

TQPACS 37

TQSGA 38

TQGGA 38

Condition, Stream and Segment Subroutines 40

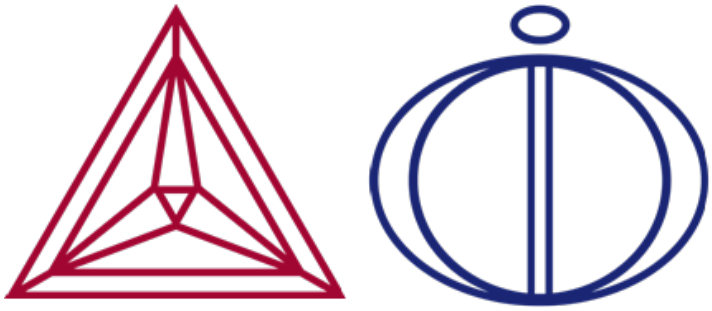
Possible State Variables to Set Conditions in TQSETC 40
TQSETC 42
TQREMC 43
TQSCURC 44
TQREMAC 45
TQRESTC 45
TQCSTM 46
TQSSC 46
TQSSIC 47
TQDSTM 49
TQNSEG 49
TQSSEG 50
Calculations and Results Subroutines 51
State Variables Available for TQGETV and TQGET1 51
TQCE 54
TQCEG 56
TQGETV and TQGET1 57
TQGMU 59
TQGGM 60
TQGPD 60
TQGDF2 62
TQGSE 64
Troubleshooting Subroutines 66
TQLS 67
TQLC 67
TQLE 68
TQFASV 68
TQSDMC 69
TQSSPC 69
TQSSV 70
TQPINI 71
TQSNL 71
TQSMNG 72
TQSECO 73
ST1ERR 73
ST2ERR 74
SG1ERR or TQG1ERR 75
SG2ERR or TQG2ERR 75
SG3ERR or TQG3ERR 76
RESERR or TQRSERR 77
TQSP3F 78
Extra Subroutines–Phase Properties 79
TQGMA, TQGMB and TQGMC 80
TQGMDY 81
TQGMOB 82
TQSTP 83
TQSYF 83
TQGSSPI 84
TQCMOBA and TQCMOBB 84
TQDGYY 85
TQGPHP 86
TQX2Y 87
TQGMDX 89

Database Subroutines 91
TQGDBN 91
TQOPDB 92
TQLIDE 93
TQAPDB 93
TQDEFEL 94
TQREJEL 94
TQREJPH 95
TQRESPH 95
TQLISPH 96
TQLISSF 97
TQGDAT 97
TQREJSY 98
Adaptive Interpolation Schemes 99
TQIPS_INIT_TOP 99
TQIPS_INIT_BRANCH 100
TQIPS_INIT_FUNCTION 103
TQIPS_GET_VALUE 104
TQIPS_WRITE_IPS_DATA_TO_FILE 105
TQIPS_READ_IPS_DATA_FROM_FILE 106
TQIPS_GET_MEMORY_USAGE 107
Composition Set Reordering Routines 108
TQROINIT 108
TQSETRX 109
TQORDER 109
TQLROX 110
Compiler Settings 111
Compiling FORTRAN code 111
Windows: Visual Studio 2010, Intel FORTRAN Composer 12 111
Linux: GNU compiler version 4.4 111
Linux: Intel FORTRAN Compiler 112
Compiling C code 112
Windows: Visual Studio 2010 112
Linux: GNU compiler version 4.4 113
TC-Toolbox for MATLAB® 1
Introduction to the TC-Toolbox for MATLAB® 2
About TC-Toolbox for MATLAB® 3
Installing TC-Toolbox for MATLAB® 4
Test the Installation 4
TC-Toolbox for MATLAB Examples 4
Commands in TC-Toolbox for MATLAB® 6
tc_root 6
tc_database 7
tc_system 7
tc_util 10
tc_ges5 10
dic_dictra 10
TC-API 1
About the TC-API Programmer's Guide 1
TC-API Programming Guide (PDF) 2

TQ-Interface

SDK Programmer's Guide

Thermo-Calc 2016b



Thermo-Calc Software

Introduction to the TQ-Interface

In this section:

Introduction to the TQ-Interface *3*

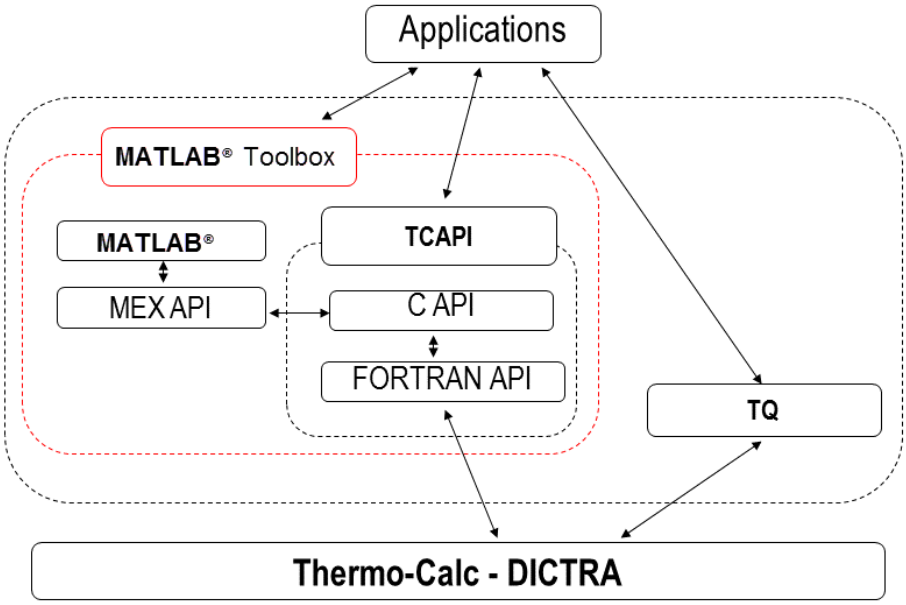
The Subroutines and Functions *4*

Introduction to the TQ-Interface

TQ-Interface is an application programming interface for Thermo-Calc, a general software package for multicomponent phase equilibrium calculations. TQ-Interface is for application programmers to write programs using the Thermo-Calc kernel. With this programming interface, it is easy to make Thermo-Calc an integral part of application programs such as those for process simulation, microstructure evolution modeling and materials property prediction.

Thermo-Calc APIs

Interfacing with the Thermo-Calc Engine



The thermodynamic properties and phase equilibrium data that can be obtained by using the TQ-Interface include Gibbs energy, enthalpy, entropy, heat capacity, first and second derivatives of Gibbs energy with respect to composition, chemical potential, phase amount, phase composition, partition coefficients, liquidus or solidus points, invariant temperature, heat of reaction, adiabatic combustion temperature, and volume, etc.

Through appending the mobility databases into the workspace, you can also obtain assessed mobility or diffusivity data via the TQ-Interface. The TQ-Interface can also be used to predict metastable or non-equilibrium states by changing the status of the phases under consideration.

The TQ-Interface is available for both Windows and Linux platforms. It is supplied in the form of DLLs (*Dynamically Linked Libraries*) meaning there is no need to recompile existing application programs when a new version of TQ-Interface is released.

TQ-Interface is written in FORTRAN as many software packages for scientific calculations are developed in this language. *The Subroutines and Functions* on the next page topic outlines categories of what is available in the TQ-Interface.

The computer language to implement application programs is not restricted to FORTRAN, for example a GUI application written in C++ can realize its various functionalities by using TQ-Interface subroutines with the appropriate calling conventions.

Also see *Programming Languages* on page 9.

The Subroutines and Functions

The FORTRAN subroutines and functions available in the TQ-Interface can be classified into categories based on purpose:

1. *Initialization Subroutines* on page 14. For example, initializing the workspace, reading the thermodynamic data files, setting default units for thermodynamic quantities, selecting the input and output options, changing the program input and output units
2. *System Data Manipulation Subroutines* on page 23. For example, identifying system components, phases, and constituents, redefining the system components, changing the status of components and phases or the system reference state.
3. *Condition, Stream and Segment Subroutines* on page 40. For example, defining conditions for an equilibrium calculation, setting conditions for a thermodynamic equilibrium calculation, and setting a new equilibrium segment.
4. *Calculations and Results Subroutines* on page 51. For example, calculate equilibriums, get molar Gibbs energy values, and calculate interfacial energy between a matrix phase and a precipitate phase.
5. *Troubleshooting Subroutines* on page 66. For example, reinitiate the calculation workspace, set error codes and messages, or set equilibrium calculation options.

Essentially, only subroutines 1, 3, and 4 are required to use the TQ Interface. In the simplest case, only one or two subroutines are needed from each category.

Additional subroutines are grouped as follows:

- *Extra Subroutines–Phase Properties* on page 79. For example, get Gibbs energy of a phase, mobility of a species in a phase, or check if mobility data for a phase is available.
- *Database Subroutines* on page 91 For example, get lists of database names, reject a selected element and get data from the selected database.
- *Adaptive Interpolation Schemes* on page 99. For example, define a function or state variable to be interpolated and get statistics on the usage of the interpolation scheme.

- *Composition Set Reordering Routines* on page 108. For example, initialize IWSR workspace and set ideal composition in a phase.

Installing and Using the TQ-Interface

If you have not used Thermo-Calc before, start with the *Basic Concepts* on page 11. Several simple application examples are given in the installed SDK folder.

If you are an experienced Thermo-Calc user you can start by copying a suitable example. You make it work for problems by changing, adding or deleting some callings to TQ-Interface subroutines and functions.

It is strongly recommended that at least one or two examples should be compiled and linked (and tested) to make sure the linked executables can be run successfully.

In this section:

Installing the TQ-Interface and Examples 7

Using this Guide 9

Programming Languages 9

Basic Concepts 11

Naming Components, Phases and Constituents 11

About Adaptive Interpolation Schemes 13

Installing the TQ-Interface and Examples

The TQ-Interface requires an additional license key, which is purchased along with the Thermo-Calc software/database package. For both Windows and Linux platforms, the TQ-Interface is supplied as a dynamically linked library.

All the examples in this document are included in the SDK installation directory. For example, for a network installation on Windows, the directory is here:

`C:\Users\<username>\Documents\Thermo-Calc\<version>\SDK\TQ\<Windows>`

On Windows, once Thermo-Calc and the SDKs are installed go to **Start → All Programs** or **All Apps → Thermo-Calc** and click **SDK** to open the folders.

For installation and directory locations, see the *Thermo-Calc Installation Guide*.

TQ-Interface Examples

Example Name	Description
TQEX01	This sample program shows how to retrieve data from a Thermo-Calc data file, then defines a set of conditions for a single equilibrium calculation, gets the equilibrium phases and their amounts and compositions. The method of calculating the liquidus and solidus temperature is also demonstrated.
TQEX02	This sample program calculates the To line for the fcc and bcc phase in the Fe-C system.
TQEX03	This sample program simulates the non-equilibrium solidification under the Scheil-Guilliver condition.
TQEX04	This sample program simulates the non-equilibrium solidification under the Scheil-Guilliver condition.
TQEX05	This example demonstrates how to use stream calculation to get the enthalpy of a reaction, i.e., the enthalpy difference between the reaction products at one temperature and the reactants at another temperature. By setting the enthalpy of reaction to zero, the adiabatic temperature can be easily calculated.
TQEX06	This example demonstrates how to use stream calculation to obtain the chill factors in the steelmaking industry.[1]O.Kubaschewski and C.B. Alock, Metellurgical Thermochemistry, 1979, Page 211.
TQEX07	This sample program calculates the A3 temperature of a steel and determines

Example Name	Description
	the influence of each alloying element on this temperature. It demonstrates that some very special quantities, such as the composition derivative of temperature, can be obtained easily via the TQ interface.
TQEX08	This sample program displays the diffusion matrix in a multicomponent system.
TQEX09	This sample program show how to retrieve Gibbs energy, Gibbs energy derivatives and mobilities.
TQEX10	This sample program is the same as Example 9 except that it demonstrates how to convert mole fractions to site fractions and first derivatives of Gm w.r.t. site fractions to that w.r.t. mole fractions.
TQEX11	This sample program shows how to get information about the paraequilibrium transformation from Fcc to Bcc in a steel.
TQEX12	This sample program demonstrates how to use subroutines getting system data from a database and how to restart new calculation on a different system in the same application program.
TQEX13	This sample program demonstrates that the number of phases can increase due to the use of global mininization for equilibrium calculation during which additional composition set(s) can be added automatically if a miscibility gap is detected.
TQEX14	This sample program show how to use the functionality for setting how different composition sets should correspond to different compositions. For example, that in the Ni-Al system the composition set fcc_l12#1 should correspond to gamma and fcc_l12#2 to gamma-prime.
TQEX15	TQ library example to illustrate the use the adaptive interpolation scheme. This example calculates the liquidus temperature in a part of the C-CR-FE system and displays a selection of the results.
	The MPI examples only show how the TQ-Interface can be used in applications together with MPI. Thermo-Calc Software AB or Thermo-Calc Software, Inc. is not available to answer support questions related to MPI.
MPExample1	This is an MPI (Message Passing Interface) example that calculates the Gibbs energy for a composition grid C with a density of "npoints" at 1273K in the Mn-Ni-Fe system.

Example Name	Description
MPExample2	This is an MPI (Message Passing Interface) example where a set of equilibrium calculations are distributed over all processes. Then the Gibbs energy of the system is retrieved and collected in a single vector in the master process.

Using this Guide

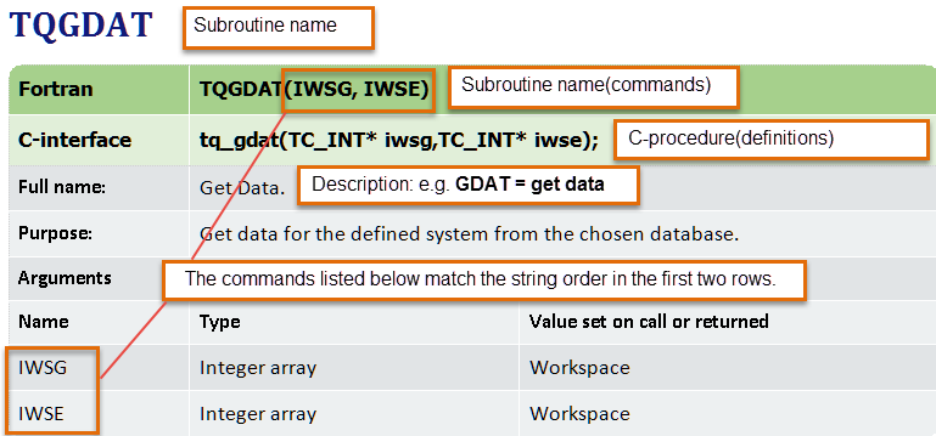
The topic names in this guide are the same as the FORTRAN routine names. Also included in each topic are details for both the FORTRAN and C programming languages.

Also see *Programming Languages* below.

Note the following conventions to distinguish between the programming languages.

- Routines starting with **TQ**XXX, for example, *TQSSU*, are in the Fortran interface
- Routines starting with **tq_**xxxx, for example *tq_ssu*, are in the C-interface.
- In Fortran, all routines are subroutines and do not return any values except where explicitly declared as functions.
- All the C procedures are declared as void and do not return any values except where explicitly otherwise declared.

An example of how to read the subroutine definitions.



Programming Languages

You can program your TQ-library with the FORTRAN or C programming languages.

FORTRAN

No special consideration is needed when interfacing the TQ-library with a program written in FORTRAN; the main core of the TQ-library is written in FORTRAN. By default all parameters are passed to routines by reference, except for strings, which are passed by descriptor.

C-Interface

The C-interface acts as a translation layer in between the calling C-program and the underlying FORTRAN TQ-library.

For a C-interface, the default parameter passing mechanism is by *value* and not by reference. Some decisions must be made as to how parameters, which are updated in the TQ-Interface, are then passed into the library. For example:

- The *C procedures* are defined in the file **tqroot.h** which should be included in the procedures using the library calls in C.
- The **tqroot.h** file also includes the file **tc_data_defs.h** where the datatypes are defined.

The definition of some of these data types vary depending on what platform and compiler is used. It is important to define these and for the definitions to be correctly set (see *Compiler Settings* on page 111).

Common C-Procedure Definitions

The commonly used definitions in the C-interface are listed in the table. Note that:

- TC_INT and TC_FLOAT are used when only the value of the variables is necessary to pass.
- TC_INT* and TC_FLOAT* are used when the variables are updated and values are returned within these.
- When a TC_STRING is updated, the allocated size of the string must be passed into the interface in a variable declared as TC_STRING_LENGTH.

Routine	Definitions
TC_INT	An integer of platform dependent length passed by value.
TC_INT*	Address of an integer of platform dependent length.
TC_FLOAT	A 64-bit real passed by value.
TC_FLOAT*	Address of a 64-bit real.

Routine	Definitions
TC_STRING	Address of a character string.
TC_STRING_LENGTH	An integer of platform dependent length passed by value defining the length of the string.

Basic Concepts

A thermodynamic system is made up of *components* and *phases*. A number of *state variables* define the properties and the relationships.

A **component** is a system-wide entity; sometimes it is specifically called a *system component*. A component has a unique name and some thermodynamic properties are associated with it, for example, its amount and activity or chemical potential. At equilibrium the activity or chemical potential of the components are the same in the whole system.

A **phase** is a system-wide entity, which has a composition expressed in the amounts of components, enthalpy content, a volume, and many other properties. The phase has *constituents* that may be different from the components. The **constituents** have a stoichiometry that can be expressed in terms of the components and possibly a charge. *Condensed phases* may have an internal structure like sub-lattices or clusters, and these clusters may be modeled as constituents.

Naming Components, Phases and Constituents

Naming Conventions

The name of a component, phase or constituent can be maximum of 24 characters and must start with a letter (A-Z or a-z) and contain only letters, digits and these special characters:

- underscore (_)
- full stop (.)
- parentheses (and)
- plus (+)
- minus (-)
- slash (/)

Components

The TQ Interface maintains a list of *components*. These are numbered sequentially from `1` up to the number of components.

A component has a name which can be identical to a chemical formula or any string of letters such as `h2o`, `c2h2cl_cis`, or `au3cu_cvm1`.

Several subroutines are available to get information about the components and to manipulate them, for

example:

- *TQGCOM* on page 26 returns the total number of components and all component names
- *TQGSCI* on page 27 returns the index of one component name
- *TQSCOM* on page 25 enables you to re-define the components.

The *component index* is used in most subroutines for defining conditions, etc.

Components can be suspended by *TQCSSC* on page 33, thus leaving gaps in the component list because suspending one component does not change the sequential numbering. The logical function *TQGSSC* on page 34 can be used to check if a specific component is suspended or not.

Phases

The TQ-Interface maintains a list of *phases*. These are numbered sequentially from `1` up to the number of phases in the system.

A phase has many properties and most importantly a list of constituents (see Phase Constituents). Subroutines are available to get information about the phases, for example:

- *TQGNP* on page 27 for the total number of phases
- *TQGPN* on page 28 for the name of a phase
- *TQGPI* on page 29 for the name of its index
- *TQGNPC* on page 32 for the number of phase constituents

Phases can be suspended or set dormant by *TQCSP* on page 35, thus leaving gaps in the list because suspending one phase does not change the sequential numbering. The logical function *TQGSP* on page 36 can be used to check if a specific phase is suspended or not.

Phase Constituents

The TQ Interface maintains a list of the constituents of each phase (the *phase constituents*). These are numbered sequentially from `1` up to the number of constituents in the phase. The number of constituents can be different in each phase. If a phase has sub-lattices, the numbering goes from the first constituent in the first sub-lattice over all sub-lattices to the last constituent in the last sub-lattice.

Subroutines are available to get information about the constituents, for example:

- *TQGNPC* on page 32 for the number of constituents of a phase
- *TQGPCN* on page 29 (or its index *TQGPCI* on page 30) for the name of a phase constituent

- *TQGPCS* on page 32 for the stoichiometry of a constituent expressed in terms of the components

About Adaptive Interpolation Schemes

A general dynamic interpolation scheme is implemented in the TQ-library. At a slight cost of accuracy, this scheme allows you to rapidly obtain equilibrium values for state variables and functions for many different values of a predefined set of conditions.

Multiple sets of conditions and requested variable values can be defined in order to obtain different values for different situations. These are stored internally as different branches.

The accuracy of the scheme can be adjusted by setting the number of steps in the composition/temperature/pressure space where the interpolation is performed.

For a given set of conditions (a *branch*), the scheme builds up an interpolation matrix within the bounds of the conditions that have been previously defined. As long as the subsequent condition values are kept within these limits, the returned values are calculated from the interpolation matrix. If the condition values are outside these limits then the scheme automatically extends the interpolation matrix. With this procedure the scheme extends the interpolation matrix so that it can return values from a growing range of conditions in composition, temperature and pressure.

For each set of condition values within a branch a unique identifying number is calculated. This number is used to find the correct position in the interpolation matrix using a *hash table*.

If the memory requirements to extend the interpolation exceeds the available memory, the nodes in the matrix that are less frequently used are removed to free up some memory.

For a general reference about the interpolation scheme, see : Larsson, Henrik, and Lars Höglund, 'A Scheme for More Efficient Usage of CALPHAD Data in Simulations', *Calphad*, 50 (2015), 1–5 http://dx.doi.org/10.1016/j.calphad.2015.04.007.

Initialization Subroutines

Purpose	Subroutine
Initialize TQ-Interface with user-specified database and temporary directories	*TQINI3* on the next page
Initialize TQ-Interface	*TQINI* on page 16
Set input/output option	*TQSIO* on page 17
Get input/output option	*TQGIO* on page 17
Read thermodynamic data file	*TQRFIL* on page 18
Set unit for a system quantity	*TQSSU* on page 19
Get unit for a system quantity	*TQGSU* on page 20
Check if the system is the same	*TQSAME* on page 21
Retrieve information about the TQ-library	*TQGVER* on page 21

Units for TQSSU and TQGSU

Quantity	Unit	Comment
Temperature	K, C, F	K=Kelvin (default); C=Celsius, calculated as K-273.15; F=Fahrenheit
Volume	M3	Cubic meter (default)
	L	Liter. Calculated as 0.001 M^3
	IN3	Cubic inch.
	FT3	Cubic feet
	USG	US gallon
Energy	J	Joule (default)
	Cal	Calories. Calculated as J/4.184
	BTU	British thermal units.
Pressure	Pa	Pascal (default)
	Psi	Pounds/sq. inch
	Bar	Bar. Calculated as 0.00001*Pa

Quantity	Unit	Comment
	Atm	Atmosphere. Calculated as Pa/101325
	Torr	Torricelli. Calculated as 758*Pa/101325
Mass	kg, g, lb	kg=Kilograms (default); g=Grams; lb=Pounds

Legal Input/Output Options for TQSIO and TQGIO

Option	Meaning	Default value
INPUT	Input unit	0
OUTPUT	Output unit	0
ERROR	Error output	0
LIST	List output	0

TQINI3

Fortran	**TQINI3(DATABASE_PATH, TEMP_PATH, NWSG, NWSE, IWSG, IWSE)**	
C-interface	**tq_ini3(TC_STRING database_path, TC_STRING temp_path, TC_INT nwsg, TC_INT nwse, TC_INT* iwsg, TC_INT* iwse);**	
Full name:	Initialize TQ-Interface with user-specified database and temporary directories. If a GES file is used (i.e. no databases are opened) the directories can be empty strings.	
Purpose:	The application program initializes the Thermo-Calc package for thermodynamic calculations. This or TQINI must be called before using any other subroutines in the TQ-Interface.	
Arguments		
Name	**Type**	**Value set on call or returned**
database_path	Character*256	Path to the directory holding the data directory, which in turn contains the databases. See theexamples collection for a default value for this parameter.

Fortran	**TQINI3(DATABASE_PATH, TEMP_PATH, NWSG, NWSE, IWSG, IWSE)**	
C-interface	**tq_ini3(TC_STRING database_path, TC_STRING temp_path, TC_INT nwsg, TC_INT nwse, TC_INT* iwsg, TC_INT* iwse);**	
temp_path	Character*256	Path to the directory for temporary and log file output. This directory has to be writable by the user who runs the application. See the examples collection for a default value for this parameter.
NWSG	Integer	Set to size of the workspace IWSG.
NWSE	Integer	Set to size of the workspace IWSE.
IWSG	Integer array	Memory area for storage of data inside the package.
IWSE	Integer array	Memory area for storage of data inside the package.

TQINI

Fortran	**TQINI(NWSG, NWSE, IWSG, IWSE)**	
C-interface	**tq_ini(TC_INT nwsg, TC_INT nwse,TC_INT* iwsg,TC_INT* iwse);**	
Full name:	Initialize TQ Interface.	
Purpose:	With this subroutine the application program initializes the Thermo-Calc package for thermodynamic calculations. This or TQINI3 must be called before using any other subroutines in the TQ-Interface.	
Arguments		
Name	**Type**	**Value set on call or returned**
NWSG	Integer	Set to size of the workspace IWSG.
NWSE	Integer	Set to size of the workspace IWSE.
IWSG	Integer array	Memory area for storage of data inside the package.

Fortran	**TQINI(NWSG, NWSE, IWSG, IWSE)**	
C-interface	**tq_ini(TC_INT nwsg, TC_INT nwse,TC_INT* iwsg,TC_INT* iwse);**	
IWSE	Integer array	Memory area for storage of data inside the package.

TQSIO

Fortran	**TQSIO(OPTION, IVAL)**	
C-interface	**tq_sio(TC_STRING option,TC_INT ival);**	
Full name:	Set Input/Output Option.	
Purpose:	With this subroutine the application program can re-direct input and output from the Thermo-Calc package.	
Comments:	OPTION is a character identifying the Input/Output option. The current internal value is set to the value in IVAL. If the value is illegal the error condition is set.	
Arguments		
Name	**Type**	**Value set on call or returned**
OPTION	Character*8	Set to a value given in *Legal Input/Output Options for TQSIO and TQGIO* on page 15
IVAL	Integer	Set to an internal value.

TQGIO

Fortran	**TQGIO(OPTION, IVAL)**
C-interface	**tq_gio(TC_STRING option,TC_INT* ival);**
Full name:	Get Input/output Unit.
Purpose:	Obtain a value of Input/Output option.
Comments:	OPTION is a character identifying the Input/Output option. IVAL is an integer where its current internal value is returned.

Fortran	**TQGIO(OPTION, IVAL)**	
C-interface	**tq_gio(TC_STRING option,TC_INT* ival);**	
Arguments		
Name	**Type**	**Value set on call or returned**
OPTION	Character*8	Set to a value given in *Legal Input/Output Options for TQSIO and TQGIO* on page 15.
IVAL	Integer	Return the current internal value.

TQRFIL

Fortran	**TQRFIL(FILE, IWSG, IWSE)**
C-interface	**tq_rfil(TC_STRING file,TC_INT* iwsg,TC_INT* iwse);**
Full name:	Read File.
Purpose:	Read a thermodynamic data file in the Thermo-Calc format.
Comments:	The default set of components is supplied by the thermodynamic input file. The thermodynamic data file should contain at least the following information. • System: name of the elements, molecular mass for elements, list of phases • Phase: list of constituents, type of solution model (if not fixed composition), thermodynamic model parameters • Constituents: name, chemical formula (stoichiometric matrix), molecular mass, thermodynamic properties All this data are not necessarily stored separately, for example the molecular weight for a constituent can be calculated from the masses of the elements.

Fortran	**TQRFIL(FILE, IWSG, IWSE)**	
C-interface	**tq_rfil(TC_STRING file,TC_INT* iwsg,TC_INT* iwse);**	
	The TQ-Interface is not intended to read from a database or a database file and thus selections of data from a database must be made in Thermo-Calc and then stored in a GES file by using the save command in the Gibbs-Energy-System module inside Thermo-Calc. When the GES file is read into the workspace by this subroutine it is possible to manipulate data by changing components and status for components or phases.	
Arguments		
Name	**Type**	**Value set on call or returned**
FILE	Character*60	Legal file name
IWSG	Integer array	Workspace
IWSE	Integer array	Workspace

TQSSU

Fortran	**TQSSU(QUANT, UNIT, IWSG, IWSE)**	
C-interface	**tq_ssu(TC_STRING quant,TC_STRING unit,TC_INT* iwsg,TC_INT* iwse);**	
Full name:	Set System Unit.	
Purpose:	Set the unit for a quantity (like mass, volume, etc.).	
Comments:	Default units are SI unless changes are made by this subroutine. The legal quantities and units are listed in *Units for TQSSU and TQGSU* on page 14.	
Arguments		
Name	**Type**	**Value set on call or returned**
QUANT	Character*60	Set to a legal quantity
UNIT	Character*60	Set to a legal unit

Fortran	**TQSSU(QUANT, UNIT, IWSG, IWSE)**	
C-interface	**tq_ssu(TC_STRING quant,TC_STRING unit,TC_INT* iwsg,TC_INT* iwse);**	
IWSG	Integer array	Workspace
IWSE	Integer array	Workspace

TQGSU

Fortran	**TQGSU(QUANT, UNIT, IWSG, IWSE)**	
C-interface	**tq_gsu(TC_STRING quant,TC_STRING unit,TC_STRING_LENGTH strlen_unit,TC_INT* iwsg,TC_INT* iwse);**	
Full name:	Get System Unit.	
Purpose:	To find what units the TQ-Interface is currently using for a system quantity.	
Comments:	The legal quantities and units are listed in *Units for TQSSU and TQGSU* on page 14.	
Arguments		
Name	**Type**	**Value set on call or returned**
QUANT	Character*60	Set to a legal quantit.
UNIT	Character*60	Return the current unit.
IWSG	Integer array	Workspace
IWSE	Integer array	Workspace

TQSAME

Fortran	**TQSAME(ICODE, IWSG, IWSE)**	
C-interface	**tq_same(TC_INT* icode,TC_INT* iwsg,TC_INT* iwse);**	
Full name:	Same System.	
Purpose:	The application program can check if the thermochemical system has been changed, i.e., not just the conditions but the components or the phases. This is useful if several independent systems operate on the same equilibrium description.	
Comments:	ICODE is an integer with positive value identifying current system. If ICODE is not the same next time TQSAME is called, the system has been changed.This routine may have to be used if the set of components or the set of phases has been changed. The value of ICODE is changed if there are changes of the components, phases, etc., but not with changes in the conditions, or values of thermodynamic model parameters etc.	
Arguments		
Name	**Type**	**Value set on call or returned**
ICODE	Integer	Returns an internal code
IWSG	Integer array	Workspace
IWSE	Integer array	Workspace

TQGVER

Fortran	**TQGVER(VERS, LNKDAT, OSNAME, BUILD, CMPLER)**
C-interface	**void tq_gver(TC_STRING version,TC_STRING_LENGTH strlen_ version,TC_STRING lnkdat,TC_STRING_LENGTH strlen_lnkdat,TC_ STRING osname,TC_STRING_LENGTH strlen_osname,TC_STRING build,TC_STRING_LENGTH strlen_build,TC_STRING cmpler,TC_ STRING_LENGTH strlen_cmpler);**
Full name:	Get version.
Purpose:	The application program gets information about the TQ-library.

Fortran	**TQGVER(VERS, LNKDAT, OSNAME, BUILD, CMPLER)**	
C-interface	**void tq_gver(TC_STRING version,TC_STRING_LENGTH strlen_ version,TC_STRING lnkdat,TC_STRING_LENGTH strlen_lnkdat,TC_ STRING osname,TC_STRING_LENGTH strlen_osname,TC_STRING build,TC_STRING_LENGTH strlen_build,TC_STRING cmpler,TC_ STRING_LENGTH strlen_cmpler);**	
Arguments		
Name	**Type**	**Value set on call or returned**
VERS	Character*32	Returns the version of TQ-library.
LNKDAT	Character*32	Returns the date and time the TQ-library was built.
OSNAME	Character*32	Returns the name of operating system the TQ-library was built for.
BUILD	Character*32	Returns the software revision version of the TQ-library.
CMPLER	Character*72	Returns the name and version of the compiler with which the TQ-library was built.

System Data Manipulation Subroutines

Purpose	Subroutine
Identify components, phases and constituents	
Get number of system components	*TQGNC* on the next page
Set system components	*TQSCOM* on page 25
Get system components	*TQGCOM* on page 26
Get system component index	*TQGSCI* on page 27
Get number of phases	*TQGNP* on page 27
Get phase name	*TQGPN* on page 28
Get phase index	*TQGPI* on page 29
Get phase constituent name	*TQGPCN* on page 29
Get phase constituent index	*TQGPCI* on page 30
Get component chemical formula	*TQGCCF* on page 31
Get phase constituent stoichiometry	*TQGPCS* on page 32
Get number of phase constituents	*TQGNPC* on page 32
Change the status of components, phases, and component reference states	
Change status of system component	*TQCSSC* on page 33
Get status of system component	*TQGSSC* on page 34*
Change status of phase	*TQCSP* on page 35
Get status of phase	*TQGSP* on page 36*
Set reference state	*TQSETR* on page 36
Add a composition set to a phase	*TQPACS* on page 37
	* Logical function
Contributions to the Gibbs energy of a phase	
Set Gibbs energy addition	*TQSGA* on page 38
Get Gibbs energy addition	*TQGGA* on page 38

Legal Component Status

Status	Meaning
ENTERED	The component is included in the system for an equilibrium calculation.
SUSPENDED	The component is excluded from the system and, as a result, some phases may become suspended if their constituents contain this component.

Legal Phase Status

Status	Meaning
ENTERED	The phase is included in an equilibrium calculation. It may be stable or unstable.
DORMANT	The phase is included in an equilibrium calculation but not allowed to become stable. The phase should be stable if the calculation shows that its driving force is positive (or activity is larger than unity)
FIXED	The phase is included in an equilibrium calculation and it must be stable.
SUSPENDED	The phase is ignored in an equilibrium calculation.

TQGNC

Fortran	**TQGNC (NCOM, IWSG, IWSE)**	
C-interface	**tq_gnc(TC_INT* ncom, TC_INT* iwsg, TC_INT* iwse);**	
Full name:	Get Number of components.	
Purpose:	With this subroutine the application program can get numbers of components.	
Arguments		
Name	**Type**	**Value set on call or returned**
NCOM	Integer	Return the number of components.
IWSG	Integer array	Workspace
IWSE	Integer array	Workspace

TQSCOM

Fortran	TQSCOM(NCOM, NAMES, STOI, IWSG, IWSE)	
C-interface	tq_scom(TC_INT num,tc_components_strings* components,TC_FLOAT* stoi,TC_INT* iwsg,TC_INT* iwse);	
Full name:	Set System Component.	
Purpose:	A new set of system components can be defined. The new number of components must be the same as previously. The number of system components can be changed by suspending a component by *TQCSSC* on page 33.	

Comments:	• The set of components must be linearly independent.
	• The names of the new system components are given in NAMES.
	• STOI is a matrix with dimension STOI (1:NCOM,1:NCOM) which gives the stoichiometry of the new components expressed in the old ones.
	• The default set for components is taken from in the input thermodynamic data file.
	• Legal values for the array elements in NAMES are constituent names.
	• The components are numbered as 1 NCOM in the order they are supplied in this call. The conversion from component name to index is also done by *TQGSCI* on page 27.
	Example
	For example, to transform from the components A, B, C to `A2B`, `B4CC2` use the following values of STOI:
	`A2B2.0 1.0 0.0`
	`B4C 0.0 4.0 1.0`
	`C2 0.0 0.0 2.0`	

Arguments		
Name	**Type**	**Value set on call or returned**
NCOM	Integer	Set to the number of components.
NAMES	Character*24 array	Set to component names.
STOI	Double precision matrix	Stoichiometry matrix in old components.

Fortran	**TQSCOM(NCOM, NAMES, STOI, IWSG, IWSE)**	
C-interface	**tq_scom(TC_INT num,tc_components_strings* components,TC_ FLOAT* stoi,TC_INT* iwsg,TC_INT* iwse);**	
IWSG	Integer array	Workspace
IWSE	Integer array	Workspace

TQGCOM

Fortran	**TQGCOM(NCOM, NAMES, IWSG, IWSE)**	
C-interface	**tq_gcom(TC_INT* num,tc_components_strings* components,TC_INT* iwsg,TC_INT* iwse);**	
Full name:	Get System Component.	
Purpose:	Get components of a system	
Comments:	The number of components are returned in NCOM and their names are returned in NAMES. They are returned in an internal sequential order of the TQ-Interface. In other subroutines one must in some cases use the index of a component rather than the name.  Also see *TQGSCI* on the next page.	
Arguments		
Name	**Type**	**Value set on call or returned**
NCOM	Integer	Return the current number of components.
NAMES	Character*24 array	Return the current names of components.
IWSG	Integer array	Workspace
IWSE	Integer array	Workspace

TQGSCI

Fortran	**TQGSCI(INDEXC, NAME, IWSG, IWSE)**	
C-interface	**tq_gsci(TC_INT* index,TC_STRING component,TC_INT* iwsg,TC_INT* iwse);**	
Full name:	Get System Component Index.	
Purpose:	Get index of a system component.	
Comments:	This is a way to translate from a name to an index. In order to translate from a component index to a name, use *TQGCOM* on the previous page. The application program may call TQGCOM only once and maintain itself a list of component names stored by indices.	
Arguments		
Name	**Type**	**Value set on call or returned**
INDEXC	Integer	Return the index of the component.
NAMES	Character*24	Set to a component name.
IWSG	Integer array	Workspace
IWSE	Integer array	Workspace

TQGNP

Fortran	**TQGNP (NPH, IWSG, IWSE)**	
C-interface	**tq_gnp(TC_INT* nph,TC_INT* iwsg,TC_INT* iwse);**	
Full name:	Get Number of Phases.	
Purpose:	The application gets the numbers of phases.	
Comments:	The phases may have any status. They are numbered sequentially from 1 to NPH. Phases with miscibility gap and thus having more than one composition set are counted separately.	
Arguments		
Name	**Type**	**Value set on call or returned**

Fortran	**TQGNP (NPH, IWSG, IWSE)**	
C-interface	**tq_gnp(TC_INT* nph,TC_INT* iwsg,TC_INT* iwse);**	
NPH	Integer	Return the number of phases.
IWSG	Integer array	Workspace
IWSE	Integer array	Workspace

TQGPN

Fortran	**TQGPN (INDEXP, NAME, IWSG, IWSE)**	
C-interface	**tq_gpn(TC_INT index,TC_STRING phase,TC_STRING_LENGTH strlen_phase,TC_INT* iwsg,TC_INT* iwse);**	
Full name:	Get Phase Name.	
Purpose:	With this subroutine theapplication program can convert a phase index to the name of the phase.	
Comments:	The conversion from phase name to phase index is done by *TQGPI* on the next page. Note that phases with miscibility gaps must appear with each possible composition set as a separate phase. These are named as BCC#1, BCC#2 etc.	
Arguments		
Name	**Type**	**Value set on call or returned**
INDEXP	Integer	Set to the index of a phase.
NAME	Character*24	Return the name of the phase.
IWSG	Integer array	Workspace
IWSE	Integer array	Workspace

TQGPI

Fortran	**TQGPI (INDEXP, NAME, IWSG, IWSE)**	
C-interface	**tq_gpi(TC_INT* index,TC_STRING phase,TC_INT* iwsg,TC_INT* iwse);**	
Full name:	Get Phase Index.	
Purpose:	With this subroutine the application program can get the index of a named phase.	
Comments:	The conversion from phase index to phase name is done by *TQGPN* on the previous page.	
Arguments		
Name	**Type**	**Value set on call or returned**
INDEXP	Integer	Return the index of a phase.
NAME	Character*24	Set to a phase name.
IWSG	Integer array	Workspace
IWSE	Integer array	Workspace

TQGPCN

Fortran	**TQGPCN(INDEXP, INDEXC, NAME, IWSG, IWSE)**
C-interface	**tq_gpcn(TC_INT indexp,TC_INT indexc,TC_STRING name,TC_STRING_LENGTH strlen_name,TC_INT* iwsg,TC_INT* iwse);**
Full name:	Get Phase Constituent Name.
Purpose:	With this subroutine the application program can get the name of an indexed constituent.
Comments:	If the same species appear in more than one sublattice site of a phase, they are named as A#2, A#3, etc., which means A on the second sublattice and A on the third sublattice, etc. The opposite conversion is done by *TQGPCI* on the next page.

Fortran	**TQGPCN(INDEXP, INDEXC, NAME, IWSG, IWSE)**	
C-interface	**tq_gpcn(TC_INT indexp,TC_INT indexc,TC_STRING name,TC_STRING_LENGTH strlen_name,TC_INT* iwsg,TC_INT* iwse);**	
Arguments		
Name	**Type**	**Value set on call or returned**
INDEXP	Integer	Set to a phase index.
INDEXC	Integer	Set to the constituent index.
NAME	Character*24	Return the constituent name.
IWSG	Integer array	Workspace
IWSE	Integer array	Workspace

TQGPCI

Fortran	**TQGPCI(INDEXP, INDEXC, NAME, IWSG, IWSE)**	
C-interface	**tq_gpci(TC_INT indexp,TC_INT* indexc,TC_STRING name,TC_INT* iwsg,TC_INT* iwse);**	
Full name:	Get Phase Constituent Index.	
Purpose:	With this subroutine the application program can get the index of a constituent if its name is known.	
Comments:	The opposite conversion is done by *TQGPCN* on the previous page.	
Arguments		
Name	**Type**	**Value set on call or returned**
INDEXP	Integer	Set to a phase index.
INDEXC	Integer	Return the constituent index.
NAME	Character*24	Set to the constituent name.
IWSG	Integer array	Workspace

Fortran	**TQGPCI(INDEXP, INDEXC, NAME, IWSG, IWSE)**	
C-interface	**tq_gpci(TC_INT indexp,TC_INT* indexc,TC_STRING name,TC_INT* iwsg,TC_INT* iwse);**	
IWSE	Integer array	Workspace

TQGCCF

Fortran	**TQGCCF(INDEXC, NEL, ELNAM, STOI, MMASS, IWSG, IWSE)**	
C-interface	**tq_gccf(TC_INT indexc,TC_INT* nel,tc_elements_strings* elname,TC_FLOAT* stoi,TC_FLOAT* mmass,TC_INT* iwsg, TC_INT* iwse);**	
Full name:	Get Component Chemical Formula.	
Purpose:	Get the stoichiometry array for a system component in terms of element.	
Comments:	Obtain the real elements in a component. All other subroutines just deal with a name that does not have to be related to the actual chemical formula. This is also the only subroutine that can provide the symbols of the actual elements in the system. The dimension of ELNAM and STOI is NEL (number of elements in the component).	
Arguments		
Name	**Type**	**Value set on call or returned**
INDEXC	Integer	Set to system a component index.
NEL	Integer	Number of elements in chemical formula.
ELNAM	Character*2 array	Element symbols.
STOI	Double precision array	Stoichiometry array.
MMASS	Double precision	Total mass.
IWSG	Integer array	Workspace
IWSE	Integer array	Workspace

TQGPCS

Fortran	**TQGPCS (INDEXP, INDEXC, STOI, MMASS, IWSG, IWSE)**	
C-interface	**tq_gpcs(TC_INT indexp,TC_INT indexc,TC_FLOAT* stoi,TC_FLOAT* mmass,TC_INT* iwsg,TC_INT* iwse);**	
Full name:	Get Phase Constituent Stoichiometry.	
Purpose:	With this subroutine the application program can obtain the stoichiometry of a constituent expressed in the system components and also the molecular mass.	
Comments:	This does not give the chemical formula in terms of elements for the constituent. The dimension of STOI is NCOM (number of components) get by calling *TQGCOM* on page 26.	
Arguments		
Name	**Type**	**Value set on call or returned**
INDEXP	Integer	Set to a phase index.
INDEXC	Integer	Set to the constituent index.
STOI	Double precision array	Return the stoichiometry array.
MMASS	Double precision	Return the mass.
IWSG	Integer array	Workspace
IWSE	Integer array	Workspace

TQGNPC

Fortran	**TQGNPC(INDEXP, NPCON, IWSG, IWSE)**
C-interface	**tq_gnpc(TC_INT indexp,TC_INT* npcon,TC_INT* iwsg,TC_INT* iwse);**
Full name:	Get Number of Phase Constituent.
Purpose:	With this subroutine the number of constituents in a phase can be obtained.
Comments:	To have also the names, fractions etc. of the constituents, use *TQGPD* on page 60.

Fortran	**TQGNPC(INDEXP, NPCON, IWSG, IWSE)**	
C-interface	**tq_gnpc(TC_INT indexp,TC_INT* npcon,TC_INT* iwsg,TC_INT* iwse);**	
Arguments		
Name	**Type**	**Value set on call or returned**
INDEXP	Integer	Set to a phase index.
NPCON	Integer	Return the number of the phase constituents.
IWSG	Integer array	Workspace
IWSE	Integer array	Workspace

TQCSSC

Fortran	**TQCSSC (INDEXC, STATUS, IWSG, IWSE)**	
C-interface	**tq_cssc(TC_INT index,TC_STRING status,TC_INT* iwsg,TC_INT* iwse);**	
Full name:	Change Status of System Component	
Purpose:	With this subroutine the application program can change status for a system component.	
Comments:	The legal values for STATUS are ENTERED and SUSPENDED.  Also see *Legal Component Status* on page 24. By suspending a system component some phases may also become suspended if they contain this component. For example, in the system Fe-O-S if O is suspended all phases that must dissolve oxygen is automatically suspended. The fraction of oxygen is set to zero in phases that can dissolve oxygen but can also exist without oxygen.	
Arguments		
Name	**Type**	**Value set on call or returned**
INDEXC	Integer	Set to a component index.

Fortran	**TQCSSC (INDEXC, STATUS, IWSG, IWSE)**	
C-interface	**tq_cssc(TC_INT index,TC_STRING status,TC_INT* iwsg,TC_INT* iwse);**	
STATUS	Character*12	Set to the new status
IWSG	Integer array	Workspace
IWSE	Integer array	Workspace

TQGSSC

This is a logical function.

Fortran	**STATUS=TQGSSC (INDEXC, IWSG, IWSE)**	
C-interface	**status=tq_gssc(TC_INT index,TC_INT* iwsg,TC_INT* iwse);**	
Full name:	Get Status of System Component.	
Purpose:	This function returns TRUE if the system component is ENTERED or FALSE if it is SUSPENDED.	
Comments:	The legal values for STATUS are given in *Legal Component Status* on page 24.If the C-interface is used the value returned is of type: TC_BOOL.	
Arguments		
Name	**Type**	**Value set on call or returned**
INDEXC	Integer	Set to a component index.
IWSG	Integer array	Workspace
IWSE	Integer array	Workspace

TQCSP

Fortran	**TQCSP (INDEXP, STATUS, VAL, IWSG, IWSE)**	
C-interface	**tq_csp(TC_INT index,TC_STRING status,TC_FLOAT amount,TC_INT* iwsg,TC_INT* iwse);**	
Full name:	Change Status of Phase.	
Purpose:	Change status for a phase.	
Comments:	The legal values for STATUS are given in *Legal Phase Status* on page 24. For ENTERED phase, VAL is provided as a start value. It is normally set to zero if the phase is not likely to be stable and one if expected to be stable. Setting a phase SUSPENDED or DORMANT is a way to calculate a metastable equilibrium if the phase would be stable. With the DORMANT status one can know if it would be stable or not. For these two statuses, VAL is irrelevant and may be simply put to zero. For FIXED phase the exact amount of the phase must be given. Note that the amount is in number of moles of atoms, which means that the vacancy in a sublattice phase is not included. Therefore, for a vacancy-containing sublattice phase with FIXED status, it is impossible to set VAL to the total number of moles in the system. Setting a phase FIXED decreases the degrees of freedom in the system by 1. To restore the lost degree of freedom the phase should be reset ENTERED. Set a FIXED phase to zero amount is the best way to get the phase stability limits like liquidus or solidus.	
Arguments		
Name	**Type**	**Value set on call or returned**
INDEXP	Integer	Set to a phase index.
STATUS	Character*12	Set to the status code
VAL	Double precision	Set to phase amount in number of moles.
IWSG	Integer array	Workspace
IWSE	Integer array	Workspace

TQGSP

This is a logical function.

Fortran	**STATUS=TQGSP (INDEXP, STATUS, VAL, IWSG, IWSE)**	
C-interface	**status=tq_gsp(TC_INT index,TC_STRING status,TC_STRING_LENGTH strlen_status,TC_FLOAT* amount,TC_INT* iwsg,TC_INT* iwse);**	
Full name:	Get Status of Phase.	
Purpose:	This function is TRUE if the phase is ENTERED or FIXED. If the phase is SUSPENDED or DORMANT it is FALSE. The status is also returned in STATUS. The application program can test the status of a phase by calling this function.	
Comments:	The legal values for STATUS are listed in *Legal Phase Status* on page 24. If the C-interface is used the value returned is of type: TC_BOOL.	
Arguments		
Name	**Type**	**Value set on call or returned**
INDEXP	Integer	Set to a phase index.
STATUS	Character*12	Return the current status code.
VAL	Double precision	Return the phase amount.
IWSG	Integer array	Workspace
IWSE	Integer array	Workspace

TQSETR

Fortran	**TQSETR (INDEXC, INDEXP, TEMP, PRES, IWSG, IWSE)**
C-interface	**tq_setr(TC_INT indexc,TC_INT indexp,TC_FLOAT temp,TC_FLOAT press,TC_INT* iwsg, TC_INT* iwse);**
Full name:	Set Reference State.
Purpose:	Reset the reference state of a system component.
Comments:	By default the reference state for a component is determined by the

Fortran	**TQSETR (INDEXC, INDEXP, TEMP, PRES, IWSG, IWSE)**	
C-interface	**tq_setr(TC_INT indexc,TC_INT indexp,TC_FLOAT temp,TC_FLOAT press,TC_INT* iwsg, TC_INT* iwse);**	
	thermodynamic data file. With this subroutine an application may select a different reference state if the one in the data file does not suit a calculation purpose. If the current temperature or pressure should be used for the calculation, the value given should not be larger than zero.	
Arguments		
Name	**Type**	**Value set on call or returned**
INDEXC	Integer	Set to a component index.
INDEXP	Integer	Set to a phase index.
TEMP	Double precision	Set to a temperature value.
PRES	Double precision	Set to a pressure value.
IWSG	Integer array	Workspace
IWSE	Integer array	Workspace

TQPACS

Fortran	**TQPACS (INDEXP, IWSG, IWSE)**	
C-interface	**tq_pacs(TC_INT indexp,TC_INT* iwsg,TC_INT* iwse);**	
Full name:	Add composition set to phase.	
Purpose:	Add another composition set to a phase.	
Arguments		
Name	**Type**	**Value set on call or returned**
INDEXP	Integer	Set to a phase index.
IWSG	Integer array	Workspace
IWSE	Integer array	Workspace

TQSGA

Fortran	**TQSGA (INDEXP, VALUE, IWSG, IWSE)**	
C-interface	**tq_gga(TC_INT indexp,TC_FLOAT value,TC_INT* iwsg,TC_INT* iwse);**	
Full name:	Set Gibbs energy addition.	
Purpose:	Add an amount of extra contribution to the Gibbs energy of a phase	
Comments:	The extra contribution may be due to elastic strain energy, surface energy, etc.	
Arguments		
Name	**Type**	**Value set on call or returned**
INDEXP	Integer	Set to a phase index.
VALUE	Double precision	Set to the value of extra contribution.
IWSG	Integer array	Workspace
IWSE	Integer array	Workspace

TQGGA

Fortran	**TQGGA (INDEXP, VALUE, IWSG, IWSE)**	
C-interface	**tq_gga(TC_INT indexp,TC_FLOAT* value,TC_INT* iwsg,TC_INT* iwse);**	
Full name:	Get Gibbs energy addition.	
Purpose:	The contribution added to the Gibbs energy of a phase can be retrieved.	
Arguments		
Name	**Type**	**Value set on call or returned**
INDEXP	Integer	Set to a phase index.
VALUE	Double precision	Return the value of extra contribution.
IWSG	Integer array	Workspace
IWSE	Integer array	Workspace

Condition, Stream and Segment Subroutines

A *stream* is considered as a non-reactive medium for transferring matter to a reaction zone. It has constant temperature and pressure, and contains one or more phases of a certain composition, i.e., for each stream, temperature, pressure, and input amounts of phase constituents must be defined. Different sets of equilibrium *conditions* can be defined for the same system in different *segments*.

Purpose	Subroutine
Set condition	*TQSETC* on page 42
Remove condition	*TQREMC* on page 43
Save current conditions	*TQSCURC* on page 44
Remove all conditions	*TQREMAC* on page 45
Restore saved conditions	*TQRESTC* on page 45
Create stream	*TQCSTM* on page 46
Set stream constituent amount	*TQSSC* on page 46
Set stream invariant state variable	*TQSSIC* on page 47
Delete stream	*TQDSTM* on page 49
Create new equilibrium segment	*TQNSEG* on page 49
Select equilibrium segment	*TQSSEG* on page 50

Possible State Variables to Set Conditions in TQSETC

STAVAR	INDEXP	INDEXC	Meaning	Comments
T			Temperature	of the whole system
P			Pressure	of the whole system
MU	note 1	Yes	Chemical potential	of a system component
MUC	Yes	Yes	Chemical potential	of a phase constituent
AC	note 1	Yes	Activity	of a system component
ACC	Yes	Yes	Activity	of a system constituent
V			Volume	of the whole system
G			Gibbs energy	of the whole system

STAVAR	INDEXP	INDEXC	Meaning	Comments
H			Enthalpy	of the whole system
S			Entropy	of the whole system
N			Moles	of all system components
N		Yes	Moles	of a system component
NP	note 2		Moles	of a phase
M			Total mass	of all system components
M		Yes	Mass	of a system component
BP	note 2		Mass	of a phase
IN	Yes	Yes	Input amount	in moles of phase constituents
IM	Yes	Yes	Input amount	in mass units of phase constituents
X		Yes	Mole fraction	of a system component
W		Yes	Mass (Weight) fraction	of a system component
X%		Yes	Mole percent	of a system component
W%		Yes	Mass (Weight) fraction	of a system component

Note 1: Giving a phase index means to define the reference state. If no phase index is given the previous reference state is used. The default reference state is SER (Standard Element Reference) if the thermodynamic data file is created from a SGTE (Scientific Group Thermodata Europe) database. It is necessary that the phase can exist with the constituent as its single constituent. It is an error to set FCC as reference state for carbon if carbon dissolves interstitially in FCC.

Note 2: Not recommended to be used for setting conditions. To calculate stability limit one should use TQCSP with FIXED status and amount of the phase set to zero.

You can add a normalizing suffix like M (per mole), W (per mass) or V (per volume) on G, H, S, etc.

TQSETC

Fortran	**TQSETC(STAVAR, INDEXP, INDEXC, VAL, NUMCON, IWSG, IWSE)**	
C-interface	**tq_setc(TC_STRING condition,TC_INT indexp,TC_INT indexc,TC_FLOAT val,TC_INT* numcon,TC_INT* iwsg,TC_INT* iwse);**	
Full name:	Set Condition.	
Purpose:	To set conditions for an equilibrium calculation.	
Comments:	In STAVAR the mnemonic of the state variable must be given, see *Possible State Variables to Set Conditions in TQSETC* on page 40. In some cases just the mnemonic is needed, like for temperature or pressure, but in many cases a phase index or a component index must be used to specify the condition. If both a phase index and a constituent index is supplied the condition is set for the specified constituent in the specified phase. The application program must set exactly the same number of conditions as degrees of freedom in the defined system. The degrees of freedom are equal to the number of system components plus two (usually temperature and pressure). Setting a phase FIXED using TQCSP decrease the degrees of freedom in the system by 1. Resetting the phase ENTERED using TQCSP restores one degree of freedom. Possible combinations of STAVAR and indices are listed in *Possible State Variables to Set Conditions in TQSETC* on page 40. Here it is shown that the same value of STAVAR may be used with or without an index. In the case there should not be an index, the value of INDEXP or INDEXC must be negative. Some combination of conditions may be thermodynamically impossible. The TQ-Interface provides relevant help for such cases.	
Arguments		
Name	**Type**	**Value set on call or returned**
STAVAR	Character*8	Set as a state variable
INDEXP	Integer	Set as a phase index (if needed).
INDEXC	Integer	Set as a component or constituent index (if needed).
VAL	Double precision	Set to the value.

Fortran	**TQSETC(STAVAR, INDEXP, INDEXC, VAL, NUMCON, IWSG, IWSE)**	
C-interface	**tq_setc(TC_STRING condition,TC_INT indexp,TC_INT indexc,TC_FLOAT val,TC_INT* numcon,TC_INT* iwsg,TC_INT* iwse);**	
NUMCON	Integer	Returned as an identification of the condition.
IWSG	Integer array	Workspace
IWSE	Integer array	Workspace

Examples

Set the temperature to 800 Celsius

```
CALL TQSSU('Temperature','C',IWSG,IWSE)
CALL TQSETC('T',-1,-1,800.0D0,NCOND,IWSG,IWSE)
Set the incoming amount of a liquid phase constituent named Al2O3 to 1.5 moles
CALL TQGPI(INDEXP,'LIQUID',IWSG,IWSE)
CALL TQGPCI(INDEXP,INDEXC,'AL2O3',IWSG,IWSE)
CALL TQSETC('IN',INDEXP,INDEXC,1.5D0,NCOND,IWSG,IWSE)
```

Set the mass percent of the system component Cr to 13%.

```
CALL TQGSCI(INDEX,'cr',IWSG,IWSE)
CALL TQSETC('W%',-1,INDEX,13.0D0,NCOND,IWSG,IWSE)
```

Set the total amount of system to 1.0 mole components

```
CALL TQSETC('N',-1,-1,1.0D0,NCOND,IWSG,IWSE)
```

Set the mole fraction of H2O in GAS to 5 mol percent

```
CALL TQGPI(INDEXP,'GAS',IWSG,IWSE)
CALL TQGPCI(INDEXP,INDEXC,'H2O1',IWSG,IWSE)
CALL TQSETC('X',INDEXP,INDEXC,0.05D0,NCOND,IWSG,IWSE)
```

TQREMC

Fortran	**TQREMC(NUMCON, IWSG, IWSE)**
C-interface	**tq_remc(TC_INT numcon,TC_INT* iwsg,TC_INT* iwse);**
Full name:	Remove Condition.
Purpose:	Remove the condition numbered NUMCON.

Fortran	**TQREMC(NUMCON, IWSG, IWSE)**	
C-interface	**tq_remc(TC_INT numcon,TC_INT* iwsg,TC_INT* iwse);**	
Comments:	*TQSETC* on page 42 and *TQCSTM* on page 46 return an index for each condition set. This value must be supplied in this call. In order to change a condition to something else, not just a new value, one must first remove the condition. If one just wants to change the value of a condition one may call TQSETC again instead.	
Arguments		
Name	**Type**	**Value set on call or returned**
NUMCON	Integer	Set to a condition number.
IWSG	Integer array	Workspace
IWSE	Integer array	Workspace

TQSCURC

Fortran	**TQSCURC(IWSG, IWSE)**	
C-interface	**tq_scurc(TC_INT* iwsg,TC_INT* iwse);**	
Full name:	Save Current Conditions.	
Purpose:	Save all conditions in case they need to be restored.	
Comments:	The saved conditions can be restored if necessary by using *TQRESTC* on the next page.	
Arguments		
Name	**Type**	**Value set on call or returned**
IWSG	Integer array	Workspace
IWSE	Integer array	Workspace

TQREMAC

Fortran	**TQREMAC(IWSG, IWSE)**	
C-interface	**tq_remac(TC_INT* iwsg,TC_INT* iwse);**	
Full name:	Remove All Conditions.	
Purpose:	TQREMAC provides the easiest way to remove all conditions. After calling TQREMAC, one can set completely new or restore previously saved conditions.	
Arguments		
Name	**Type**	**Value set on call or returned**
IWSG	Integer array	Workspace
IWSE	Integer array	Workspace

TQRESTC

Fortran	**TQRESTC(IWSG, IWSE)**	
C-interface	**tq_restc(TC_INT* iwsg,TC_INT* iwse);**	
Full name:	Restore Condition.	
Purpose:	Restore saved conditions.	
Comments:	Before calling TQRESTC, remove all present conditions.	
Arguments		
Name	**Type**	**Value set on call or returned**
IWSG	Integer array	Workspace
IWSE	Integer array	Workspace

TQCSTM

Fortran	**TQCSTM(IDENT, TEMP, PRESS, IWSG, IWSE)**	
C-interface	**tq_cstm(TC_STRING stream,TC_FLOAT temp,TC_FLOAT press,TC_INT* iwsg,TC_INT* iwse);**	
Full name:	Create Stream	
Purpose:	To set the system conditions by stream input. Stream calculations are useful when calculating differences between an initial state and a final state. The streams define the initial state of the system components by specifying reactants of different phases at given temperatures and pressures.	
Comments:	A stream is a non-reacting media for transferring matter to a reaction zone. A stream may contain several phases at the same given temperature and pressure. Phases with different temperatures and pressures should be grouped into different streams. Several streams can be transferred to a reaction zone. The input constituents of each phase do not react in a stream.	
Arguments		
Name	**Type**	**Value set on call or returned**
IDENT	Character*24	Set as identifier of the stream.
TEMP	Double precision	Input temperature of stream.
PRESS	Double precision	Input pressure of stream.
IWSG	Integer array	Workspace
IWSE	Integer array	Workspace

TQSSC

Fortran	**TQSSC(IDENT, INDEXP, INDEXC, VALUE, NUMIN, IWSG, IWSE)**
C-interface	**tq_ssc(TC_STRING stream,TC_INT iph,TC_INT icmp,TC_FLOAT value,TC_INT icond,TC_INT* iwsg,TC_INT* iwse);**
Full name:	Set Stream Constituent Amount.
Purpose:	Set the amount of phase constituent in a stream.

Fortran	**TQSSC(IDENT, INDEXP, INDEXC, VALUE, NUMIN, IWSG, IWSE)**	
C-interface	**tq_ssc(TC_STRING stream,TC_INT iph,TC_INT icmp,TC_FLOAT value,TC_INT icond,TC_INT* iwsg,TC_INT* iwse);**	
Comments:	The last one takes effect if the amount of the same phase constituent have been set several times, i.e., the amount cannot be set additively.	
Arguments		
Name	**Type**	**Value set on call or returned**
IDENT	Character*24	Set as identifier of the stream.
INDEXP	Integer	Set as a phase index
INDEXC	Integer	Set as a constituent index.
VALUE	Double precision	Set to an amount of the constituent INDEXC in the stream.
NUMIN	Integer	Returned as identification of the input constituent in the stream.
IWSG	Integer array	Workspace
IWSE	Integer array	Workspace

TQSSIC

Fortran	**TQSSIC(STAVAR, VALUE, IWSG, IWSE)**
C-interface	**tq_ssic(TC_STRING stavar,TC_FLOAT value,TC_INT* iwsg,TC_INT* iwse);**
Full name:	Set Stream Invariant State Variable.
Purpose:	To specify the invariant state variable for calculating the reaction of all streams.
Comments:	The state variables that could be used are G, H, S, and V with a suffix D, which means difference between initial and final states of the reaction.
Arguments	

Fortran	**TQSSIC(STAVAR, VALUE, IWSG, IWSE)**	
C-interface	**tq_ssic(TC_STRING stavar,TC_FLOAT value,TC_INT* iwsg,TC_INT* iwse);**	
Name	**Type**	**Value set on call or returned**
STAVAR	Character*8	Set as the mnemonic of a state variable.
VALUE	Double precision	Set to change in value of STAVAR.
IWSG	Integer array	Workspace
IWSE	Integer array	Workspace

Examples

Calculation of adiabatic temperature for knallgas.

```
DIMENSION TPA(2)
C...set input temperature and pressure
TEMP=298.15D0
PRES=1.0D5
C...create the stream
CALL TQCSTM('knallgas',TEMP,PRES,IWSG,IWSE)
C...set amount of H2 and O2 in the stream
CALL TQGPCI(1,INDEXC,'H2',IWSG,IWSE)
CALL TQSSC('knallgas',1,INDEXC,2.0D0,NUMIN,IWSG,IWSE)
CALL TQGPCI(1,INDEXC,'O2',IWSG,IWSE)
CALL TQSSC('knallgas',1,INDEXC,1.0D0,NUMIN,IWSG,IWSE)
C...set the global temperature and pressure for the reaction
CALL TQSETC('T',-1,-1,500.0D+0,NUMC,IWSG,IWSE)
CALL TQSETC('P',-1,-1,PRES,NUMC,IWSG,IWSE)
C...get  the enthalpy of reaction
CALL TQCE(' ',-1,-1,0.0D+0,IWSG,IWSE)
CALL TQGETV1('HD',-1,-1,ENT,IWSG,IWSE)
WRITE(*,*)'Calculated enthalpy of reaction are '
&, ENT, ' at 500 K. '
C...set that the enthalpy shall be constant in the calculation
CALL TQSSIC('HD',0.0D0,IWSG,IWSE)
```

```
C...calculate
CALL TQCE('T',-1,-1,1.0D+0,IWSG,IWSE)
C...get temperature
CALL TQGETV1('T',-1,-1,TEMP,IWSG,IWSE)
WRITE(*,*)'Calculated temperature ',TEMP
```

TQDSTM

Fortran	**TQDSTM(IDENT, IWSG, IWSE)**	
C-interface	**tq_dstm(TC_STRING stream, TC_INT* iwsg,TC_INT* iwse);**	
Full name:	Delete Stream.	
Purpose:	Delete all or one stream.	
Comments:	Use an empty string as IDENT removes all the streams entered.	
Arguments		
Name	**Type**	**Value set on call or returned**
IDENT	Character*24	Set as identifier of the stream.
IWSG	Integer array	Workspace
IWSE	Integer array	Workspace

TQNSEG

Fortran	**TQNSEG(ID, IWSG, IWSE)**
C-interface	**tq_nseg(TC_STRING id, TC_INT* iwsg,TC_INT* iwse);**
Full name:	New Equilibrium Segment.
Purpose:	With this subroutine the application program can create a new equilibrium description with the same thermodynamic data. This subroutine is useful when simulating several equilibria representing local conditions, for example, in the reactor simulator.
Comments:	TQNSEG does not read any thermodynamic file. This must have already been done with *TQRFIL* on page 18. Note that when several segments are used, an

Fortran	**TQNSEG(ID, IWSG, IWSE)**	
C-interface	**tq_nseg(TC_STRING id, TC_INT* iwsg,TC_INT* iwse);**	
	equilibrium should be computed when a segment is selected before any data is retrieved.	
Arguments		
Name	**Type**	**Value set on call or returned**
ID	Character*24	Set as identifier of the equilibrium segment.
IWSG	Integer array	Workspace
IWSE	Integer array	Workspace

TQSSEG

Fortran	**TQSSEG(ID, IWSG, IWSE)**	
C-interface	**tq_sseg(TC_STRING id, TC_INT* iwsg,TC_INT* iwse);**	
Full name:	Select Equilibrium.	
Purpose:	When the application program has created several equilibrium segments using *TQNSEG* on the previous page, this subroutine makes it possible to select a current equilibria which the subroutine calls refer to.	
Comments:	When several segments are used, and before any data is retrieved, an equilibrium should be computed when a segment is selected.	
Arguments		
Name	**Type**	**Value set on call or returned**
ID	Character*24	Set to an equilibrium identification.
IWSG	Integer array	Workspace
IWSE	Integer array	Workspace

Calculations and Results Subroutines

Purpose	Subroutine
Calculate equilibrium	*TQCE* on page 54
Calculate global equilibrium	*TQCEG* on page 56
Get equilibrium property values (TQGET) and Get one value (TQGET1)	*TQGETV and TQGET1* on page 57
Get chemical potential value. It is a double precision function.	*TQGMU* on page 59
Get molar Gibbs energy value. It is a double precision function.	*TQGGM* on page 60
Get phase data	*TQGPD* on page 60
Get driving force and local equilibrium compositions for ortho- or para-equilibrium phase transformation	*TQGDF2* on page 62
Get interfacial energy between a matrix phase and a precipitate phase	*TQGSE* on page 64

State Variables Available for TQGETV and TQGET1

STAVAR	INDEXP	INDEXC	Meaning	Comments
T			Temperature	of the whole system
P			Pressure	of the whole system
MU	(yes)	Yes	Chemical potential	of a system component
MUC	Yes	yes	Chemical potential	of a constituent in a gas phase
AC	(yes)	Yes	Activity	of a system component
ACC	Yes	Yes	Activity	of a constituent in a gas phase
V			Volume	of the whole system
V	Yes		Volume	of a phase
G*			Gibbs energy	of the whole system
G*	Yes		Gibbs energy	of a phase
H*			Enthalpy	of the whole system
H*	Yes		Enthalpy	of a phase

STAVAR	INDEXP	INDEXC	Meaning	Comments
S*			Entropy	of the whole system
S*	Yes		Entropy	of a phase
CP			Heat capacity	of the system
CP	Yes		Heat capacity	of a phase
DG	Yes		Driving force	of a phase
N			Moles	of all system components
N		Yes	Moles	of a system component
NP	Yes		Moles	of a system phase
M			Total mass	of all system components
M		Yes	Mass	of a system component
BP	Yes		Mass	of a system phase
IN	Yes	Yes	Input amount	in moles of phase constituents
IM	Yes	Yes	Input amount	in mass units of phase constituents
X		Yes	Mole fraction	of a component in the whole system
X	Yes	Yes	Mole fraction	of a component in a phase
W		Yes	Mass (Weight) fraction	of a component in the whole system
W	Yes	Yes	Mass (Weight) fraction	of a component in a phase
X%		Yes	Mole percent	of a component in the whole system
X%	Yes	Yes	Mole percent	of a component in a phase
W%		Yes	Mass (Weight) fraction	of a component in the whole system
W%	Yes	Yes	Mass (Weight) fraction	of a component in a phase
Y	Yes	Yes	Constituent fraction	of a phase constituent

STAVAR	INDEXP	INDEXC	Meaning	Comments
* You can add a normalizing suffix like M (per mole), W (per mass) or V (per volume) on G, H, S, etc. R can also be added as a suffix on G, H, S to get a value that is calculated with respect to the reference state specified by calling TQSETR.				

Additional Variables Available for TQGETV and TQGET1

STAVAR	Meaning	Unit
M(phase,J)	Mobility coefficient where J=diffusing species	m^2/s
LOGM(phase,J)	10log of the mobility coefficient	m^2/s
DT(phase,J)	Tracer diffusion coefficient where J=diffusing species	m^2/s
LOGDT(phase,J)	10log of the tracer diffusion coefficient	m^2/s
DC(phase,J,K,N)	Chemical diffusion coefficient where K=gradient specie, and N=reference specie	m^2/s
LOGDC (phase,J,K,N)	10log of the chemical diffusion coefficient	m^2/s
DI(phase,J,K,N)	Intrinsic diffusion coefficient	m^2/s
LOGDI(phase,J,K,N)	10log of the intrinsic diffusion coefficient	m^2/s
QC(phase,J,K,N)	Q=R(*ln*(DC{T1})-*ln*(DC{T1+ε}))/(1/(T1+ε)-1/T1)	J/mol
QT(phase,J)	Q=R(*ln*(DT{T1})-*ln*(DT{T1+ε}))/(1/(T1+ε)-1/T1)	J/mol
QI(phase,J,K,N)	Q=R(*ln*(DI{T1})-*ln*(DI{T1+ε}))/(1/(T1+ε)-1/T1)	J/mol
FC(phase,J,K,N)	D0=exp(ln(DC{T1})+Q/R/T1)	m^2/s
FI(phase,J,K,N)	D0=exp(ln(DI{T1})+Q/R/T1)	m^2/s
FT(phase,J)	D0=exp(ln(DT{T1})+Q/R/T1)	m^2/s

TQCE

Fortran	**TQCE(TARGET, INDEXP, INDEXC, VALUE, IWSG, IWSE)**
C-interface	**tq_ce(TC_STRING var,TC_INT indexp,TC_INT indexc,TC_FLOAT value,TC_INT* iwsg,TC_INT* iwse);**
Full name:	Calculate Equilibrium.
Purpose:	Calculate the equilibrium with current settings of conditions or streams.
Comments:	Some software needs a TARGET specified for certain types of calculations. A TARGET is a state variable as specified in *TQSETC* on page 42. When working with Thermo-Calc, it is only useful in stream reaction calculations, where an initial guess of the target variable may be of some help. Otherwise, TARGET is

Fortran	**TQCE(TARGET, INDEXP, INDEXC, VALUE, IWSG, IWSE)**	
C-interface	**tq_ce(TC_STRING var,TC_INT indexp,TC_INT indexc,TC_FLOAT value,TC_INT* iwsg,TC_INT* iwse);**	
	normally set as an empty string and the values of INDEXP, INDEXC, and VALUE are irrelevant.	
Arguments		
Name	**Type**	**Value set on call or returned**
TARGET	Character*8	Set to a state variable, if necessary.
INDEXP	Integer	Set to a phase index, if necessary.
INDEXC	Integer	Set to a component index, if necessary.
VALUE	Double precision	Set to an estimate of the target variable.
IWSG	Integer array	Workspace
IWSE	Integer array	Workspace

Example

Calculate enthalpy for an equilibrium gas mixture SO3, SO2 and O2. Input SO3 2%, O2 10% and 88% SO2.

```
CALL TQGPI('GAS',INDEXP,IWSG,IWSE)
C...set temperature, pressure and total amount of moles
CALL TQSETC('T',-1,-1,800.0D0,NCOND,IWSG,IWSE)
CALL TQSETC('P',-1,-1,1.0D5,NCOND,IWSG,IWSE)
CALL TQSETC('N',-1,-1,1.0D0,NCOND,IWSG,IWSE)
C...set mole fraction of SO3 and O2
CALL TQGPI(INDEXP,'GAS',IWSG,IWSE)
CALL TQGPCI(INDEXP,INDEXC,'SO2',IWSG,IWSE)
CALL TQSETC('IN',INDEXP,INDEXC,8.8D-1,NCOND,IWSG,IWSE)
CALL TQGPCI(INDEXP,INDEXC,'O2',IWSG,IWSE)
CALL TQSETC('IN',INDEXP,INDEXC,1.0D-1,NCOND,IWSG,IWSE)
CALL TQGPCI(INDEXP,INDEXC,'SO3',IWSG,IWSE)
CALL TQSETC('IN',INDEXP,INDEXC,2.0D-2,NCOND,IWSG,IWSE)
CALL TQCE(' ',0,0,0.0D+0,IWSG,IWSE)
CALL TQGETV1('H',-1,-1,ENT,IWSG,IWSE)
```

In this way an application program can calculate the incoming enthalpy into the system. If there is more than one incoming flow it can calculate the enthalpies for each flow and sum them up.

TQCEG

Fortran	**TQCEG(IWSG, IWSE)**	
C-interface	**tq_ceg(TC_INT* iwsg,TC_INT* iwse);**	
Full name:	Calculate Equilibrium Global.	
Purpose:	Calculate Equilibrium using Global Minimization Algorithm.	
Comments:	The use of global minimization algorithm is meant to avoid metastable or unstable equilibrium and to obtain truly stable equilibrium. This is mainly due to its ability to find automatically miscibility gap and create accordingly new composition sets. As a consequence, the number of phases may increase after calling TQCEG. The newly added phases (new composition sets of old phases) are always put in the end of the phase list. In this way, the indexes of old phases remain the same as before See Example 13. The global minimization technique starts with discretizing the composition space and calculating Gibbs energy values at each grid point for each phase at a given temperature. This usually leads to a significant increase of computation time. Therefore, it is not recommended to use TQCEG in time-critical application programs. In the cases where phases involved are well known, for example, identifying the local equilibrium at a phase interface, it is absolutely not necessary to use TQCEG. If TQCEG is needed, irrelevant phases should better be rejected in the beginning when fetching thermodynamic data from a database.	
Arguments		
Name	**Type**	**Value set on call or returned**
IWSG	Integer array	Workspace
IWSE	Integer array	Workspace

TQGETV and TQGET1

Fortran	**TQGETV(STAVAR, INDEXP, INDEXC, NUMBER, VALAR, IWSG, IWSE)** **TQGET1(STAVAR, INDEXP, INDEXC, VAL, IWSG, IWSE)**
C-interface	**tq_getv(TC_STRING stavar,TC_INT indexp,TC_INT indexc,TC_INT number,TC_FLOAT* valar,TC_INT* iwsg,TC_INT* iwse);** **tq_get1(TC_STRING stavar,TC_INT indexp,TC_INT indexc,TC_FLOAT* val,TC_INT* iwsg,TC_INT* iwse);**
Full name:	Get Values. With TQGETV an array of values can be returned; with TQGET1 a single value only.
Purpose:	These subroutines return the value of any variable in the system after an equilibrium calculation, for example, thermodynamic properties for phases and constituents, temperature, pressure and volume of the system, and amount of the system, a phase or a constituent.
Comments:	If an equilibrium is not established, the error code is set on return. Go to *State Variables Available for TQGETV and TQGET1* on page 51 for obtaining values. Valid INDEXC or INDEXP has a positive value. Setting INDEXC or INDEXP to -1 means it is not relevant. Using 0 for INDEXC or INDEXP in TQGETV means all components or all phases, respectively.

Arguments

Name	Type	Value set on call or returned
STAVAR	Character*32	Set to mnemonic of state variable
INDEXP	Integer	Set to a phase index
INDEXC	Integer	Set to a component or constituent index
NUMBER	Integer	Set to the number of values in VALAR.
VALAR	Double precision array	Return the values
VAL	Double precision	Return the value
IWSG	Integer array	Workspace

Fortran	**TQGETV(STAVAR, INDEXP, INDEXC, NUMBER, VALAR, IWSG, IWSE)** **TQGET1(STAVAR, INDEXP, INDEXC, VAL, IWSG, IWSE)**	
C-interface	**tq_getv(TC_STRING stavar,TC_INT indexp,TC_INT indexc,TC_INT number,TC_FLOAT* valar,TC_INT* iwsg,TC_INT* iwse);** **tq_get1(TC_STRING stavar,TC_INT indexp,TC_INT indexc,TC_FLOAT* val,TC_INT* iwsg,TC_INT* iwse);**	
IWSE	Integer array	Workspace

Examples

Get temperature of the system

```
CALL TQGET1('T',-1,-1,VAL,IWSG,IWSE)
```

Get overall mole fraction of system component Cr

```
CALL TQGSCI(INDEXC,'CR',IWSG,IWSE)
CALL TQGET1('X',-1,INDEXC,VAL,IWSG,IWSE)
```

Get overall mole fractions of all components

```
CALL TQGETV('x',-1,0,NCOM,VALAR,IWSG,IWSE)
```

Get activity of system component SiC

```
CALL TQGSCI(INDEXC,'sic',IWSG,IWSE)
CALL TQGET1('AC',-1,INDEXC,VAL,IWSG,IWSE)
```

Get activity of gas phase constituent SiC (gas is phase 1)

```
CALL TQGPCI(1,INDEXC,'sic',IWSG,IWSE)
CALL TQGET1('AC',1,INDEXC,VAL,IWSG,IWSE)
```

Get total mass of system

```
CALL TQGET1('M',-1,-1,VAL,IWSG,IWSE)
CALL TQGET1('M',0,0,VAL,IWSG,IWSE)
```

Get total mass of liquid phase

```
CALL TQGPI(INDEXP,'LIQUID',IWSG,IWSE)
```

or

```
CALL TQGET1('BP',INDEXP,-1,VAL,IWSG,IWSE)
```

Get mass of all constituents of liquid phase

```
CALL TQGPI(INDEXP,'LIQUID',IWSG,IWSE)
CALL TQGETV('IM',INDEXP,0,NVAL,VALAR,IWSG,IWSE)
```

Get mass of SIC in liquid phase

```
CALL TQGPI(INDEXP,'LIQUID',IWSG,IWSE)
CALL TQGPCI(INDEXP,INDEXC,'sic',IWSG,IWSE)
CALL TQGET1('IM',INDEXP,INDEXC,VAL,IWSG,IWSE)
```

Get volume of GAS phase

```
CALL TQGET1('V',1,-1,VAL,IWSG,IWSE)
```

Get constituent mole fraction of H2O in GAS

```
CALL TQGPCI(1,INDEXC,'h2o',IWSG,IWSE)
CALL TQGET1('y',1,INDEXC,VAL,IWSG,IWSE)
```

Get partial pressure of H2O in GAS (equal to the total pressure times the constituent mole fraction)

```
CALL TQGPCI(1,INDEXC,'h2o',IWSG,IWSE)
CALL TQGET1('y',1,INDEXC,VAL,IWSG,IWSE)
CALL TQGET1('p',-1,-1,PVAL,IWSG,IWSE)
PH2O = PVAL*VAL
```

Get chemical potentials of all constituents in slag

```
CALL TQGPI(INDEXP,'slag',IWSG,IWSE)
CALL TQGETV('MUC',INDEXP,0,NCON,VALAR,IWSG,IWSE)
```

TQGMU

This is a double precision function.

Fortran	**TQGMU (INDEXC, IWSG, IWSE)**	
C-interface	**tq_gmu(TC_INT indexc, TC_INT* iwsg, TC_INT* iwse);**	
Full name:	Get Chemical Potential.	
Purpose:	This function returns the chemical potential of a component in a faster way.	
Arguments		
Name	**Type**	**Value set on call or returned**
INDEXC	Integer	Set to a component index
IWSG	Integer array	Workspace

Fortran	**TQGMU (INDEXC, IWSG, IWSE)**	
C-interface	**tq_gmu(TC_INT indexc, TC_INT* iwsg, TC_INT* iwse);**	
IWSE	Integer array	Workspace

TQGGM

This is a double precision function.

Fortran	**TQGGM (INDEXP, IWSG, IWSE)**	
C-interface	**tq_ggm(TC_INT indexp, TC_INT* iwsg, TC_INT* iwse);**	
Full name:	Get Molar Gibbs Energy.	
Purpose:	This function returns the molar Gibbs energy of a phase more quickly.	
Arguments		
Name	**Type**	**Value set on call or returned**
INDEXP	Integer	Set to a phase index
IWSG	Integer array	Workspace
IWSE	Integer array	Workspace

TQGPD

Fortran	**TQGPD (INDEXP, NSUB, NSCON, SITES, YFRAC, EXTRA, IWSG, IWSE)**
C-interface	**tq_gpd(TC_INT indexp,TC_INT* nsub,TC_INT* nscon,TC_FLOAT* sites,TC_FLOAT* yfrac,TC_FLOAT* extra,TC_INT* iwsg,TC_INT* iwse);**
Full name:	Get Phase Data.
Purpose:	The application program can get data for the constituents of a phase.
Comments:	With this subroutine the application program can determine the structure of the phase and the fraction of the constituents and other things. Note that

Fortran	**TQGPD (INDEXP, NSUB, NSCON, SITES, YFRAC, EXTRA, IWSG, IWSE)**	
C-interface	**tq_gpd(TC_INT indexp,TC_INT* nsub,TC_INT* nscon,TC_FLOAT* sites,TC_FLOAT* yfrac,TC_FLOAT* extra,TC_INT* iwsg,TC_INT* iwse);**	
	YFRAC is constituent fraction, not mole fractions.A substitutional phase has NSUB equal to 1, which is identical to no sublattice. That is true for the gas phase too. The maximum number of sublattices are 10. The constituents of a phase are numbered sequentially from 1 for the first constituent on the first sublattice, to NPCON (See *TQGNPC* on page 32) for the last constituent on the last sublattice. NSCON (L) is the number of constituents on sublattice L. The sum of NSCON over all sublattices is equal to NPCON. Note that constituents that are DORMANT and SUSPENDED still are counted in NPCON and NSCON. They also have a fraction in YFRAC (which must be zero of course). EXTRA may contain extra information about the phase, total mass for example. These are yet to be defined.	
Arguments		
Name	**Type**	**Value set on call or returned**
INDEXP	Integer	Set to a phase index
NSUB	Integer	Return the number of sublattices.
NSCON	Integer array	Return the number of constituents on each sublattice.
SITES	Double precision array	Return the number of sites on each sublattice.
YFRAC	Double precision array	Return the fractions of the constituents.
EXTRA	Double precision array	Return some special values (see Comments)
IWSG	Integer array	Workspace
IWSE	Integer array	Workspace

Examples

To list the constituent names and fractions by sublattices. It is assumed that there are max 10 sublattices and max 500 constituents on all sublattices altogether.

```
DIMENSION NSCON(10),SITES(10),YFRAC(500),EXTRA(5)
```

```
CHARACTER NAME*24
LOGICAL TQGSPC
...
CALL TQGPN(INDEXP,NAME,IWSG,IWSE)
CALL TQGPD(INDEXP,NSUB,NSCON, SITES,YFRAC,EXTRA, &IWSG,IWSE)
KK=0
WRITE(*,190)NAME,NSUB
190 FORMAT(' The phase ',A,' has ',I2,' sublattices')
DO 300 LS=1,NSUB
WRITE(*,191)LS,SITES(LS),NSCON(LS)
191 FORMAT('On sublattice ',I2,' there are ',F8.4,&' sites and',I3,' constituents')
DO 200 LC=1,NSCON(LS)
KK=KK+1
CALL TQGPCN(INDEXP,KK,NAME,IWSG,IWSE)
WRITE(*,192)NAME,YFRAC(KK)
192 FORMAT('Constituent ',A,' has fraction',&1P1E15.8)
200 CONTINUE
300 CONTINUE
```

TQGDF2

Fortran	**TQGDF2 (MODE, IMATR, IPREC, NIE, IIE, XMATR, TEMP, DF, XPREC, XEM, XEP, MUI, IWSG, IWSE)**
C-interface	**tq_gdf2(TC_INT mode,TC_INT imatr,TC_INT iprec,TC_INT nie,TC_INT *iie,TC_FLOAT* xmatr,TC_FLOAT temp,TC_FLOAT* df,TC_FLOAT* xprec,TC_FLOAT* xem,TC_FLOAT* xep,TC_FLOAT* mui,TC_INT* iwsg,TC_INT* iwse);**
Full name:	Get the driving force of nucleation and local equilibrium concentration for a phase transformation under para- or ortho-equilibrium condition.
Purpose:	Obtain data on both the chemical driving force for the nucleation of a precipitate and the local equilibrium concentration at the matrix/precipitate interface under para- or ortho-equilibrium conditions.  See Example 11.
Comments:	For ortho-equilibrium calculations, XPREC can be inputs or outputs, depending

Fortran	**TQGDF2 (MODE, IMATR, IPREC, NIE, IIE, XMATR, TEMP, DF, XPREC, XEM, XEP, MUI, IWSG, IWSE)**	
C-interface	**tq_gdf2(TC_INT mode,TC_INT imatr,TC_INT iprec,TC_INT nie,TC_INT *iie,TC_FLOAT* xmatr,TC_FLOAT temp,TC_FLOAT* df,TC_FLOAT* xprec,TC_FLOAT* xem,TC_FLOAT* xep,TC_FLOAT* mui,TC_INT* iwsg,TC_INT* iwse);**	
	on whether its values are known before the calculation or not. If unknown, the values of XPREC should be set to zero or negative when calling this subroutine and on return one obtains the composition of the precipitate at which the maximum driving force is available. The use of this subroutine for the ortho-equilibrium calculation supersedes that of the obsolete subroutine TQGDF.	
Arguments		
Name	**Type**	**Value set on call or returned**
MODE	Integer	Set type of output and type of composition to use (±1, ±2, and ±3 correspond to mole fraction, weight fraction and U-fraction respectively. Obviously, ±3 has nothing to do with para-equilibrium calculations. If negative, calculate and output only driving force data. This saves the time for equilibrium calculation when you are not interested in local equilibrium concentrations)
IMATR	Integer	Set index of matrix phase
IPREC	Integer	Set index of precipitate phase
NIE	Integer	Set number of interstitial element(s). Zero implies no para-equilibrium calculation
IIE	Integer array	Set index of interstitial element(s), only relevant for para-equilibrium condition
XMATR	Double precision array	Set composition of matrix phase. Composition type depends on MODE
TEMP	Double precision	Set temperature in Kelvin
DF	Double precision	Return driving force in J/mol
XPREC	Double precision array	Return composition of the precipitate phase at the maximum driving force under para/ ortho-equilibrium condition or set to a known composition of the

Fortran	**TQGDF2 (MODE, IMATR, IPREC, NIE, IIE, XMATR, TEMP, DF, XPREC, XEM, XEP, MUI, IWSG, IWSE)**	
C-interface	**tq_gdf2(TC_INT mode,TC_INT imatr,TC_INT iprec,TC_INT nie,TC_INT *iie,TC_FLOAT* xmatr,TC_FLOAT temp,TC_FLOAT* df,TC_FLOAT* xprec,TC_FLOAT* xem,TC_FLOAT* xep,TC_FLOAT* mui,TC_INT* iwsg,TC_INT* iwse);**	
		precipitate in order to get the driving force of phase transformation. Composition type depends on MODE
XEM	Double precision array	Return, if both MODE and DF are positive, local equilibrium composition of matrix phase. Composition type depends on MODE
XEP	Double precision array	Return, if both MODE and DF positive, local equilibrium composition of precipitate phase. Composition type depends on MODE
MUI	Double precision array	Return, if both MODE and DF are positive, chemical potential of interstitial elements. Relevant for only para-equilibrium calculation.
IWSG	Integer array	Workspace
IWSE	Integer array	Workspace

TQGSE

Fortran	**TQGSE (IMATR, IPREC, IMC,TEMP, X, VOLM,VOLP, IWSG, IWSE)**
C-interface	**tq_gse (TC_INT imatr,TC_INT iprec,TC_INT imc,TC_FLOAT temp,TC_FLOAT* x,TC_FLOAT volm,TC_FLOAT volp,TC_INT* iwsg,TC_INT* iwse);**
Full name:	Get interfacial energy between a matrix phase and a precipitate phase.
Purpose:	With this subroutine the application program can estimate the interfacial energy between a matrix phase and a precipitate phase using thermodynamic data from a CALPHAD database. The approximation model is based on Becker's bond energy approach is available as the *Interfacial Energy* model included with the Property Model Calculator and Precipitation Module (TC-PRISMA).
Arguments	

Fortran	**TQGSE (IMATR, IPREC, IMC,TEMP, X, VOLM,VOLP, IWSG, IWSE)**	
C-interface	**tq_gse (TC_INT imatr,TC_INT iprec,TC_INT imc,TC_FLOAT temp,TC_FLOAT* x,TC_FLOAT volm,TC_FLOAT volp,TC_INT* iwsg,TC_INT* iwse);**	
Name	**Type**	**Value set on call or returned**
IMATR	Integer	Set index of matrix phase
IPREC	Integer	Set index of precipitate phase
IMC	Integer	Set index of major component
TEMP	Double precision	Set temperature in Kelvin
X	Double precision array	Set overall alloy composition
VOLM	Double precision	Set molar volume of matrix phase
VOLP	Double precision	Set molar volume of precipitate phase
IWSG	Integer array	Workspace
IWSE	Integer array	Workspace

Troubleshooting Subroutines

Purpose	Subroutine
List status	*TQLS* on the next page
List conditions	*TQLC* on the next page
List equilibrium	*TQLE* on page 68
Force automatic start values	*TQFASV* on page 68
Set default major constituent	*TQSDMC* on page 69
Set start phase constitution	*TQSSPC* on page 69
Set start value of a state variable	*TQSSV* on page 70
Reinitiate the calculation workspace	*TQPINI* on page 71
Set numerical limits	*TQSNL* on page 71
Set maximum number of grid points	*TQSMNG* on page 72
Set equilibrium calculation options	*TQSECO* on page 73
Set error code and give message	*ST1ERR* on page 73
Set error code	*ST2ERR* on page 74
Get error code and give message	*SG1ERR or TQG1ERR* on page 75 *
Get error code	*SG2ERR or TQG2ERR* on page 75 *
Get error code and message	*SG3ERR or TQG3ERR* on page 76 *
Reset error code and message	*RESERR or TQRSERR* on page 77
Save a POLY-3 file	*TQSP3F* on page 78
	* Logical function

TQLS

Fortran	**TQLS(IWSG, IWSE)**	
C-interface	**tq_ls(TC_INT* iwsg,TC_INT* iwse);**	
Full name:	List Status.	
Purpose:	Listing status of all components, phases, and species in a system.	
Comments:	If necessary, use this subroutine to check if the status of all components, phases, and species has been correctly set in an application program. It should only be used for debugging purpose.	
Arguments		
Name	**Type**	**Value set on call or returned**
IWSG	Integer array	Workspace
IWSE	Integer array	Workspace

TQLC

Fortran	**TQLC(IWSG, IWSE)**	
C-interface	**tq_lc(TC_INT* iwsg,TC_INT* iwse);**	
Full name:	List Conditions.	
Purpose:	Listing conditions set for the current equilibrium calculation.	
Comments:	If necessary, use this subroutine to check if the conditions for an equilibrium calculation in the application program has been correctly set. It should only be used for debugging purpose.	
Arguments		
Name	**Type**	**Value set on call or returned**
IWSG	Integer array	Workspace
IWSE	Integer array	Workspace

TQLE

Fortran	**TQLE(IWSG, IWSE)**	
C-interface	**tq_le(TC_INT* iwsg,TC_INT* iwse);**	
Full name:	List Equilibrium.	
Purpose:	Listing results from the most recent equilibrium calculation. The output depends on the package used and the listing displays on the current output unit.	
Comments:	If necessary, use this subroutine to check if an equilibrium calculation is successful. It should only be used for debugging purpose.	
Arguments		
Name	**Type**	**Value set on call or returned**
IWSG	Integer array	Workspace
IWSE	Integer array	Workspace

TQFASV

Fortran	**TQFASV(IWSG, IWSE)**	
C-interface	**tq_fasv(TC_INT* iwsg,TC_INT* iwse);**	
Full name:	Force Automatic Start Value.	
Purpose:	To force automatic start-values for all phases in a single equilibrium calculation.	
Comments:	This is not required unless the calculation fails.	
Arguments		
Name	**Type**	**Value set on call or returned**
IWSG	Integer array	Workspace
IWSE	Integer array	Workspace

TQSDMC

Fortran	**TQSDMC(INDEXP, IWSG, IWSE)**	
C-interface	**tq_sdmc(TC_INT indexp,TC_INT* iwsg,TC_INT* iwse);**	
Full name:	Set Default Major Constituents.	
Purpose:	To set the major phase constituents to the default ones defined in the thermodynamic data file.	
Comments:	Major constituents in a phase can be set in the GES module of Thermo-Calc and then saved into a thermodynamic data file for the use of this interface.	
Arguments		
Name	**Type**	**Value set on call or returned**
INDEXP	Integer	Set as a phase index
IWSG	Integer array	Workspace
IWSE	Integer array	Workspace

TQSSPC

Fortran	**TQSSPC(INDEXP, YF, IWSG, IWSE)**	
C-interface	**tq_sspc(TC_INT indexp,TC_FLOAT* yf,TC_INT* iwsg,TC_INT* iwse);**	
Full name:	Set Start Phase Constitution.	
Purpose:	To set start-values for the constitution of an individual phase.	
Comments:	It is not necessary unless the calculation fails, especially when involving a miscibility gap or an ordering phase.	
Arguments		
Name	**Type**	**Value set on call or returned**
INDEXP	Integer	Set as a phase index
YF	Double precision array	Set to the site fraction of each constituent.

Fortran	TQSSPC(INDEXP, YF, IWSG, IWSE)	
C-interface	tq_sspc(TC_INT indexp,TC_FLOAT* yf,TC_INT* iwsg,TC_INT* iwse);	
IWSG	Integer array	Workspace
IWSE	Integer array	Workspace

TQSSV

Fortran	TQSSV(STAVAR, IP, IC, VALUE, IWSG, IWSE)	
C-interface	tq_ssv(TC_STRING stavar,TC_INT ip,TC_INT ic,TC_FLOAT value,TC_INT* iwsg,TC_INT* iwse);	
Full name:	Set Start Variable.	
Purpose:	To set start-value for a state variable.	
Comments:	It is not necessary unless the calculation fails.	
Arguments		
Name	Type	Value set on call or returned
STAVAR	Character*8	Set as a state variable listed in *Possible State Variables to Set Conditions in TQSETC* on page 40
IP	Integer	Set as a phase index (if needed).
IC	Integer	Set as a component or constituent index (if needed).
VALUE	Double precision	Set to the value.
IWSG	Integer array	Workspace
IWSE	Integer array	Workspace

TQPINI

Fortran	**TQPINI(IWSG, IWSE)**	
C-interface	**tq_pini(TC_INT* iwsg,TC_INT* iwse);**	
Full name:	Poly-3 reINItiation.	
Purpose:	Reinitiate the POLY-3 workspace in Thermo-Calc kernel.	
Comments:	Preparing for a fresh calculation.	
Arguments		
Name	**Type**	**Value set on call or returned**
IWSG	Integer array	Workspace
IWSE	Integer array	Workspace

TQSNL

Fortran	**TQSNL(MAXIT, ACC, YMIN, ADG, IWSG, IWSE)**	
C-interface	**tq_snl(TC_INT maxit,TC_FLOAT acc,TC_FLOAT ymin,TC_FLOAT adg,TC_INT* iwsg,TC_INT* iwse);**	
Full name:	Set Numerical Limits	
Purpose:	To set the Numerical Limits to be used inside POLY-3.	
Comments:	It is not necessary unless the calculation fails.	
Arguments		
Name	**Type**	**Value set on call or returned**
MAXIT	Double precision	Set maximum number of iterations when calculating equilibrium. Default value is 500.
ACC	Double precision	Set required relative accuracy when calculating equilibrium. Default value is 1E-6.
YMIN	Double precision	Set smallest fraction to assign to unstable constituents. Default value is 1E-30.

Fortran	**TQSNL(MAXIT, ACC, YMIN, ADG, IWSG, IWSE)**	
C-interface	**tq_snl(TC_INT maxit,TC_FLOAT acc,TC_FLOAT ymin,TC_FLOAT adg,TC_INT* iwsg,TC_INT* iwse);**	
ADG	Character*	Specify if the calculation should be forced to converge also for the meta stable phases. Legal options are Y or N, where Y means yes and N means no. N is default.
IWSG	Integer array	Workspace
IWSE	Integer array	Workspace

TQSMNG

Fortran	**TQSMNG(NGP, IWSG, IWSE)**	
C-interface	**tq_smng(TC_INT ngp,TC_INT* iwsg,TC_INT* iwse);**	
Full name:	Set Maximum Number of Grid points for each phase.	
Purpose:	To change the maximum number of grid points that can be used for each phase.	
Comments:	The global minimization technique starts with discretizing the composition space and calculating Gibbs energy values at each grid point for each phase. To balance its efficiency and robustness, an appropriate density of grid points should be chosen. The default value of NGP is 2000. In practice, the number of grid points generated during a normal calculation is much less than this value. However, under certain circumstances, one does need to increase the density of grid point for some phases in order to find a true stable equilibrium.	
Arguments		
Name	**Type**	**Value set on call or returned**
NGP	Integer	Number of grid points
IWSG	Integer array	Workspace
IWSE	Integer array	Workspace

TQSECO

Fortran	**TQSECO(IPDH, ICSS, IWSG, IWSE)**	
C-interface	**tq_seco(TC_INT ipdh,TC_INT icss,TC_INT* iwsg,TC_INT* iwse);**	
Full name:	Set Equilibrium Calculation Option.	
Purpose:	To choose equilibrium calculation options.	
Comments:	TQ starts with IPDH=1 and ICSS=1 by default.	
Arguments		
Name	**Type**	**Value set on call or returned**
IPDH	Integer	1 = Force positive definite Hessian0 = Do not force positive definite Hessian
ICSS	Integer	1 = Control stepsize during minimization 0 = Do not control stepsize during minimization
IWSG	Integer array	Workspace
IWSE	Integer array	Workspace

ST1ERR

Fortran	**ST1ERR(IERR, SUBR, MESS)**
C-interface	**tq_st1err(TC_INT ierr,TC_STRING subr,TC_STRING mess);**
Full name:	Set Error Code and Give Message.
Purpose:	This is called when an error that cannot be handled by the current program unit occurs. The error message is printed on the error unit but also saved internally in the error handling package. The program unit should return to the calling program.
Comments:	The error-handling routines are those defined by SGTE for use in the thermodynamic model package. Note that the error-handling is constructed in such a way that when a subroutine detects an error it cannot handle, it should first call an ST* subroutine to set an appropriate error code and then return to the calling subroutine. In that subroutine the error code should be tested, and

Fortran	**ST1ERR(IERR, SUBR, MESS)**	
C-interface	**tq_st1err(TC_INT ierr,TC_STRING subr,TC_STRING mess);**	
	possibly that subroutine can correct the error and proceed, otherwise it should return to its calling subroutine and so on, until either the error is corrected or the top level of the program is reached.In this way it is possible to design a program where minor problems at a low level do not cause program to terminate. Instead, the error is passed up to a higher level where it can be corrected or ignored. The normal subroutines to use are ST2ERR to set the error code and SG2ERR to check it. The other subroutines are less used.NOTE: The TQ subroutines normally do not clear the error code when called. An error set in an earlier subroutine but not tested and detected after that call may cause strange error messages later on. This should be used only for fatal or almost fatal errors.	
Arguments		
Name	**Type**	**Value set on call or returned**
IERR	Integer	Set to an error code.
SUBR	Character*6	Set to the current subroutine name.
MESS	Character*72	Set to the error message to be printed

ST2ERR

Fortran	**ST2ERR(IERR, SUBR, MESS)**	
C-interface	**tq_st2err(TC_INT ierr,TC_STRING subr,TC_STRING mess);**	
Full name:	Set Error Code.	
Purpose:	Called when an error occurs that cannot be handled by the current program unit. The program unit should return to the calling program.	
Comments:	Identical to ST1ERR except that it is silent, i.e., no error message is printed. This should be the normal subroutine to call when detecting errors that should be handled by a higher level of the program.	
Arguments		
Name	**Type**	**Value set on call or returned**

Fortran	**ST2ERR(IERR, SUBR, MESS)**	
C-interface	**tq_st2err(TC_INT ierr,TC_STRING subr,TC_STRING mess);**	
IERR	Integer	Set to an error code.
SUBR	Character*6	Set to the current subroutine name.
MESS	Character*72	Set to the error message to be printed

SG1ERR or TQG1ERR

This is a logical function.

Fortran	**ERROR=SG1ERR(IERR) or ERROR=TQG1ERR(IERR)**	
C-interface	**error=tq_sg1err(TC_INT* ierr);**	
Full name:	Get Error Code and Give Message.	
Purpose:	This is a logical function which could be called after calling a TQ subroutine that can detect an error when the error message should be displayed. If there is an error the function value is .TRUE and the appropriate error code is in IERR. This subroutine also prints the error message on the error unit.	
Comments:	Use when the error is almost fatal. Note that it is possible that the error message is already printed by ST1ERR. Use SG2ERR in most cases. If no error the function value is .FALSE and IERR is zero. If the C-interface is used the value returned is of type: TC_BOOL.	
Arguments		
Name	**Type**	**Value set on call or returned**
IERR	Integer	Set to the error code

SG2ERR or TQG2ERR

This is a logical function.

Fortran	**ERROR=SG2ERR(IERR) or ERROR=TQG2ERR(IERR)**	
C-interface	**error=tq_sg2err(TC_INT* ierr);**	
Full name:	Get Error Code.	
Purpose:	This is a logical function which should be called after calling any TQ subroutine that can detect an error. If there is an error the function value is .TRUE and the appropriate error code is in IERR. This subroutine does not print the error message.	

Comments:	Use for the normal error checking. Note that it is possible that the error message has already been printed by ST1ERR. The program may be able to handle the error to pass it on upwards. If no error the function value is .FALSE and IERR is zero. If the C-interface is used the value returned is of type: TC_ BOOL.
	Example
	`LOGICAL SG2ERR`
	`...`
	`CALL TQCE(' ',IWSG,IWSE)`
	`IF(SG2ERR(IERR)) GOTO 900`
	`...`
	`900 RETURN`	

Arguments		
Name	**Type**	**Value set on call or returned**
IERR	Integer	Set to the error code

SG3ERR or TQG3ERR

This is a logical function.

Fortran	**ERROR=SG3ERR(IERR, SUBR, MESS) or ERROR=TQG3ERR(IERR, SUBR, MESS)**	
C-interface	**error=tq_sg3err(TC_INT* ierr,TC_STRING subr,TC_STRING_LENGTH strlen_subr,TC_STRING mess,TC_STRING_LENGTH strlen_mess);**	
Full name:	Get Error Code and Message.	
Purpose:	This is a logical function which could be called after calling any TQ subroutine that can detect an error. If there is an error the function value is .TRUE and the appropriate error code is in IERR, the subroutine that detected the error in SUBR and the message in MESS. No printing on the error unit. This is useful if the calling program wants to print the message itself in an appropriate context.	
Comments:	This should be used when the error testing subroutine wants to handle the printing of the error message itself. It is possible that the error message has already been printed by ST1ERR. If the C-interface is used the value returned is of type: TC_BOOL.	
Arguments		
Name	**Type**	**Value set on call or returned**
IERR	Integer	Return the error code.
SUBR	Character*6	Return the name of the subroutine detecting an error.
MESS	Character*72	Return the error message

RESERR or TQRSERR

Fortran	**RESERR or TQRSERR**
C-interface	**tq_reserr();**
Full name:	Reset Error Code and Message.
Purpose:	This subroutine resets the error code. A subsequent call to the SG* functions gives no error.
Comments:	This should be used when the error has been cleared so that execution can continue. Unless the error code is cleared by this subroutine the SG* functions continue to report the same error.

Fortran	**RESERR or TQRSERR**
C-interface	**tq_reserr();**
Arguments	None

TQSP3F

Fortran	**TQSP3F(FILE, IWSG, IWSE)**	
C-interface	**tq_sp3f(TC_STRING filename,TC_INT* iwsg,TC_INT* iwse);**	
Full name:	Save the workspaces on a POLY-3 file.	
Purpose:	This subroutine save the current workspaces of a POLY-3 file that can be read into the Thermo-Calc program to see what conditions are set.	
Arguments		
Name	**Type**	**Value set on call or returned**
FILE	Character*72	Set to file name to which workspaces should be saved.
IWSG	Integer array	Workspace
IWSE	Integer array	Workspace

Extra Subroutines–Phase Properties

Purpose	Subroutine
Get Gibbs energy of a phase, Method A, B, and C*	*TQGMA, TQGMB and TQGMC* on the next page
Get 1st partial derivative of Gibbs energy w.r.t. site fractions.*	*TQGMDY* on page 81
Get mobility of a species in a phase.	*TQGMOB* on page 82
Set temperature and pressure for TQGMC, TQGMDY, and TQGMOB.	*TQSTP* on page 83
Set site fractions for TQGMB, TQGMC, TQGMDY, and TQGMOB.	*TQSYF* on page 83
Get index of a system species.	*TQGSSPI* on page 84
Check if Mobility data is available Method A and B	*TQCMOBA and TQCMOBB* on page 84†
Get 1st and 2nd partial derivative of Gibbs energy w.r.t. site fractions.*	*TQDGYY* on page 85
Get constitutional properties of a phase.*	*TQGPHP* on page 86
Convert mole fraction to site fraction for phases with no internal degree of freedom.*	*TQX2Y* on page 87
Convert 1st partial derivative of Gibbs energy w.r.t. site fractions to that w.r.t. mole fractions.*	*TQGMDX* on page 89
*in SI unit for one mole of formula unit.	† Logical function.

TQGMA, TQGMB and TQGMC

Fortran	**TQGMA(INDEXP, TP, YF, VAL, IWSG, IWSE)** **TQGMB(INDEXP, TP, VAL, IWSG, IWSE)** **TQGMC(INDEXP, VAL, IWSG, IWSE)**	
C-interface	**tq_gma(TC_INT indexp,TC_FLOAT* tp,TC_FLOAT* yf,TC_FLOAT* val,TC_INT* iwsg,TC_INT* iwse);tq_gmb(TC_INT indexp,TC_FLOAT* tp,TC_FLOAT* val,TC_INT* iwsg,TC_INT* iwse);** **tq_gmc(TC_INT indexp,TC_FLOAT* val,TC_INT* iwsg,TC_INT* iwse);**	
Full name:	Get Gibbs Energy – Method A, B, and C.	
Purpose:	Getting Gibbs energy of a phase if temperature, pressure, and site fractions are given as arguments or by other subroutines shown.	
Comments:	The returned value is in J/mole of formula unit. Remember this subroutine requires no action of setting condition and calculating equilibrium. It is for getting the Gibbs energy of a single phase with given temperature, pressure and atomic arrangements no matter if the phase is stable, metastable, or unstable in competition with other phases. The application program should take care of the phase stability or phase equilibrium if it is of interest.	
Arguments		
Name	**Type**	**Value set on call or returned**
INDEXP	Integer	Set to a phase index.
TP	Double precision array	Set to temperature and pressure values.
YF	Double precision array	Set to site fraction values in the index order of phase constituent.
VAL	Double precision	Return Gibbs energy value.
IWSG	Integer array	Workspace
IWSE	Integer array	Workspace

TQGMDY

Fortran	**TQGMDY(INDEXP, VARR, IWSG, IWSE)**	
C-interface	**tq_gmdy(TC_INT indexp,TC_FLOAT* varr,TC_INT* iwsg,TC_INT* iwse);**	
Full name:	Get Gibbs Energy and its partial Derivative w.r.t. y-fraction.	
Purpose:	Getting Gibbs energy and its 1st partial derivatives with respect to site fractions for a phase if temperature, pressure, and site fractions have been given by other subroutines shown.	
Comments:	The returned value is in J/mole of formula unit. Remember this subroutine requires no action of setting condition and calculating equilibrium. It is for getting the Gibbs energy and its 1st partial derivative w.r.t site fractions for a single phase with given temperature, pressure and atomic arrangements no matter if the phase is stable, metastable, or unstable in competition with other phases. The application program should take care of the phase stability or phase equilibrium if it is of interest.  This subroutine is demonstrated in Example 9.	
Arguments		
Name	**Type**	**Value set on call or returned**
INDEXP	Integer	Set to a phase index.
VARR	Double precision array	Return values of Gibbs energy and its 1st partial derivatives w.r.t. site fractions in the index order of phase constituents.
IWSG	Integer array	Workspace
IWSE	Integer array	Workspace

TQGMOB

Fortran	**TQGMOB(INDEXP, ISP, VAL, IWSG, IWSE)**	
C-interface	**tq_gmob(TC_INT indexp,TC_INT isp,TC_FLOAT* val,TC_INT* iwsg,TC_INT* iwse);**	
Full name:	Get Mobility	
Purpose:	Getting mobility of a species in a phase with the temperature, pressure, and site fractions given by other subroutines shown.	
Comments:	Remember this subroutine requires no action of setting condition and calculating equilibrium. It is for getting the atomic or species mobility in a single phase with given temperature, pressure and atomic arrangements no matter if the phase is stable, metastable, or unstable in competition with other phases. The application program should take care of the phase stability or phase equilibrium if it is of interest.  The use of this subroutine is demonstrated in Example 9.	
Arguments		
Name	**Type**	**Value set on call or returned**
INDEXP	Integer	Set to a phase index.
ISP	Integer	Set to a system species index
VAL	Double precision	Return species or atomic mobility value.
IWSG	Integer array	Workspace
IWSE	Integer array	Workspace

TQSTP

Fortran	**TQSTP(TP, IWSG, IWSE)**	
C-interface	**tq_stp(TC_FLOAT* tp,TC_INT* iwsg,TC_INT* iwse);**	
Full name:	Set Temperature and Pressure	
Purpose:	Setting temperature and pressure.	
Comments:	This subroutine is used before calling TQGMC, TQGMDY, TQDGYY, and TQGMOB.  See Example 9.	
Arguments		
Name	**Type**	**Value set on call or returned**
TP	Double precision array	Set temperature and pressure.
IWSG	Integer array	Workspace
IWSE	Integer array	Workspace

TQSYF

Fortran	**TQSYF(INDEXP, YF, IWSG, IWSE)**	
C-interface	**tq_syf(TC_INT indexp,TC_FLOAT* yf,TC_INT* iwsg,TC_INT* iwse);**	
Full name:	Set Site Fractions	
Purpose:	Setting site fractions for a phase.	
Comments:	This subroutine is used before calling TQGMB, TQGMC, TQGMDY, TQDGYY, and TQGMOB.  See Example 9.	
Arguments		
Name	**Type**	**Value set on call or returned**

Fortran	TQSYF(INDEXP, YF, IWSG, IWSE)	
C-interface	**tq_syf(TC_INT indexp,TC_FLOAT* yf,TC_INT* iwsg,TC_INT* iwse);**	
INDEXP	Integer	Set to a phase index.
YF	Double precision array	Set to site fraction values in the index order of the phase constituents.
IWSG	Integer array	Workspace
IWSE	Integer array	Workspace

TQGSSPI

Fortran	TQGSSPI(SPN, ISP, IWSG, IWSE)	
C-interface	**tq_gsspi(TC_STRING name,TC_INT* index,TC_INT* iwsg,TC_INT* iwse);**	
Full name:	Get System Species Index	
Purpose:	Getting index of a system species with given name.	
Comments:	Useful if you want to use *TQGMOB* on page 82. See Example 9.	
Arguments		
Name	**Type**	**Value set on call or returned**
SPN	Character*24	Set to a system species name.
ISP	Integer	Return index value of the system species
IWSG	Integer array	Workspace
IWSE	Integer array	Workspace

TQCMOBA and TQCMOBB

These are logical functions.

Fortran	**STATUS=TQCMOBA(INDEXP, ISP, IWSG, IWSE)** **STATUS=TQCMOBB(INDEXP, IWSG, IWSE)**	
C-interface	**status=tq_cmoba(TC_INT indexp,TC_INT* iwsg,TC_INT* iwse);** **status=tq_cmobb(TC_INT indexp,TC_INT* iwsg,TC_INT* iwse);**	
Full name:	Check if Mobility data available – Method A and B	
Purpose:	Check if mobility data have been appended into thermodynamic data file.	
Comments:	If the C-interface is used the value returned is of type: TC_BOOL.	
Arguments		
Name	**Type**	**Value set on call or returned**
INDEXP	Integer	Set to a phase index
ISP	Integer	Set to a system species index
IWSG	Integer array	Workspace
IWSE	Integer array	Workspace

TQDGYY

Fortran	**TQDGYY(INDEXP, VARR1, VARR2, IWSG, IWSE)**
C-interface	**tq_dgyy(TC_INT indexp,TC_FLOAT* varr1,TC_FLOAT* varr2,TC_INT* iwsg,TC_INT* iwse);**
Full name:	Get Gibbs Energy and its 1st and 2nd Partial Derivative w.r.t. site-fractions.
Purpose:	Getting Gibbs energy and its 1st and 2nd partial derivatives with respect to site fractions for a phase if temperature, pressure, and site fractions have been given by other subroutines shown below in this Section.
Comments:	The returned value is in J/mole of formula unit. Remember this subroutine requires no action of setting condition and calculating equilibrium. It is for getting the Gibbs energy and its 1st and 2nd partial derivatives w.r.t site fractions for a single phase with given temperature, pressure and atomic arrangements no matter if the phase is stable, metastable, or unstable in competition with other phases. The application program should take care of the phase stability or phase equilibrium if it is of interest.

Fortran	**TQDGYY(INDEXP, VARR1, VARR2, IWSG, IWSE)**	
C-interface	**tq_dgyy(TC_INT indexp,TC_FLOAT* varr1,TC_FLOAT* varr2,TC_INT* iwsg,TC_INT* iwse);**	
Arguments		
Name	**Type**	**Value set on call or returned**
INDEXP	Integer	Set to a phase index.
VARR1	Double precision array	Return values of Gibbs energy and its 1st partial derivatives w.r.t. site fractions in the index order of phase constituents.
VARR2	Double precision array	Return values of 2nd partial derivatives of Gibbs energy w.r.t. site fractions in the index order of IR: IR=J+I*(I-1)/2, I>=J; IR=I+J*(J-1)/2, I<J, where I and J are row and column indexes of phase constituents, respectively.
IWSG	Integer array	Workspace
IWSE	Integer array	Workspace

TQGPHP

Fortran	**TQGPHP(INDEXP, NE, NCNV, NC, IWORK, WORK, IWSG, IWSE)**
C-interface	**tq_gphp(TC_INT indexp,TC_INT* ne,TC_INT* ncnv,TC_INT* nc,TC_INT* iwork,TC_FLOAT* work,TC_INT* iwsg,TC_INT* iwse);**
Full name:	Get phase constitution properties.
Purpose:	Getting phase constitution properties such as number of components, number of constituents, number of constituents without counting vacancies, etc.
Comments:	This subroutine is designed to speed up conversions of quantities involving mole fractions and site fractions in dynamic calculations where such operations are needed at each local time and space grid point. For each phase involved, one call of this subroutine is enough for subsequent conversions concerning this phases.

Fortran	**TQGPHP(INDEXP, NE, NCNV, NC, IWORK, WORK, IWSG, IWSE)**	
C-interface	**tq_gphp(TC_INT indexp,TC_INT* ne,TC_INT* ncnv,TC_INT* nc,TC_INT* iwork,TC_FLOAT* work,TC_INT* iwsg,TC_INT* iwse);**	
	See Example 10.	
Arguments		
Name	**Type**	**Value set on call or returned**
INDEXP	Integer	Set to a phase index.
NE	Integer	Return number of components
NCNV	Integer	Return number of constituents without counting vacancies
NC	Integer	Return number of constituents
IWORK	Integer array	Return values needed in X to Y conversion, array size >= 4*NCNV
WORK	Double precision array	Return values needed in X to Y conversion, array size >= (NE+1)*NCNV
IWSG	Integer array	Workspace
IWSE	Integer array	Workspace

TQX2Y

Fortran	**TQX2Y(INDEXP, NE, NCNV, NC, IWORK, WORK, XF, YF, IWSG, IWSE)**
C-interface	**tq_x2y(TC_INT indexp,TC_INT ne,TC_INT ncnv,TC_INT nc,TC_INT* iwork,TC_FLOAT* work,TC_FLOAT* xf,TC_FLOAT* yf,TC_INT* iwsg,TC_INT* iwse);**
Full name:	Get Y-fraction given X-fraction.
Purpose:	Converting mole fractions to site fractions in a phase without internal degree of freedom.
Comments:	This subroutine uses the phase constitution properties obtained by TQGPHP as

Fortran	**TQX2Y(INDEXP, NE, NCNV, NC, IWORK, WORK, XF, YF, IWSG, IWSE)**	
C-interface	**tq_x2y(TC_INT indexp,TC_INT ne,TC_INT ncnv,TC_INT nc,TC_INT* iwork,TC_FLOAT* work,TC_FLOAT* xf,TC_FLOAT* yf,TC_INT* iwsg,TC_INT* iwse);**	
	input.  See Example 10.	
Arguments		
Name	**Type**	**Value set on call or returned**
INDEXP	Integer	Set to a phase index
NE	Integer	Set to number of components
NCNV	Integer	Set to number of constituents without counting vacancies
NC	Integer	Set to number of constituents
IWORK	Integer array	Set to values needed in X to Y conversion
WORK	Double precision array	Set to values needed in X to Y conversion
XF	Double precision array	Set to mole fractions
YF	Double precision array	Return site fractions
IWSG	Integer array	Workspace
IWSE	Integer array	Workspace

TQGMDX

Fortran	**TQGMDX(IP, NE, NCNV, NC, IWORK, WORK, YF, VARR, GM, DGDX, XF, IWSG, IWSE)**	
C-interface	**tq_gmdx(TC_INT indexp,TC_INT ne,TC_INT ncnv,TC_INT nc,TC_INT* iwork, TC_FLOAT* work,TC_FLOAT* yf,TC_FLOAT* varr,TC_FLOAT* gm,TC_FLOAT* dgdx,TC_FLOAT* xf,TC_INT* iwsg,TC_INT* iwse);**	
Full name:	Get Gibbs energy and its partial derivative w.r.t. X-fraction.	
Purpose:	Converting Gibbs energy and its 1st partial derivatives with respect to site fractions (VARR obtained by calling TQGMDY) to that w.r.t mole fractions for a phase.	
Comments:	Uses the phase constitution properties obtained by TQGPHP as input. Note VARR obtained by calling TQGMDY is in unit of J/mole of formula unit. GM and DGDX in the present subroutine is in unit of J/mole of atoms. For the use of this subroutine together with *TQGPHP* on page 86 and *TQX2Y* on page 87, see Example 10.	
Arguments		
Name	**Type**	**Value set on call or returned**
IP	Integer	Set to a phase index.
NE	Integer	Set to number of components
NCNV	Integer	Set to number of constituents without counting vacancies
NC	Integer	Set to number of constituents
IWORK	Integer array	Set to values needed in X to Y conversion
WORK	Double precision array	Set to values needed in X to Y conversion
YF	Double precision array	Set to site fractions
VARR	Double precision array	Set to Gibbs energy and its first derivative with respect to site fractions
GM	Double precision	Return Gibbs energy

Fortran	**TQGMDX(IP, NE, NCNV, NC, IWORK, WORK, YF, VARR, GM, DGDX, XF, IWSG, IWSE)**	
C-interface	**tq_gmdx(TC_INT indexp,TC_INT ne,TC_INT ncnv,TC_INT nc,TC_INT* iwork, TC_FLOAT* work,TC_FLOAT* yf,TC_FLOAT* varr,TC_FLOAT* gm,TC_FLOAT* dgdx,TC_FLOAT* xf,TC_INT* iwsg,TC_INT* iwse);**	
DGDX	Double precision array	Return Gibbs energy and its first derivative with respect to mole fractions
XF	Double precision array	Return mole fractions
IWSG	Integer array	Workspace
IWSE	Integer array	Workspace

Database Subroutines

 See Example 12 .

Purpose	Subroutine
Get lists of database names	*TQGDBN* below
Open or switch to a database	*TQOPDB* on the next page
List database elements	*TQLIDE* on page 93
Append a database	*TQAPDB* on page 93
Select an element	*TQDEFEL* on page 94
Reject a selected element	*TQREJEL* on page 94
Reject a phase or all phases	*TQREJPH* on page 95
Restore a phase	*TQRESPH* on page 95
List phases related to the selected element(s)	*TQLISPH* on page 96
List retained phases for the selected element(s)	*TQLISSF* on page 97
Get data from the selected database	*TQGDAT* on page 97
Reject defined system and reinitiate workspace	*TQREJSY* on page 98

TQGDBN

Fortran	**TQGDBN(DB_ARR, N, IWSG, IWSE)**
C-interface	**tq_gdbn(tc_databases_strings* databases,TC_INT* n,TC_INT* iwsg,TC_INT* iwse);**
Full name:	Get Database Names and Number.
Purpose:	Get the names and total number of thermodynamic and kinetic databases listed in the database initiation file of Thermo-Calc: TC_INITD.TDB.
Comments:	IERR = 1001 is Failed to find the initiation file.
Arguments	

Fortran	**TQGDBN(DB_ARR, N, IWSG, IWSE)**	
C-interface	**tq_gdbn(tc_databases_strings* databases,TC_INT* n,TC_INT* iwsg,TC_INT* iwse);**	
Name	**Type**	**Value set on call or returned**
DB_ARR	Character*24 array	Return database names.
N	Integer	Return total number of databases available.
IWSG	Integer array	Workspace
IWSE	Integer array	Workspace

TQOPDB

Fortran	**TQOPDB(TDB, IWSG, IWSE)**	
C-interface	**tq_opdb(TC_STRING database,TC_INT* iwsg,TC_INT* iwse);**	
Full name:	Open Database.	
Purpose:	Open a thermodynamic or kinetic database.	
Comments:	IERR = 1001 Failed to find the initiation file. IERR = 1002 Database or its license not available.	
Arguments		
Name	**Type**	**Value set on call or returned**
TDB	Character*256	Set to the name of a database.
IWSG	Integer array	Workspace
IWSE	Integer array	Workspace

TQLIDE

Fortran	**TQLIDE(EL_ARR, N, IWSG, IWSE)**	
C-interface	**tq_lide(tc_elements_strings* elements,TC_INT* num,TC_INT* iwsg,TC_INT* iwse);**	
Full name:	List Database Elements.	
Purpose:	List all elements available in the chosen database.	
Arguments		
Name	**Type**	**Value set on call or returned**
EL_ARR	Character*2 array	Return the names of all elements.
N	Integer	Return the total number of elements.
IWSG	Integer array	Workspace
IWSE	Integer array	Workspace

TQAPDB

Fortran	**TQAPDB(TDB, IWSG, IWSE)**	
C-interface	**tq_apdb(TC_STRING database,TC_INT* iwsg,TC_INT* iwse);**	
Full name:	Append Database.	
Purpose:	Append a thermodynamic or kinetic database.	
Comments:	IERR = 1002 Database or its license not available.	
Arguments		
Name	**Type**	**Value set on call or returned**
TDB	Character*256	Set to the name of a database.
IWSG	Integer array	Workspace
IWSE	Integer array	Workspace

TQDEFEL

Fortran	**TQDEFEL(ELNAM, IWSG, IWSE)**	
C-interface	**tq_defel(TC_STRING element,TC_INT* iwsg,TC_INT* iwse);**	
Full name:	Define Element.	
Purpose:	Define a system element.	
Comments:	IERR = 1011 Element not included in the chosen database. IERR = 1012 Element already defined.	
Arguments		
Name	**Type**	**Value set on call or returned**
ELNAM	Character*2	Set to the name of an element.
IWSG	Integer array	Workspace
IWSE	Integer array	Workspace

TQREJEL

Fortran	**TQREJEL(ELNAM, IWSG, IWSE)**	
C-interface	**tq_rejel(TC_STRING element,TC_INT* iwsg,TC_INT* iwse);**	
Full name:	Reject Element.	
Purpose:	Reject a defined system element.	
Comments:	IERR = 1013 Element not included in the chosen database. IERR = 1014 Element already rejected.	
Arguments		
Name	**Type**	**Value set on call or returned**
ELNAM	Character*2	Set to the name of an element.
IWSG	Integer array	Workspace
IWSE	Integer array	Workspace

TQREJPH

Fortran	**TQREJPH(PHNAM, IWSG, IWSE)**	
C-interface	**tq_rejph(TC_STRING phase,TC_INT* iwsg,TC_INT* iwse);**	
Full name:	Reject Phase.	
Purpose:	Reject a system phase.	
Comments:	IERR = 1017 Phase not included in the chosen database.IERR = 1018 Phase already rejected.	
Arguments		
Name	**Type**	**Value set on call or returned**
PHNAM	Character*24	Set to a phase name If * is used then all phases are rejected
IWSG	Integer array	Workspace
IWSE	Integer array	Workspace

TQRESPH

Fortran	**TQRESPH(PHNAM, IWSG, IWSE)**	
C-interface	**tq_resph(TC_STRING phase,TC_INT* iwsg,TC_INT* iwse);**	
Full name:	Restore Phase.	
Purpose:	Restore a rejected system phase.	
Comments:	IERR = 1015 Phase not included in the chosen database. IERR = 1016 Phase already restored.	
Arguments		
Name	**Type**	**Value set on call or returned**

Fortran	**TQRESPH(PHNAM, IWSG, IWSE)**	
C-interface	**tq_resph(TC_STRING phase,TC_INT* iwsg,TC_INT* iwse);**	
PHNAM	Character*24	Set to a phase name If * is used then all phases are rejected
IWSG	Integer array	Workspace
IWSE	Integer array	Workspace

TQLISPH

Fortran	**TQLISPH(PH_ARR, N, IWSG, IWSE)**	
C-interface	**tq_lisph(tc_phases_strings* phases,TC_INT* num,TC_INT* iwsg,TC_INT* iwse);**	
Full name:	List System Phase.	
Purpose:	List all phases (both rejected and restored) available for the defined system.	
Arguments		
Name	**Type**	**Value set on call or returned**
PH_ARR	Character*24 array	Return phase names.
N	Integer	Return the total number of phases
IWSG	Integer array	Workspace
IWSE	Integer array	Workspace

TQLISSF

Fortran	**TQLISSF(PH_ARR, N, IWSG, IWSE)**	
C-interface	**tq_lissf(tc_phases_strings* phases,TC_INT* num,TC_INT* iwsg,TC_INT* iwse);**	
Full name:	List Selected System Phase.	
Purpose:	List phases not rejected for the defined system.	
Arguments		
Name	**Type**	**Value set on call or returned**
PH_ARR	Character*24 array	Return phase names.
N	Integer	Return the total number of phases
IWSG	Integer array	Workspace
IWSE	Integer array	Workspace

TQGDAT

Fortran	**TQGDAT(IWSG, IWSE)**	
C-interface	**tq_gdat(TC_INT* iwsg,TC_INT* iwse);**	
Full name:	Get Data.	
Purpose:	Get data for the defined system from the chosen database.	
Arguments		
Name	**Type**	**Value set on call or returned**
IWSG	Integer array	Workspace
IWSE	Integer array	Workspace

TQREJSY

Fortran	**TQREJSY(IWSG, IWSE)**	
C-interface	**tq_rejsy(TC_INT* iwsg,TC_INT* iwse);**	
Full name:	Reject system.	
Purpose:	Reject the defined system and reinitiate the workspace in order to do a completely new calculation for a different system selected from the same or a different database.	
Comments:	In any application programs, either *TQINI* on page 16 or *TQINI3* on page 15 should be called only once. If there is a need to do a completely new calculation on a totally different system without exiting the application program, one should call TQREJS instead before going to (open a new database and) define a new system, get data, and make calculations.	
Arguments		
Name	**Type**	**Value set on call or returned**
IWSG	Integer array	Workspace
IWSE	Integer array	Workspace

Adaptive Interpolation Schemes

Also see *About Adaptive Interpolation Schemes* on page 13.

In order to perform a simulation using the scheme, the TQ-library must be initialized in the normal way using the routine *TQINI* on page 16 and thermodynamic information must be loaded, usually with the *TQRFIL* on page 18 routine.

The scheme is then initialized using the TQIPS_INIT_TOP routine and each branch in the calculation is initialized using the TQIPS_INIT_BRANCH routine. For each set of interpolated values which are to be defined and obtained from a certain branch of the scheme, the TQIPS_INIT_FUNCTION routine is called. The values for all functions defined in the branch are then returned using the TQIPS_GET_VALUE routine.

Purpose	Subroutine
Initiate the interpolation scheme	*TQIPS_INIT_TOP* below
Initiate a branch in the interpolation scheme	*TQIPS_INIT_BRANCH* on the next page
Define a function or state variable to be interpolated	*TQIPS_INIT_FUNCTION* on page 103
Retrieve the interpolated value	*TQIPS_GET_VALUE* on page 104
Write the data of the interpolation scheme to file.	*TQIPS_WRITE_IPS_DATA_TO_FILE* on page 105
Read interpolation scheme data from file.	*TQIPS_READ_IPS_DATA_FROM_FILE* on page 106
Get statistics on the usage of the interpolation scheme.	*TQIPS_GET_MEMORY_USAGE* on page 107

TQIPS_INIT_TOP

Fortran	**TQIPS_INIT_TOP(IERR, IWSG, IWSE)**
C-interface	**tq_ips_init_top(TC_INT* err,TC_INT* iwsg,TC_INT* iwse)**
Full name:	Initiate the top structure of the adaptive interpolation scheme.
Purpose:	Initiates the top structure of the interpolation scheme which may contain several branches with different conditions, phases and values to be interpolated.
Comments:	IERR is returned with 0 if no error occurs.

Fortran	**TQIPS_INIT_TOP(IERR, IWSG, IWSE)**	
C-interface	**tq_ips_init_top(TC_INT* err,TC_INT* iwsg,TC_INT* iwse)**	
Arguments		
Name	**Type**	**Value set on call or returned**
IERR	Integer	Returns the error code.
IWSG	Integer array	Workspace.
IWSE	Integer array	Workspace.

TQIPS_INIT_BRANCH

Fortran	**TQIPS_INIT_BRANCH(TISCOND, TISCNST, PISCOND, PISCNST, IDEPEL, IDISCRT, NSTEP, IPHSTA, T, TMIN, TMAX, P, PMIN, PMAX, RMEMFR, PHAMNT,XMIN, XMAX, IBRANCH, IERR IWSG, IWSE**
C-interface	**tq_ips_init_branch(TC_BOOL t_is_condition, TC_BOOL t_is_constant, TC_BOOL p_is_condition, TC_BOOL p_is_constant, TC_BOOL* independent_elements,TC_INT dicretization_type, TC_INT nr_of_ steps, TC_INT* state_of_phases, TC_FLOAT t, TC_FLOAT tmin, TC_ FLOAT tmax,TC_FLOAT p,TC_FLOAT pmin, TC_FLOAT pmax, TC_FLOAT memory_fraction, TC_FLOAT* amount_of_phases, TC_FLOAT* xmin, TC_FLOAT* xmax, TC_INT* branch_nr, TC_INT* err, TC_INT* iwsg, TC_INT* iwse)**
Full name:	Initiate a branch of the adaptive interpolation scheme.
Purpose:	Initiates a branch of the interpolation scheme with a set of conditions, phases and values to be interpolated.
Comments:	The size of the arrays IDEPEL, XMIN and XMAX are defined as the number of all components supplied by TQGNC must be provided in the same order as supplied by TQGCOM.Composition conditions are set as the normalized number of moles for each component (N(c) =value).The size of the arrays IPHSTA and PHAMNT are defined as the number of all phases supplied by TQGNP must be provided in the same order as supplied by TQGPN.The allocation of memory to each branch is performed at the first time values are retrieved using TQIPS_GET_VALUE, therefore some considerations must be made when using several branches in order to allocate the same amount of memory to each branch.

Fortran	**TQIPS_INIT_BRANCH(TISCOND, TISCNST, PISCOND, PISCNST, IDEPEL, IDISCRT, NSTEP, IPHSTA, T, TMIN, TMAX, P, PMIN, PMAX, RMEMFR, PHAMNT,XMIN, XMAX, IBRANCH, IERR IWSG, IWSE**	
C-interface	**tq_ips_init_branch(TC_BOOL t_is_condition, TC_BOOL t_is_constant, TC_BOOL p_is_condition, TC_BOOL p_is_constant, TC_BOOL* independent_elements,TC_INT dicretization_type, TC_INT nr_of_ steps, TC_INT* state_of_phases, TC_FLOAT t, TC_FLOAT tmin, TC_ FLOAT tmax,TC_FLOAT p,TC_FLOAT pmin, TC_FLOAT pmax, TC_FLOAT memory_fraction, TC_FLOAT* amount_of_phases, TC_FLOAT* xmin, TC_FLOAT* xmax, TC_INT* branch_nr, TC_INT* err, TC_INT* iwsg, TC_INT* iwse)**	
Arguments		
Name	**Type**	**Value set on call or returned**
TISCOND	Logical	Set to TRUE if temperature is a condition in this branch.
TISCNST	Logical	Set to TRUE if temperature is constant in this branch.
PISCOND	Logical	Set to TRUE if pressure is a condition in this branch.
PISCNST	Logical	Set to TRUE if pressure is constant in this branch.
IDEPEL	Logical array	Set to TRUE for each element for which conditions are present.
IDISCRT	Integer	Indicates the type of discretization where: 1=linear2=logarithmic
NSTEP	Integer	Set to the logarithm (10 base) number of steps to be used to interpolate in the temperature / pressure / composition space.
IPHSTA	Integer array	Set to the status for the phases in this branch, where: 1=ENTERED, 2=SUSPENDED, 3=DORMANT, 4=FIXED
T	Double precision	Set to temperature if temperature is a condition,

Fortran	**TQIPS_INIT_BRANCH(TISCOND, TISCNST, PISCOND, PISCNST, IDEPEL, IDISCRT, NSTEP, IPHSTA, T, TMIN, TMAX, P, PMIN, PMAX, RMEMFR, PHAMNT,XMIN, XMAX, IBRANCH, IERR IWSG, IWSE**	
C-interface	**tq_ips_init_branch(TC_BOOL t_is_condition, TC_BOOL t_is_constant, TC_BOOL p_is_condition, TC_BOOL p_is_constant, TC_BOOL* independent_elements,TC_INT dicretization_type, TC_INT nr_of_ steps, TC_INT* state_of_phases, TC_FLOAT t, TC_FLOAT tmin, TC_ FLOAT tmax,TC_FLOAT p,TC_FLOAT pmin, TC_FLOAT pmax, TC_FLOAT memory_fraction, TC_FLOAT* amount_of_phases, TC_FLOAT* xmin, TC_FLOAT* xmax, TC_INT* branch_nr, TC_INT* err, TC_INT* iwsg, TC_INT* iwse)**	
		otherwise used as starting value.
TMIN	Double precision	Set to lower limit of temperature range.
TMAX	Double precision	Set to upper limit of temperature range.
P	Double precision	Set to pressure if pressure is a condition, otherwise used as starting value.
PMIN	Double precision	Set to lower limit of pressure range.
PMAX	Double Precision	Set to upper limit of pressure range.
RMEMFR	Double precision	Set to the fraction of the amount of free physical memory to be allocated to the interpolation scheme for this branch (value < 1.0). If a value larger than 1.0 if set, it is interpreted as the number of megabytes allocated to the branch.
PHAMNT	Double precision array	Set to the amount of the phase if defined as a fixed phase with IPHSTA.
XMIN	Double precision array	Set to the lower limit of the composition range of each component.
XMAX	Double precision array	Set to the upper limit of the composition range of each component.
IBRANCH	Integer	Set to branch number for which the variable in STRING is to be interpolated.
IERR	Integer	Returns error code.

Fortran	**TQIPS_INIT_BRANCH(TISCOND, TISCNST, PISCOND, PISCNST, IDEPEL, IDISCRT, NSTEP, IPHSTA, T, TMIN, TMAX, P, PMIN, PMAX, RMEMFR, PHAMNT,XMIN, XMAX, IBRANCH, IERR IWSG, IWSE**	
C-interface	**tq_ips_init_branch(TC_BOOL t_is_condition, TC_BOOL t_is_constant, TC_BOOL p_is_condition, TC_BOOL p_is_constant, TC_BOOL* independent_elements,TC_INT dicretization_type, TC_INT nr_of_ steps, TC_INT* state_of_phases, TC_FLOAT t, TC_FLOAT tmin, TC_ FLOAT tmax,TC_FLOAT p,TC_FLOAT pmin, TC_FLOAT pmax, TC_FLOAT memory_fraction, TC_FLOAT* amount_of_phases, TC_FLOAT* xmin, TC_FLOAT* xmax, TC_INT* branch_nr, TC_INT* err, TC_INT* iwsg, TC_INT* iwse)**	
IWSG	Integer array	Workspace.
IWSE	Integer array	Workspace

TQIPS_INIT_FUNCTION

Fortran	**TQIPS_INIT_FUNCTION(STRING, IBRANCH, IERR, IWSG, IWSE)**	
C-interface	**tq_ips_init_function(TC_STRING function_string, TC_INT branch_nr, TC_INT* err, TC_INT* iwsg, TC_INT* iwse);**	
Full name:	Initiates a function for a specific branch whose value(s) are to be retrieved from the adaptive interpolation scheme.	
Purpose:	Initiates a function or state variable for a specific branch whose value(s) are to be retrieved from the adaptive interpolation scheme.	
Arguments		
Name	**Type**	**Value set on call or returned**
STRING	Character*128	Set to the name of the function or state variable to be interpolated, wildcards (*) may be used in place of element and/or phase names.
IBRANCH	Integer	Set to branch number for which the variable in STRING is to be interpolated.
IERR	Integer	Returns the error code.

Fortran	**TQIPS_INIT_FUNCTION(STRING, IBRANCH, IERR, IWSG, IWSE)**	
C-interface	**tq_ips_init_function(TC_STRING function_string, TC_INT branch_nr, TC_INT* err, TC_INT* iwsg, TC_INT* iwse);**	
IWSG	Integer array	Workspace
IWSE	Integer array	Workspace

TQIPS_GET_VALUE

Fortran	**TQIPS_GET_VALUE(IBRANCH, NOSCHEME, ARR, RESULT, IERR, ISHORT, IWSG, IWSE)**	
C-interface	**tq_ips_get_value(TC_INT branch_nr, TC_INT noscheme, TC_FLOAT* variable_values, TC_FLOAT* function_values, TC_INT* err, TC_INT* shortcut, TC_INT* iwsg,TC_INT* iwse);**	
Full name:	Retrieve interpolated value(s) from the adaptive interpolation scheme.	
Purpose:	Retrieves all the values defined by all TQIPS_INIT_FUNCTION defined for branch IBRANCH in sequential order.	
Arguments		
Name	**Type**	**Value set on call or returned**
IBRANCH	Integer	Set to branch number.
NOSCHEME	Integer	Set to 1 if the interpolation scheme is to be disabled.
ARR	Double precision array	Array set to the mole-fractions of all the components followed by the temperature and the pressure, if a component is dependent the value may be arbitrary. The same applies if the temperature or pressure is constant.
RESULT	Double precision array	Returns the interpolated values in the same order as they were defined in TQS_INIT_FUNCTION.
IERR	Integer	Returns the error code.
ISHORT	Integer	Set to the last returned value or zero, 0.

Fortran	**TQIPS_GET_VALUE(IBRANCH, NOSCHEME, ARR, RESULT, IERR, ISHORT, IWSG, IWSE)**	
C-interface	**tq_ips_get_value(TC_INT branch_nr, TC_INT noscheme, TC_FLOAT* variable_values, TC_FLOAT* function_values, TC_INT* err, TC_INT* shortcut, TC_INT* iwsg,TC_INT* iwse);**	
		Returns a shortcut to data pertaining to the grid point in virtual composition/temperature/pressure space for the values in ARR
IWSG	Integer array	Workspace.
IWSE	Integer array	Workspace.

TQIPS_WRITE_IPS_DATA_TO_FILE

Fortran	**TQIPS_WRITE_IPS_DATA_TO_FILE(FILENAME,IERR,IWSG,IWSE)**	
C-interface	**tq_ips_write_ips_data_to_file(TC_STRING filename, TC_INT* ierr, TC_INT* iwsg, TC_INT* iwse)**	
Full name:	Write the data of the interpolation scheme to file.	
Purpose:	To save all the data of the interpolation scheme in order to read them at a later time with routine tqips_read_ips_data_from_file.	
Arguments		
Name	**Type**	**Value set on call or returned**
FILENAME	Character*256	The name of the file to be saved
IERR	Integer	Returns the error code
IWSG	Integer array	Workspace
IWSE	Integer array	Workspace

TQIPS_READ_IPS_DATA_FROM_FILE

Fortran	**TQIPS_READ_IPS_DATA_FROM_FILE(FILENAME, MEMORY_FRACTION, IERR, IWSG, IWSE)**	
C-interface	**tq_ips_read_ips_data_from_file(TC_STRING filename, TC_FLOAT* memory_fraction, TC_INT* ierr, TC_INT* iwsg, TC_INT* iwse)**	
Full name:	Read interpolation scheme data from file.	
Purpose:	To read from file interpolation scheme data that has been saved previously with routine tqips_write_ips_data_to_file.	
Comments:	If memory_fraction has a value smaller than zero the amount of allocated memory will be determined by the value read from file. If memory_fraction is larger than zero it will be interpreted in the same way as argument RMEMFR of routine tqips_init_branch.	
Arguments		
Name	**Type**	**Value set on call or returned**
FILENAM	Character*256	The name of the file to be read
MEMORY_ FRACTION	double precision	See comment
IERR	Integer	Returns the error code
IWSG	Integer array	Workspace
IWSE	Integer array	Workspace

TQIPS_GET_MEMORY_USAGE

Fortran	**TQIPS_GET_MEMORY_USAGE(IBRANCH, FRACTION, ISLOTS, IUSEDSLOTS, ICALLS, IEQCALCS, IERR, IWSG, IWSE)**	
C-interface	**tq_ips_get_memory_usage(TC_INT branch_nr, TC_FLOAT* fraction,TC_INT* total_number_of_data_slots, TC_INT* number_of_used_data_slots, TC_INT* total_number_of_calls, TC_INT* total_number_of_equil_calcs, TC_INT* ierr,TC_INT* iwsg,TC_INT* iwse)**	
Full name:	Get statistics on the usage of the interpolation scheme.	
Purpose:	To get some statistics on the performance of the interpolation scheme.	
Arguments		
Name	**Type**	**Value set on call or returned**
IBRANCH	Integer	If IBRANCH>0 it is the branch number for which the data should be returned. If IBRANCH=0 then data is returned summed over all branches.
FRACTION	double precision	This is simply equal to IUSEDSLOTS/ISLOTS
ISLOTS	Integer	The total number of data slots allocated
IUSEDSLOTS	Integer	The number of used data slots
ICALLS	Integer	The number of calls to tqips_get_value
IEQCALCS	Integer	The number of equilibrium calculations performed by Thermo-Calc on behalf of the interpolation scheme
IERR	Integer	error code
IWSG	Integer array	Workspace
IWSE	Integer array	Workspace

Composition Set Reordering Routines

Purpose	Subroutine
Initialize IWSR workspace for reordering of CS in TQ.	*TQROINIT* below
Set ideal composition in this phase	*TQSETRX* on the next page
Reorder CS in current EQ	*TQORDER* on the next page
List content of IWSR set by user.	*TQLROX* on page 110

TQROINIT

Fortran	**TQROINIT(NWSR, IWSR, IWSG, IWSE)**	
C-interface	**tq_roinit(TC_INT nwsr, TC_INT* iwsr,TC_INT* iwsg, TC_INT* iwse);**	
Full name:	Initialize IWSR workspace for reordering of CS in TQ.	
Purpose:	With this subroutine the application program initializes the Thermo-Calc package for use of the reordering subroutines. It must be called before using any of the subroutines TQSETRX, TQORDER, TQLROX.	
Comments:	NWSR=1000 should be enough for several composition sets	
Arguments		
Name	**Type**	**Value set on call or returned**
NWSR	Integer	On call set to size of the workspace IWSR.
IWSR	Integer array	Memory area for storage of data inside the package.
IWSG	Integer array	Workspace
IWSE	Integer array	Workspace

TQSETRX

Fortran	**TQSETRX(PHASE, X, IWSR, IWSG, IWSE)**	
C-interface	**tq_setrx(TC_STRING phase, TC_FLOAT* x, TC_INT* iwsr, TC_INT* iwsg, TC_INT* iwse);**	
Full name:	Set ideal composition in this phase	
Purpose:	Store composition of phase in IWSR for future use.	
Comments:	The order in the X array is the order of the components in the system	
Arguments		
Name	**Type**	**Value set on call or returned**
Phase	Character*24	Phase name (e.g. 'fcc#2')
X	Double precision array	On call set to the ideal composition in this composition set in this phase.
IWSR	Integer array	Workspace
IWSG	Integer array	Workspace
IWSE	Integer array	Workspace

TQORDER

Fortran	**TQORDER(IWSR, IWSG, IWSE)**
C-interface	**tq_order(TC_INT* iwsr, TC_INT* iwsg,TC_INT* iwse);**
Full name:	Reorder CS in current EQ
Purpose:	The ideal composition set by the user is used to reorder the CS in respective phase to minimize the distance compared to present eq.
Comments:	Calling routines more than once in a row should affect nothing. Routines minimize the distance between the set ideal composition and the composition found in the present equilibria, and reorder the CS in the equilibria to achieve the minima. This does not affect the properties of the equilibria.
Arguments	

Fortran	**TQORDER(IWSR, IWSG, IWSE)**	
C-interface	**tq_order(TC_INT* iwsr, TC_INT* iwsg,TC_INT* iwse);**	
Name	**Type**	**Value set on call or returned**
IWSR	Integer array	Workspace
IWSG	Integer array	Workspace
IWSE	Integer array	Workspace

TQLROX

Fortran	**TQLROX(IWSR, IWSG, IWSE)**	
C-interface	**tq_lrox(TC_INT* iwsr,TC_INT* iwsg,TC_INT* iwse);**	
Full name:	List content of IWSR set by user.	
Purpose:	List the ideal composition set in the output unit using TQSETRX in IWSR. It is for debugging.	
Arguments		
Name	**Type**	**Value set on call or returned**
IWSR	Integer array	Workspace
IWSG	Integer array	Workspace
IWSE	Integer array	Workspace

Compiler Settings

Also see *Programming Languages* on page 9.

In the compiler flag paths, *<libraryversion>* is the current name of the library that changes between software releases. Look through your operating system's file structure to determine the current name.

Compiling FORTRAN code

Windows: Visual Studio 2010, Intel FORTRAN Composer 12

32-bit configuration

Compiler flags:

```
/iface:default
```

Example:

```
ifort /iface:default  /c tqex01.F
ifort/exe:tqex01.exe tqex01.obj libtq-win-Win32-<libraryversion>.lib
```

64-bit configuration

Compiler flags:

```
/integer_size:64
/real_size:64
/double_size:64
/iface:default
```

Example:

```
ifort /integer_size:64 /real_size:64 /double_size:64 /iface:default /c
tqex01.F
ifort/exe:tqex01.exe tqex01.obj libtq-win-x64-<libraryversion>.lib
```

Linux: GNU compiler version 4.4

64-bit configuration

Compiler flags:

```
-fdefault-real-8
-fdefault-double-8
-fdefault-integer-8
```

Example:

```
gfortran –c -fdefault-real-8 -fdefault-double-8 \fdefault-integer-8  tqex01.F
gfortran –o tqex01 tqex01.o libtq-linux-x86_64-gfortran44-<libraryversion>.so
```

Linux: Intel FORTRAN Compiler

64-bit configuration

Compiler flags:

```
-real-size 64
-double-size 64
-integer-size 64
```

Example:

```
ifort –c real-size 64 -double-size 64 \ integer-size 64 tqex01.F
ifort –o tqex01 tqex01.o libtq-linux-x86_64-ifort-<libraryversion>.so
```

Compiling C code

When compiling the C-code it is necessary to include the files **tqroot.h** and **tc_data_defs.h**, therefore the path to where these files are located must be specified.

Windows: Visual Studio 2010

C programs linked with TQ in Windows, must use release libraries (/MT or /MD) due to clashes in the memory allocation routines causing the global minimization procedure to fail if debug libraries are used.

32-bit configuration

Compiler flags:

```
/DWIN32
/I..\tq\C\include
```

Example:

```
cl /c /DWIN32 /I..\tq\C\include tqex01.c
link /OUT:tqex01.exe tqex01.obj libtq-win-Win32-<libraryversion>.lib
```

64-bit configuration

Compiler flags:

```
/DWIN32
/DWIN64
/I..\tq\C\include
```

Example:

```
cl /c /DWIN32 /DWIN64 /I..\tq\C\include tqex01.c
link /OUT:tqex01.exe tqex01.obj libtq-win-x64-<libraryversion>.lib
```

Linux: GNU compiler version 4.4

64-bit configuration

Compiler flags:

```
-I../tq/C/include
```

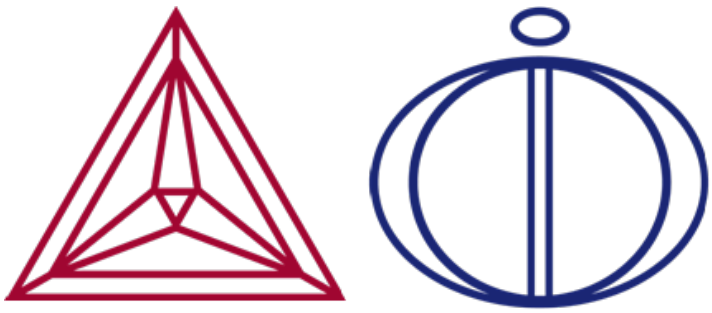
Example:

```
gcc –c -I../tq/C/include tqex01.c
gcc –o tqex01 tqex01.o libtq-linux-x86_64-gfortran44-<libraryversion>.so
```

TC-Toolbox for MATLAB®

SDK Programmer's Guide

Version 2016b



Thermo-Calc Software

Introduction to the TC-Toolbox for MATLAB®

Thermo-Calc is a general software package for manipulation of thermodynamic quantities and multicomponent phase equilibrium calculations. Currently, there are three application programming interfaces available for Thermo-Calc: TQ-Interface, TC-API and TC-Toolbox for MATLAB. In this guide TC-Toolbox for MATLAB, the interface between Thermo-Calc and MATLAB®, is discussed.

In this section:

About TC-Toolbox for MATLAB® *3*

Installing TC-Toolbox for MATLAB® *4*

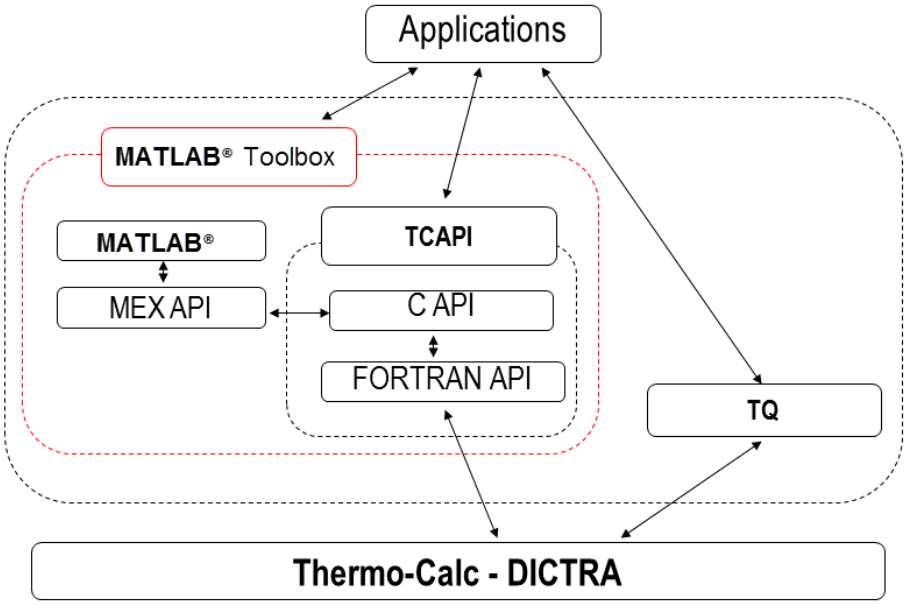
TC-Toolbox for MATLAB Examples *4*

About TC-Toolbox for MATLAB®

The concept of the application programming interfaces for Thermo-Calc is that an application programmer does not need to understand the Thermo-Calc kernel but can use its powerful features in other programs.

MATLAB is a flexible software for technical computing and visualization of data. The software has more than 600 mathematical, statistical and engineering functions, and graphical capabilities. It is a matrix-oriented programming language and contains compilers, links and libraries for different scientific applications. This flexibility is enhanced with being able to retrieve thermodynamic and kinetic quantities through the TC-Toolbox for MATLAB. This programming interface is ideal for fast realization of ideas during research and development activities.

Thermo-Calc APIs



To be able to call MATLAB from programs written in C or FORTRAN there are MEX-files (MATLAB Executable) included with the MATLAB software. These MEX-files were utilised when interfacing MATLAB with Thermo-Calc.

For every Thermo-Calc function implemented in the MEX-files there is a corresponding m-file, making it possible to call Thermo-Calc from MATLAB just by running the corresponding m-file.

More than 50 commands are available for the application programmer. For more information, general functionality and applications of the MATLAB software refer to the documentation provided by the MathWorks Ltd. (www.mathworks.com/help/).

Installing TC-Toolbox for MATLAB®

TC-Toolbox for MATLAB® needs to be installed on the same computer or on a server with the Thermo-Calc software and database package. TC-Toolbox for MATLAB is available for Windows operating systems.

For installation details, see in the *Thermo-Calc Installation Guide* .

Test the Installation

Once the installation is complete, you can test the connectivity in MATLAB.

Start MATLAB and type: **tc_init_root** in the command window and press return. This should result in no return message for a successful installation. All of the commands available in the toolbox are described in this document.

To get a short description of each command type in the command window **help Thermo-Calc-Toolbox *X*** (where *X* is the installed version number of the toolbox).

TC-Toolbox for MATLAB Examples

Examples for the TC-Toolbox are placed under MATLAB in the same documents folder as the Thermo-Calc files (My Documents or Public Documents):

```
..\Documents\MATLAB\Thermo-Calc-Toolbox-X\Examples
```

Where *X* is the installed version number of the toolbox.

For most installations the examples are available in the MATLAB window when the software is opened.

For installation details, see in the *Thermo-Calc Installation Guide*.

Example Descriptions

Example Name	Description
ex01.m	Calculation of a single equilibrium in Fe-Cr-C at 1200 K.
ex02.m	Calculation of a molar Gibbs energy surface in an Al-Cu-Si alloy.
calc_para_eq.m paraf.m qparaf.m	Example 3. calc_para_eq.m (calls the functions paraf.m and qparaf.m) Calculation of paraequilibrium and quasi-paraconditions for an alloy with at least one interstitial component (e.g. N or C). Demonstrates also the coding of an interactive program.

Example Name	Description
ex04.m	Calculation of the so-called T-zero line in Fe-C.
ex05.m	Calculation of the influence of composition on the A3 temperature in an Fe-Cr-C alloy. The A3 temperature is calculated for a large number of uniformly distributed compositions in composition space. The carbon content belongs to the interval [1E-4:5E-3] (weight fraction) and the cromium content belongs to the interval [1E-2,3E-2]. The relative frequency (the fraction) of compositions belonging to a certain A3 temperature interval is then plotted.
ex06_interfacial_energy.m	Calculation of interfacial energy between BCC and M7C3 for a Fe-12Cr-0.1C alloy.

Commands in TC-Toolbox for MATLAB®

To avoid conflict with reserved names all commands in the TC-Toolbox for MATLAB® start with **tc_** and the DICTRA module commands start with **dic_**.

Group	Description
tc_root below	General information and miscellaneous commands
tc_database on the next page	Information and commands in the database module
tc_system on the next page	Information and commands in the database module
tc_util on page 10	Various commands e.g. “tc_define_system”
tc_ges5 on page 10	Information and commands in the GES5 module
dic_dictra on page 10	Information and commands in the DICTRA module

tc_root

Name	Arguments	Description
`tc_init_root`	None	Initialise the Thermo-Calc subsystem. Must be called before any other command in the Toolbox.
`tc_deinit`	None	Closes the Thermo-Calc session and returns the license key.
`tc_version`	string: version_name	Returns the current version of the Thermo-Calc subsystem.
`tc_poly3_command`	string: command	Sends a command to the POLY-3 module.
`tc_read_poly3_file`	string: file_name	Reads stored POLY-3 file name.
`tc_save_poly3_file`	string: file_name	Saves a POLY-3 file name.

tc_database

Name	Arguments	Description
`tc_append_database`	string: database_name	Appends database name.
`tc_element_select`	string: element_name	Selects an element name from the current database.
`tc_get_data`	None	Executes the GET_DATA command.
`tc_open_database`	string: database_name	Opens a named database.
`tc_phase`	integer: no_phases string array phase_names	Returns the number (no.) of phases and the phase names.
`tc_phase_reject`	string: phase_name	Rejects phase name in the current database.
`tc_phase_select`	string: phase_name	Selects phase name in the current database.

tc_system

Name	Arguments	Description
`tc_error`	integer: error_code string: error_message	Checks if an error occurred, then returns an error code and message.
`tc_reset_error`	None	Resets the error handling in the Thermo-Calc subsystem
`tc_compute_equilibrium`	None	Executes the COMPUTE_EQULIBRIUM command in POLY-3
`tc_component_status`	string: status string: comp_name	Returns the status for component (comp) name. The status can be ENTERED or SUSPENDED.
`tc_create_new_equilibrium`	integer: eq_number	Create a new equilibrium with equilibrium number.
`tc_define_components`	string: new_components	Changes the set of components to those in new components.
`tc_degrees_of_freedom`	integer: number	Returns the degrees of freedom number in the system.

Name	Arguments	Description
`tc_delete_condition`	string: condition_name	Deletes the named condition.
`tc_delete_symbol`	string: symbol_name	Deletes the named symbol.
`tc_enter_constant`	string: constant_name double: value	Enters a symbol of type CONSTANT with constant_name and value.
`tc_enter_function`	string: function_name string: function_ expression	Enters a symbol of type FUNCTION with function_name and expression.
`tc_enter_symbol`	string: symbol_name string: symbol_type integer: argument_type integer: int_value double: double_value string: char_value	Enters a named symbol and type (=CONSTANT, FUNCTION, TABLE or VARIABLE) with an argument type (=1 for integer, 2 for double or 3 for string).
`tc_enter_table`	string: table_name string: table_expression	Enters a symbol of type TABLE with table_name and expression.
`tc_enter_variable`	string: variable_name double: value	Enters a symbol of type VARIABLE with variable_name and value.
`tc_get_derivatives`	string: phase string array: arr1 string array: arr2	Returns the Gibbs energy and the first and second derivatives with respect to site-fractions for phase. The array arr1 contains the Gibbs energy and the first derivatives and the array arr2 contains the second derivatives.
`tc_get_value`	string: expression double: value	Retrieves the current value of any state variable, function or variable set in expression.
`tc_list_component`	integer: no_components string array: components	Returns the number (no.) of components and a list of all components.
`tc_list_conditions`	integer: no_conditions string array: conditions	Returns the number (no.) of conditions and a list of all conditions.

Name	Arguments	Description
`tc_list_phase`	integer: no_phases string array: phases	Returns the number (no.) of phases and a list of all phases.
`tc_list_species`	integer: no_species string array: species	Returns the number (no.) of species and a list of all species.
`tc_list_symbols`	integer: no_symbols string array: symbols	Returns the number (no.) of symbols and a list of all defined symbols.
`tc_phase_status`	string: status string: phase_name	Returns the status for the named phase.
`tc_select_equilibrium`	integer: eq_number	Command to switch to another set of conditions and equilibria. The desired set of conditions and equilibria are indicated by its equilibrium (eq) number.
`tc_set_component_status`	string: comp_name string: status	Sets the status (ENTERED or SUSPENDED) for a named component.
`tc_set_condition`	string: expression double: value	Sets a condition for expression to value.
`tc_set_minimization`	string: flag	Turns global minimization on or off by setting the string flag to on or off.
`tc_set_phase_addition`	string: phase_name double: value	Command to add a value to the Gibbs energy expression of a named phase.
`tc_set_phase_status`	string: phase_name string: status double: value	Sets status (ENTERED, DORMANT, FIXED or SUSPENDED) to a named phase. A value is to set for status ENTERED and FIXED.
`tc_set_start_value`	string: name double: value	Sets a start value for a state variable name.
`tc_species_status`	string: status string: species_name	Returns the status for a named species.

tc_util

Name	Arguments	Description
`tc_check_error`	string:	Checks for errors and resets them. It is a combination of tc_error and tc_ reset_error.
`tc_define_system`	string: database_name string: element_names string: reject_phases string: restore_phases	Define a system with a named database, element names, phases to reject and phases to restore.
`tc_prompt`	string: tprompt integer: defval	Prompt to input an integer value.
`tc_promptr`	string: tprompt double:defval	Prompt to input a double value.
`tc_prompts`	string: tprompt string: defval	Prompt to input a string.
`tc_promptsn`	string: tprompt string array: defval	Prompt to input a string array.

tc_ges5

Name	Arguments	Description
`tc_enter_ges5_ parameter`	string: parameter_name string: parameter_ expression	Enters a named parameter in parameter_ expression.
`tc_ges5_command`	string: command	Sends a command to the GES5 monitor.
`tc_get_ges5_ parameter`	string: parameter_ expression string: parameter_name	Returns a parameter_expression for parameter_name.

dic_dictra

A Diffusion Module (DICTRA) license is required to use these commands.

Name	Arguments	Description
`dic_command`	string: command	Sends a command to the DICTRA module.
`dic_convert_ sitefractions`	double array: new_fractions string: phase_name double array: sitefractions integer: fraction_type	Convert site fractions in for a named phase. Set new fractions and fraction type=1, 2, 3 return mole-, mass- or u-fractions, respectively.
`dic_get_ independent_ component`	integer: no_idepc string array: comp_names string region_name	Returns the number (no.) of independent components (idepc) and a list of component names for a named region.
`dic_list_profile`	integer: no_gridpoints integer: no_sitefractions double: sitefractions double array: gridpoints string: region_name string: phase_name	Returns a stored profile for a named phase and region.
`dic_list_timesteps`	integer: no_timesteps double array: timesteps	Returns the number (no.) of time steps and a list of time steps.
`dic_read_workspace`	string: file_name	Reads the stored simulation file name.
`dic_region_info`	integer: no_gridpoints double: region_size double: start_coordinate string: region_name	Returns information about the named region: the size of the region, number (no.) of grid points and value of the first (start) coordinate.
`dic_save_workspace`	string: file_name	Saves a simulation file to a new name.
`dic_select_timestep`	integer: time_step	Selects a time step from a stored simulation file.
`dic_simulate_ reaction`	None	Start the simulation.

TC-API

SDK Programmer's Guide

Version 2016b

About the TC-API Programmer's Guide

Installation File Location

A shortcut to this guide is also included in the SDK installation directory. For example, for a network installation on Windows, the directory is here:

`C:\Users\<username>\Documents\Thermo-Calc\<version>\SDK\`

On Windows, once Thermo-Calc and the SDKs are installed go to **Start → All Programs** or **All Apps → Thermo-Calc** and click **SDK** to open the folder.

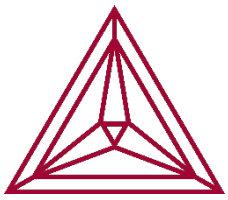
For installation and directory locations, see the *Thermo-Calc Installation Guide*.

Download the PDF from the Website

You can also download the PDF from the website

1. Go to the Thermo-Calc website.
2. From the menu, select **Support>Documentation**.
3. To the right of **Software Development Kits (SDK)** click **Read More**.
4. Click **TC-API Programmer's Guide** to open the PDF in your web browser.
5. Click **Download**.

TC-API: SDK Programmer's Guide
Version 2016b



TC-API

7.5-SNAPSHOT

Generated by Doxygen 1.8.9.1

Tue Oct 11 2016 11:14:45

Contents

1 Thermo-Calc c-api **1**

1.1 Installed files . . . 1

1.1.1 TC-API libraries. . . . 1

1.1.2 Source folder. . . . 1

1.1.3 Project folders for building the example code . . . 1

1.2 Explicit loading or Dynamic linking . . . 1

1.3 About the source code . . . 2

1.3.1 The dynamic linking examples use the files . . . 2

1.3.2 The explicit loading examples use the files . . . 2

1.4 When starting developing proprietary projects the following code is needed . . . 2

1.4.1 The dynamic linking examples use the files . . . 2

1.4.2 The explicit loading examples use the files . . . 2

2 Class Index **5**

2.1 Class List . . . 5

3 File Index **7**

3.1 File List . . . 7

4 Class Documentation **9**

4.1 _tc_function_library Struct Reference . . . 9

4.1.1 Detailed Description . . . 10

4.1.2 Member Data Documentation . . . 10

4.1.2.1 tc_append_database . . . 10

4.1.2.2 tc_check_license . . . 10

4.1.2.3 tc_component_status . . . 10

4.1.2.4 tc_compute_equilibrium . . . 10

4.1.2.5 tc_create_new_equilibrium . . . 10

4.1.2.6 tc_database . . . 11

4.1.2.7 tc_define_components . . . 11

4.1.2.8 tc_degrees_of_freedom . . . 11

4.1.2.9 tc_deinit . . . 11

4.1.2.10 tc_delete_condition . . . 11
4.1.2.11 tc_delete_symbol . . . 11
4.1.2.12 tc_element . . . 11
4.1.2.13 tc_element_reject . . . 11
4.1.2.14 tc_element_select . . . 11
4.1.2.15 tc_enter_ges5_parameter . . . 11
4.1.2.16 tc_enter_symbol . . . 11
4.1.2.17 tc_error . . . 11
4.1.2.18 tc_ges5 . . . 12
4.1.2.19 tc_get_data . . . 12
4.1.2.20 tc_get_derivatives . . . 12
4.1.2.21 tc_get_ges5_parameter . . . 12
4.1.2.22 tc_get_value . . . 12
4.1.2.23 tc_init_root . . . 12
4.1.2.24 tc_init_root3 . . . 12
4.1.2.25 tc_list_component . . . 12
4.1.2.26 tc_list_conditions . . . 12
4.1.2.27 tc_list_phase . . . 12
4.1.2.28 tc_list_species . . . 12
4.1.2.29 tc_list_symbols . . . 12
4.1.2.30 tc_nr_of_constituents_in_phase . . . 13
4.1.2.31 tc_open_database . . . 13
4.1.2.32 tc_phase . . . 13
4.1.2.33 tc_phase_all_constituents . . . 13
4.1.2.34 tc_phase_constituents . . . 13
4.1.2.35 tc_phase_reject . . . 13
4.1.2.36 tc_phase_select . . . 13
4.1.2.37 tc_phase_status . . . 13
4.1.2.38 tc_phase_structure . . . 13
4.1.2.39 tc_poly3 . . . 13
4.1.2.40 tc_put_sitefractions . . . 13
4.1.2.41 tc_read_poly3_file . . . 13
4.1.2.42 tc_reject_constituent . . . 14
4.1.2.43 tc_reset_error . . . 14
4.1.2.44 tc_restore_constituent . . . 14
4.1.2.45 tc_save_poly3_file . . . 14
4.1.2.46 tc_select_equilibrium . . . 14
4.1.2.47 tc_set_component_status . . . 14
4.1.2.48 tc_set_condition . . . 14
4.1.2.49 tc_set_license_code . . . 14

4.1.2.50 tc_set_minimization_option . . . 14
4.1.2.51 tc_set_phase_addition . . . 14
4.1.2.52 tc_set_phase_status . . . 14
4.1.2.53 tc_set_start_value . . . 14
4.1.2.54 tc_species_status . . . 15
4.1.2.55 tc_version . . . 15
4.2 tc_components_strings Struct Reference . . . 15
4.2.1 Detailed Description . . . 15
4.2.2 Member Data Documentation . . . 15
4.2.2.1 component . . . 15
4.3 tc_conditions_as_arrays_of_strings Struct Reference . . . 15
4.3.1 Detailed Description . . . 15
4.3.2 Member Data Documentation . . . 16
4.3.2.1 condition . . . 16
4.4 tc_constituents_strings Struct Reference . . . 16
4.4.1 Detailed Description . . . 16
4.4.2 Member Data Documentation . . . 16
4.4.2.1 constituent . . . 16
4.5 tc_databases_strings Struct Reference . . . 16
4.5.1 Detailed Description . . . 16
4.5.2 Member Data Documentation . . . 16
4.5.2.1 database . . . 16
4.6 tc_elements_strings Struct Reference . . . 17
4.6.1 Detailed Description . . . 17
4.6.2 Member Data Documentation . . . 17
4.6.2.1 element . . . 17
4.7 tc_phases_strings Struct Reference . . . 17
4.7.1 Detailed Description . . . 17
4.7.2 Member Data Documentation . . . 17
4.7.2.1 phase . . . 17
4.8 tc_reference_strings Struct Reference . . . 17
4.8.1 Detailed Description . . . 18
4.8.2 Member Data Documentation . . . 18
4.8.2.1 reference . . . 18
4.9 tc_species_strings Struct Reference . . . 18
4.9.1 Detailed Description . . . 18
4.9.2 Member Data Documentation . . . 18
4.9.2.1 specie . . . 18

5 File Documentation **19**

5.1 example1.c File Reference . . . 19
5.2 example2.c File Reference . . . 19
5.2.1 Function Documentation . . . 19
5.2.1.1 main . . . 19
5.3 example3.c File Reference . . . 19
5.3.1 Typedef Documentation . . . 20
5.3.1.1 ivect . . . 20
5.3.1.2 rvect . . . 20
5.3.1.3 str8 . . . 20
5.3.1.4 strvect . . . 20
5.3.2 Function Documentation . . . 20
5.3.2.1 main . . . 20
5.4 libtc.c File Reference . . . 20
5.4.1 Function Documentation . . . 20
5.4.1.1 importFunctions . . . 20
5.4.1.2 tcloadfunc . . . 21
5.5 libtc.h File Reference . . . 21
5.5.1 Typedef Documentation . . . 22
5.5.1.1 BoolFuncIntPStringInt . . . 22
5.5.1.2 BoolFuncStringStringStrLen . . . 22
5.5.1.3 FloatFuncString . . . 22
5.5.1.4 IntFuncNoParams . . . 22
5.5.1.5 IntFuncString . . . 22
5.5.1.6 IntFuncStringInt . . . 22
5.5.1.7 IntFuncStringIntIntP . . . 22
5.5.1.8 IntFuncStringIntPStringIntFloatP . . . 22
5.5.1.9 IntFuncStringIntPStringStrLenFloatP . . . 22
5.5.1.10 IntFuncStringString . . . 22
5.5.1.11 StringFuncString . . . 22
5.5.1.12 tc_function_library . . . 23
5.5.1.13 VoidFuncInt . . . 23
5.5.1.14 VoidFuncIntPIntPIntPIntP . . . 23
5.5.1.15 VoidFuncNoParams . . . 23
5.5.1.16 VoidFuncString . . . 23
5.5.1.17 VoidFuncStringFloat . . . 23
5.5.1.18 VoidFuncStringFloatP . . . 23
5.5.1.19 VoidFuncStringFloatPFloatP . . . 23
5.5.1.20 VoidFuncStringInt . . . 23
5.5.1.21 VoidFuncStringIntInt . . . 23
5.5.1.22 VoidFuncStringIntString . . . 23

5.5.1.23 VoidFuncStringString . . . 23
5.5.1.24 VoidFuncStringStringFloat . . . 23
5.5.1.25 VoidFuncStringStringInt . . . 24
5.5.1.26 VoidFuncStringStringIntIntFloatString . . . 24
5.5.2 Function Documentation . . . 24
5.5.2.1 importFunctions . . . 24
5.6 ReadMe.txt File Reference . . . 24
5.7 ReadMe.txt File Reference . . . 24
5.8 tc_data_defs.h File Reference . . . 24
5.8.1 Macro Definition Documentation . . . 25
5.8.1.1 BOOL_FUNC_WIN . . . 25
5.8.1.2 DllExport . . . 25
5.8.1.3 false . . . 25
5.8.1.4 FLOAT_FUNC_WIN . . . 26
5.8.1.5 INTEGER_FUNC . . . 26
5.8.1.6 INTEGER_FUNC_GNU . . . 26
5.8.1.7 INTEGER_FUNC_WIN . . . 26
5.8.1.8 TC_EPS . . . 26
5.8.1.9 TC_MAX_NR_OF_AXES . . . 26
5.8.1.10 TC_MAX_NR_OF_CONST_PER_SUBLATTICE . . . 26
5.8.1.11 TC_MAX_NR_OF_CONST_PER_SUBLATTICE_IN_IDEAL_GAS . . . 26
5.8.1.12 TC_MAX_NR_OF_CONSTITUENTS . . . 26
5.8.1.13 TC_MAX_NR_OF_DATABASES . . . 26
5.8.1.14 TC_MAX_NR_OF_ELEMENTS . . . 26
5.8.1.15 TC_MAX_NR_OF_PHASES . . . 26
5.8.1.16 TC_MAX_NR_OF_SPECIES . . . 27
5.8.1.17 TC_MAX_NR_OF_SUBLATTICES . . . 27
5.8.1.18 TC_NWSE . . . 27
5.8.1.19 TC_NWSG . . . 27
5.8.1.20 TC_STRLEN_COMPONENTS . . . 27
5.8.1.21 TC_STRLEN_CONSTITUENTS . . . 27
5.8.1.22 TC_STRLEN_DATABASE . . . 27
5.8.1.23 TC_STRLEN_ELEMENTS . . . 27
5.8.1.24 TC_STRLEN_MAX . . . 27
5.8.1.25 TC_STRLEN_PATH_MAX . . . 27
5.8.1.26 TC_STRLEN_PHASES . . . 27
5.8.1.27 TC_STRLEN_REFERENCE . . . 27
5.8.1.28 TC_STRLEN_SPECIES . . . 28
5.8.1.29 TC_STRLEN_STOICHIOMETRY . . . 28
5.8.1.30 TC_VARS . . . 28

5.8.1.31 TCFuncExport . . . 28
5.8.1.32 TCHANDLE . . . 28
5.8.1.33 true . . . 28
5.8.1.34 VOID_FUNC_WIN . . . 28
5.8.2 Typedef Documentation . . . 28
5.8.2.1 pointer . . . 28
5.8.2.2 TC_BOOL . . . 28
5.8.2.3 tc_components_strings . . . 28
5.8.2.4 tc_conditions_as_arrays_of_strings . . . 28
5.8.2.5 tc_constituents_strings . . . 28
5.8.2.6 tc_databases_strings . . . 29
5.8.2.7 tc_elements_strings . . . 29
5.8.2.8 TC_FLOAT . . . 29
5.8.2.9 TC_IARR . . . 29
5.8.2.10 TC_INT . . . 29
5.8.2.11 TC_LABEL_STRING . . . 29
5.8.2.12 tc_phases_strings . . . 29
5.8.2.13 tc_reference_strings . . . 29
5.8.2.14 tc_species_strings . . . 29
5.8.2.15 TC_STRING . . . 29
5.8.2.16 TC_STRING_LENGTH . . . 29
5.9 tcapi.h File Reference . . . 29
5.9.1 Function Documentation . . . 31
5.9.1.1 tc_append_database . . . 31
5.9.1.2 tc_check_license . . . 31
5.9.1.3 tc_component_status . . . 31
5.9.1.4 tc_compute_equilibrium . . . 31
5.9.1.5 tc_create_new_equilibrium . . . 31
5.9.1.6 tc_database . . . 31
5.9.1.7 tc_define_components . . . 31
5.9.1.8 tc_degrees_of_freedom . . . 31
5.9.1.9 tc_deinit . . . 31
5.9.1.10 tc_delete_condition . . . 31
5.9.1.11 tc_delete_symbol . . . 32
5.9.1.12 tc_element . . . 32
5.9.1.13 tc_element_reject . . . 32
5.9.1.14 tc_element_select . . . 32
5.9.1.15 tc_enter_ges5_parameter . . . 32
5.9.1.16 tc_enter_symbol . . . 32
5.9.1.17 tc_error . . . 32

5.9.1.18 tc_ges5 . . . 32
5.9.1.19 tc_get_data . . . 32
5.9.1.20 tc_get_derivatives . . . 32
5.9.1.21 tc_get_ges5_parameter . . . 32
5.9.1.22 tc_get_value . . . 33
5.9.1.23 tc_init_root . . . 33
5.9.1.24 tc_init_root3 . . . 33
5.9.1.25 tc_list_component . . . 33
5.9.1.26 tc_list_conditions . . . 33
5.9.1.27 tc_list_phase . . . 33
5.9.1.28 tc_list_species . . . 33
5.9.1.29 tc_list_symbols . . . 33
5.9.1.30 tc_nr_of_constituents_in_phase . . . 33
5.9.1.31 tc_open_database . . . 33
5.9.1.32 tc_phase . . . 33
5.9.1.33 tc_phase_all_constituents . . . 34
5.9.1.34 tc_phase_constituents . . . 34
5.9.1.35 tc_phase_reject . . . 34
5.9.1.36 tc_phase_select . . . 34
5.9.1.37 tc_phase_status . . . 34
5.9.1.38 tc_phase_structure . . . 34
5.9.1.39 tc_poly3 . . . 34
5.9.1.40 tc_put_sitefractions . . . 34
5.9.1.41 tc_read_poly3_file . . . 34
5.9.1.42 tc_reject_constituent . . . 34
5.9.1.43 tc_reset_error . . . 34
5.9.1.44 tc_restore_constituent . . . 35
5.9.1.45 tc_save_poly3_file . . . 35
5.9.1.46 tc_select_equilibrium . . . 35
5.9.1.47 tc_set_component_status . . . 35
5.9.1.48 tc_set_condition . . . 35
5.9.1.49 tc_set_license_code . . . 35
5.9.1.50 tc_set_minimization_option . . . 35
5.9.1.51 tc_set_phase_addition . . . 35
5.9.1.52 tc_set_phase_status . . . 35
5.9.1.53 tc_set_start_value . . . 35
5.9.1.54 tc_species_status . . . 35
5.9.1.55 tc_version . . . 35
5.10 tcExamples.c File Reference . . . 36
5.10.1 Typedef Documentation . . . 36

5.10.1.1 ivect . . . 36
5.10.1.2 rvect . . . 36
5.10.1.3 str8 . . . 36
5.10.1.4 strvect . . . 36
5.10.2 Function Documentation . . . 36
5.10.2.1 example1 . . . 36
5.10.2.2 example2 . . . 36
5.10.2.3 example3 . . . 37
5.11 tcExamples.h File Reference . . . 37
5.11.1 Function Documentation . . . 37
5.11.1.1 example1 . . . 37
5.11.1.2 example2 . . . 37
5.11.1.3 example3 . . . 37
5.12 tcMain.c File Reference . . . 37
5.12.1 Macro Definition Documentation . . . 38
5.12.1.1 TC_API_LIBRARY_NAME . . . 38
5.12.2 Function Documentation . . . 38
5.12.2.1 importLibThermoCalc . . . 38
5.12.2.2 loadTCLibraryInCurrentDir . . . 38
5.12.2.3 main . . . 38
5.13 tcutils.c File Reference . . . 38
5.13.1 Function Documentation . . . 38
5.13.1.1 getTempEnvironmentPath . . . 38
5.13.1.2 getThermoCalcEnvironmentPath . . . 38
5.14 tcutils.h File Reference . . . 38
5.14.1 Macro Definition Documentation . . . 39
5.14.1.1 getCurrentWorkingDir . . . 39
5.14.1.2 SLASH . . . 39
5.14.1.3 TC61_HOME . . . 39
5.14.1.4 TCPATH . . . 39
5.14.1.5 TEMP . . . 39
5.14.2 Function Documentation . . . 39
5.14.2.1 getTempEnvironmentPath . . . 39
5.14.2.2 getThermoCalcEnvironmentPath . . . 39

Index **41**

Chapter 1

Thermo-Calc c-api

The main part of this manual is a technical description of the TC-API. To find details on the Thermodynamic applications of each library function see the section on tcapi.h.

1.1 Installed files

In the distribution of Thermo-Calc c-api, the following folders and files can be found.

1.1.1 TC-API libraries.

These could be .lib, .dll, .so -files depending on your installation.
E.g. on a 64 bit Windows system you will find:
tcapi-win-x64-7.5-SNAPSHOT.dll and tcapi-win-x64-7.5-SNAPSHOT.lib

1.1.2 Source folder.

This is the c code for running the the different examples. Some of this code can be reused in user-written projects.

1.1.3 Project folders for building the example code

Linux:
-> Linux/Linux-Dynamic-Linking
-> Linux/Linux-Explicit-Loading
Windows:
-> Windows-Mingw32-Explicit-Loading
-> Windows-Studio-Project-Dynamic-Linking
-> Windows-Studio-Project-Explicit-Loading

1.2 Explicit loading or Dynamic linking

When using Dynamic Linking, the libraries (.so or .lib) and the header-files of the tcapi are needed when the program is being built.

When using Explicit Loading no libraries are needed for the build. They are loaded at runtime (.so or .dll).

1.3 About the source code

1.3.1 The dynamic linking examples use the files

Simple thermodynmic calculations to demonstrate the use of this api.
example1.c
example2.c
example3.c

Utility functions for finding the correct environment variables
tcutils.c
tcutils.h

Declarations and documentation on all TC-API functions
tcapi.h

Thermo-Calc proprietary declarations and definitions. DO NOT EDIT.
tc_data_defs.h

1.3.2 The explicit loading examples use the files

Utilities to simplify working with explicitly loaded dll.
libtc.c
libtc.h

Simple thermodynmic calculations to demonstrate the use of this api.
tcExamples.c
tcExamples.h

Main program to exemplify hos this api can be used
tcMain.c

Utility functions for finding the correct environment variables
tcutils.c
tcutils.h

Thermo-Calc proprietary declarations and definitions. DO NOT EDIT.
tc_data_defs.h

1.4 When starting developing proprietary projects the following code is needed

1.4.1 The dynamic linking examples use the files

tcapi.h
tc_data_defs.h

1.4.2 The explicit loading examples use the files

libtc.c
libtc.h
tc_data_defs.h

(tcMain.c can give an idea of how the above files can be used)

Chapter 2

Class Index

2.1 Class List

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

_tc_function_library 9
tc_components_strings 15
tc_conditions_as_arrays_of_strings 15
tc_constituents_strings 16
tc_databases_strings 16
tc_elements_strings 17
tc_phases_strings 17
tc_reference_strings 17
tc_species_strings 18

Chapter 3

File Index

3.1 File List

Here is a list of all files with brief descriptions:

example1.c . . . 19
example2.c . . . 19
example3.c . . . 19
libtc.c . . . 20
libtc.h . . . 21
tc_data_defs.h . . . 24
tcapi.h . . . 29
tcExamples.c . . . 36
tcExamples.h . . . 37
tcMain.c . . . 37
tcutils.c . . . 38
tcutils.h . . . 38

Chapter 4

Class Documentation

4.1 _tc_function_library Struct Reference

`#include <libtc.h>`

Public Attributes

- VoidFuncString tc_append_database
- BoolFuncStringStringStrLen tc_check_license
- StringFuncString tc_component_status
- VoidFuncNoParams tc_compute_equilibrium
- VoidFuncInt tc_create_new_equilibrium
- IntFuncStringInt tc_database
- VoidFuncStringIntInt tc_define_components
- IntFuncNoParams tc_degrees_of_freedom
- VoidFuncNoParams tc_deinit
- VoidFuncString tc_delete_condition
- VoidFuncString tc_delete_symbol
- IntFuncStringInt tc_element
- VoidFuncString tc_element_reject
- VoidFuncString tc_element_select
- VoidFuncStringString tc_enter_ges5_parameter
- VoidFuncStringStringIntIntFloatString tc_enter_symbol
- BoolFuncIntPStringInt tc_error
- VoidFuncString tc_ges5
- VoidFuncNoParams tc_get_data
- VoidFuncStringFloatPFloatP tc_get_derivatives
- VoidFuncStringStringInt tc_get_ges5_parameter
- FloatFuncString tc_get_value
- IntFuncNoParams tc_init_root
- IntFuncStringString tc_init_root3
- VoidFuncStringInt tc_list_component
- VoidFuncStringInt tc_list_conditions
- VoidFuncStringInt tc_list_phase
- VoidFuncStringInt tc_list_species
- IntFuncStringIntIntP tc_list_symbols
- IntFuncString tc_nr_of_constituents_in_phase
- VoidFuncString tc_open_database
- IntFuncStringInt tc_phase

- IntFuncStringIntPStringIntFloatP tc_phase_all_constituents
- IntFuncStringIntPStringIntFloatP tc_phase_constituents
- VoidFuncString tc_phase_reject
- VoidFuncString tc_phase_select
- StringFuncString tc_phase_status
- IntFuncStringIntPStringStrLenFloatP tc_phase_structure
- VoidFuncString tc_poly3
- VoidFuncStringFloatP tc_put_sitefractions
- VoidFuncString tc_read_poly3_file
- VoidFuncStringIntString tc_reject_constituent
- VoidFuncNoParams tc_reset_error
- VoidFuncStringIntString tc_restore_constituent
- VoidFuncString tc_save_poly3_file
- VoidFuncInt tc_select_equilibrium
- VoidFuncStringString tc_set_component_status
- VoidFuncStringFloat tc_set_condition
- VoidFuncInt tc_set_license_code
- VoidFuncIntPIntPIntPIntP tc_set_minimization_option
- VoidFuncStringFloat tc_set_phase_addition
- VoidFuncStringStringFloat tc_set_phase_status
- VoidFuncStringFloat tc_set_start_value
- StringFuncString tc_species_status
- VoidFuncStringInt tc_version

4.1.1 Detailed Description

Definition at line 67 of file libtc.h.

4.1.2 Member Data Documentation

4.1.2.1 VoidFuncString _tc_function_library::tc_append_database

Definition at line 68 of file libtc.h.

4.1.2.2 BoolFuncStringStringStrLen _tc_function_library::tc_check_license

Definition at line 69 of file libtc.h.

4.1.2.3 StringFuncString _tc_function_library::tc_component_status

Definition at line 70 of file libtc.h.

4.1.2.4 VoidFuncNoParams _tc_function_library::tc_compute_equilibrium

Definition at line 71 of file libtc.h.

4.1.2.5 VoidFuncInt _tc_function_library::tc_create_new_equilibrium

Definition at line 72 of file libtc.h.

4.1.2.6 IntFuncStringInt _tc_function_library::tc_database

Definition at line 73 of file libtc.h.

4.1.2.7 VoidFuncStringIntInt _tc_function_library::tc_define_components

Definition at line 74 of file libtc.h.

4.1.2.8 IntFuncNoParams _tc_function_library::tc_degrees_of_freedom

Definition at line 75 of file libtc.h.

4.1.2.9 VoidFuncNoParams _tc_function_library::tc_deinit

Definition at line 76 of file libtc.h.

4.1.2.10 VoidFuncString _tc_function_library::tc_delete_condition

Definition at line 77 of file libtc.h.

4.1.2.11 VoidFuncString _tc_function_library::tc_delete_symbol

Definition at line 78 of file libtc.h.

4.1.2.12 IntFuncStringInt _tc_function_library::tc_element

Definition at line 79 of file libtc.h.

4.1.2.13 VoidFuncString _tc_function_library::tc_element_reject

Definition at line 80 of file libtc.h.

4.1.2.14 VoidFuncString _tc_function_library::tc_element_select

Definition at line 81 of file libtc.h.

4.1.2.15 VoidFuncStringString _tc_function_library::tc_enter_ges5_parameter

Definition at line 82 of file libtc.h.

4.1.2.16 VoidFuncStringStringIntIntFloatString _tc_function_library::tc_enter_symbol

Definition at line 83 of file libtc.h.

4.1.2.17 BoolFuncIntPStringInt _tc_function_library::tc_error

Definition at line 84 of file libtc.h.

4.1.2.18 VoidFuncString _tc_function_library::tc_ges5

Definition at line 85 of file libtc.h.

4.1.2.19 VoidFuncNoParams _tc_function_library::tc_get_data

Definition at line 86 of file libtc.h.

4.1.2.20 VoidFuncStringFloatPFloatP _tc_function_library::tc_get_derivatives

Definition at line 87 of file libtc.h.

4.1.2.21 VoidFuncStringStringInt _tc_function_library::tc_get_ges5_parameter

Definition at line 88 of file libtc.h.

4.1.2.22 FloatFuncString _tc_function_library::tc_get_value

Definition at line 89 of file libtc.h.

4.1.2.23 IntFuncNoParams _tc_function_library::tc_init_root

Definition at line 90 of file libtc.h.

4.1.2.24 IntFuncStringString _tc_function_library::tc_init_root3

Definition at line 91 of file libtc.h.

4.1.2.25 VoidFuncStringInt _tc_function_library::tc_list_component

Definition at line 92 of file libtc.h.

4.1.2.26 VoidFuncStringInt _tc_function_library::tc_list_conditions

Definition at line 93 of file libtc.h.

4.1.2.27 VoidFuncStringInt _tc_function_library::tc_list_phase

Definition at line 94 of file libtc.h.

4.1.2.28 VoidFuncStringInt _tc_function_library::tc_list_species

Definition at line 95 of file libtc.h.

4.1.2.29 IntFuncStringIntIntP _tc_function_library::tc_list_symbols

Definition at line 96 of file libtc.h.

4.1.2.30 IntFuncString _tc_function_library::tc_nr_of_constituents_in_phase

Definition at line 97 of file libtc.h.

4.1.2.31 VoidFuncString _tc_function_library::tc_open_database

Definition at line 98 of file libtc.h.

4.1.2.32 IntFuncStringInt _tc_function_library::tc_phase

Definition at line 99 of file libtc.h.

4.1.2.33 IntFuncStringIntPStringIntFloatP _tc_function_library::tc_phase_all_constituents

Definition at line 100 of file libtc.h.

4.1.2.34 IntFuncStringIntPStringIntFloatP _tc_function_library::tc_phase_constituents

Definition at line 101 of file libtc.h.

4.1.2.35 VoidFuncString _tc_function_library::tc_phase_reject

Definition at line 102 of file libtc.h.

4.1.2.36 VoidFuncString _tc_function_library::tc_phase_select

Definition at line 103 of file libtc.h.

4.1.2.37 StringFuncString _tc_function_library::tc_phase_status

Definition at line 104 of file libtc.h.

4.1.2.38 IntFuncStringIntPStringStrLenFloatP _tc_function_library::tc_phase_structure

Definition at line 105 of file libtc.h.

4.1.2.39 VoidFuncString _tc_function_library::tc_poly3

Definition at line 106 of file libtc.h.

4.1.2.40 VoidFuncStringFloatP _tc_function_library::tc_put_sitefractions

Definition at line 107 of file libtc.h.

4.1.2.41 VoidFuncString _tc_function_library::tc_read_poly3_file

Definition at line 108 of file libtc.h.

4.1.2.42 VoidFuncStringIntString _tc_function_library::tc_reject_constituent

Definition at line 109 of file libtc.h.

4.1.2.43 VoidFuncNoParams _tc_function_library::tc_reset_error

Definition at line 110 of file libtc.h.

4.1.2.44 VoidFuncStringIntString _tc_function_library::tc_restore_constituent

Definition at line 111 of file libtc.h.

4.1.2.45 VoidFuncString _tc_function_library::tc_save_poly3_file

Definition at line 112 of file libtc.h.

4.1.2.46 VoidFuncInt _tc_function_library::tc_select_equilibrium

Definition at line 113 of file libtc.h.

4.1.2.47 VoidFuncStringString _tc_function_library::tc_set_component_status

Definition at line 114 of file libtc.h.

4.1.2.48 VoidFuncStringFloat _tc_function_library::tc_set_condition

Definition at line 115 of file libtc.h.

4.1.2.49 VoidFuncInt _tc_function_library::tc_set_license_code

Definition at line 116 of file libtc.h.

4.1.2.50 VoidFuncIntPIntPIntPIntP _tc_function_library::tc_set_minimization_option

Definition at line 117 of file libtc.h.

4.1.2.51 VoidFuncStringFloat _tc_function_library::tc_set_phase_addition

Definition at line 118 of file libtc.h.

4.1.2.52 VoidFuncStringStringFloat _tc_function_library::tc_set_phase_status

Definition at line 119 of file libtc.h.

4.1.2.53 VoidFuncStringFloat _tc_function_library::tc_set_start_value

Definition at line 120 of file libtc.h.

4.1.2.54 StringFuncString _tc_function_library::tc_species_status

Definition at line 121 of file libtc.h.

4.1.2.55 VoidFuncStringInt _tc_function_library::tc_version

Definition at line 122 of file libtc.h.

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

- libtc.h

4.2 tc_components_strings Struct Reference

`#include <tc_data_defs.h>`

Public Attributes

- char component [TC_STRLEN_COMPONENTS]

4.2.1 Detailed Description

Definition at line 111 of file tc_data_defs.h.

4.2.2 Member Data Documentation

4.2.2.1 char tc_components_strings::component[TC_STRLEN_COMPONENTS]

Definition at line 113 of file tc_data_defs.h.

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

- tc_data_defs.h

4.3 tc_conditions_as_arrays_of_strings Struct Reference

`#include <tc_data_defs.h>`

Public Attributes

- char condition [TC_STRLEN_MAX]

4.3.1 Detailed Description

Definition at line 98 of file tc_data_defs.h.

4.3.2 Member Data Documentation

4.3.2.1 char tc_conditions_as_arrays_of_strings::condition[TC_STRLEN_MAX]

Definition at line 100 of file tc_data_defs.h.

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

- tc_data_defs.h

4.4 tc_constituents_strings Struct Reference

`#include <tc_data_defs.h>`

Public Attributes

- char constituent [TC_STRLEN_CONSTITUENTS]

4.4.1 Detailed Description

Definition at line 130 of file tc_data_defs.h.

4.4.2 Member Data Documentation

4.4.2.1 char tc_constituents_strings::constituent[TC_STRLEN_CONSTITUENTS]

Definition at line 132 of file tc_data_defs.h.

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

- tc_data_defs.h

4.5 tc_databases_strings Struct Reference

`#include <tc_data_defs.h>`

Public Attributes

- char database [TC_STRLEN_DATABASE]

4.5.1 Detailed Description

Definition at line 136 of file tc_data_defs.h.

4.5.2 Member Data Documentation

4.5.2.1 char tc_databases_strings::database[TC_STRLEN_DATABASE]

Definition at line 138 of file tc_data_defs.h.

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

- tc_data_defs.h

4.6 tc_elements_strings Struct Reference

`#include <tc_data_defs.h>`

Public Attributes

- char element [TC_STRLEN_ELEMENTS]

4.6.1 Detailed Description

Definition at line 105 of file tc_data_defs.h.

4.6.2 Member Data Documentation

4.6.2.1 char tc_elements_strings::element[TC_STRLEN_ELEMENTS]

Definition at line 107 of file tc_data_defs.h.

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

- tc_data_defs.h

4.7 tc_phases_strings Struct Reference

`#include <tc_data_defs.h>`

Public Attributes

- char phase [TC_STRLEN_PHASES]

4.7.1 Detailed Description

Definition at line 124 of file tc_data_defs.h.

4.7.2 Member Data Documentation

4.7.2.1 char tc_phases_strings::phase[TC_STRLEN_PHASES]

Definition at line 126 of file tc_data_defs.h.

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

- tc_data_defs.h

4.8 tc_reference_strings Struct Reference

`#include <tc_data_defs.h>`

Public Attributes

- char reference [TC_STRLEN_REFERENCE]

4.8.1 Detailed Description

Definition at line 142 of file tc_data_defs.h.

4.8.2 Member Data Documentation

4.8.2.1 char tc_reference_strings::reference[TC_STRLEN_REFERENCE]

Definition at line 144 of file tc_data_defs.h.

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

- tc_data_defs.h

4.9 tc_species_strings Struct Reference

```
#include <tc_data_defs.h>
```

Public Attributes

- char specie [TC_STRLEN_SPECIES]

4.9.1 Detailed Description

Definition at line 117 of file tc_data_defs.h.

4.9.2 Member Data Documentation

4.9.2.1 char tc_species_strings::specie[TC_STRLEN_SPECIES]

Definition at line 119 of file tc_data_defs.h.

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

- tc_data_defs.h

Chapter 5

File Documentation

5.1 example1.c File Reference

```
#include <stdio.h>
#include <string.h>
#include "tcapi.h"
#include "tcutils.h"
```

Include dependency graph for example1.c:

5.2 example2.c File Reference

```
#include <stdio.h>
#include <string.h>
#include "tcapi.h"
#include "tcutils.h"
```

Include dependency graph for example2.c:

Functions

- int main (int argc, char **argv)

5.2.1 Function Documentation

5.2.1.1 int main (int *argc,* char ** *argv*)

Definition at line 14 of file example2.c.

5.3 example3.c File Reference

```
#include <stdio.h>
#include <string.h>
#include "tcapi.h"
#include "tcutils.h"
```

Include dependency graph for example3.c:

Typedefs

- typedef char str8[8]
- typedef str8 strvect[100]
- typedef TC_INT ivect[50]
- typedef TC_FLOAT rvect[50]

Functions

- int main (int argc, char **argv)

5.3.1 Typedef Documentation

5.3.1.1 typedef TC_INT ivect[50]

Definition at line 16 of file example3.c.

5.3.1.2 typedef TC_FLOAT rvect[50]

Definition at line 17 of file example3.c.

5.3.1.3 typedef char str8[8]

Definition at line 14 of file example3.c.

5.3.1.4 typedef str8 strvect[100]

Definition at line 15 of file example3.c.

5.3.2 Function Documentation

5.3.2.1 int main (int *argc,* char ** *argv*)

Definition at line 19 of file example3.c.

5.4 libtc.c File Reference

```
#include "libtc.h"
```

Include dependency graph for libtc.c:

Functions

- void * tcloadfunc (TCHANDLE tcHandle, char *function_name)
- int importFunctions (TCHANDLE tcHandle, tc_function_library *tc, char *message)

5.4.1 Function Documentation

5.4.1.1 int importFunctions (TCHANDLE *tcHandle,* tc_function_library * *tc,* char * *message*)

Definition at line 23 of file libtc.c.

5.4.1.2 void* tcloadfunc (TCHANDLE *tcHandle,* char * *function_name*)

Definition at line 11 of file libtc.c.

5.5 libtc.h File Reference

```
#include "tc_data_defs.h"
#include <dlfcn.h>
#include <errno.h>
#include <string.h>
```

Include dependency graph for libtc.h: This graph shows which files directly or indirectly include this file:

Classes

- struct _tc_function_library

Typedefs

- typedef void(* VoidFuncNoParams) ()
- typedef void(* VoidFuncString) (TC_STRING)
- typedef void(* VoidFuncInt) (TC_INT)
- typedef void(* VoidFuncStringString) (TC_STRING, TC_STRING)
- typedef void(* VoidFuncStringFloat) (TC_STRING, TC_FLOAT)
- typedef void(* VoidFuncStringFloatP) (TC_STRING, TC_FLOAT *)
- typedef void(* VoidFuncStringInt) (TC_STRING, TC_INT)
- typedef void(* VoidFuncStringStringInt) (TC_STRING, TC_STRING, TC_INT)
- typedef void(* VoidFuncStringStringFloat) (TC_STRING, TC_STRING, TC_FLOAT)
- typedef void(* VoidFuncStringIntInt) (TC_STRING, TC_INT, TC_INT)
- typedef void(* VoidFuncStringFloatPFloatP) (TC_STRING, TC_FLOAT *, TC_FLOAT *)
- typedef void(* VoidFuncStringIntString) (TC_STRING, TC_INT, TC_STRING)
- typedef void(* VoidFuncIntPIntPIntPIntP) (TC_INT *, TC_INT *, TC_INT *, TC_INT *)
- typedef void(* VoidFuncStringStringIntIntFloatString) (TC_STRING, TC_STRING, TC_INT, TC_INT, TC_FLOAT, TC_STRING)
- typedef TC_INT(* IntFuncNoParams) ()
- typedef TC_INT(* IntFuncString) (TC_STRING)
- typedef TC_INT(* IntFuncStringString) (TC_STRING, TC_STRING)
- typedef TC_INT(* IntFuncStringInt) (TC_STRING, TC_INT)
- typedef TC_INT(* IntFuncStringIntIntP) (TC_STRING, TC_INT, TC_INT *)
- typedef TC_INT(* IntFuncStringIntPStringStrLenFloatP) (TC_STRING, TC_INT *, TC_STRING, TC_STRING_LENGTH, TC_FLOAT *)
- typedef TC_INT(* IntFuncStringIntPStringIntFloatP) (TC_STRING, TC_INT *, TC_STRING, TC_INT, TC_FLOAT *)
- typedef TC_FLOAT(* FloatFuncString) (TC_STRING)
- typedef TC_STRING(* StringFuncString) (TC_STRING)
- typedef TC_BOOL(* BoolFuncIntPStringInt) (TC_INT *, TC_STRING, TC_INT)
- typedef TC_BOOL(* BoolFuncStringStringStrLen) (TC_STRING, TC_STRING, TC_STRING_LENGTH)
- typedef struct _tc_function_library tc_function_library

Functions

- int importFunctions (TCHANDLE tcHandle, tc_function_library *tc, char *message)

5.5.1 Typedef Documentation

5.5.1.1 typedef TC_BOOL(* BoolFuncIntPStringInt) (TC_INT *, TC_STRING, TC_INT)

Definition at line 55 of file libtc.h.

5.5.1.2 typedef TC_BOOL(* BoolFuncStringStringStrLen) (TC_STRING, TC_STRING, TC_STRING_LENGTH)

Definition at line 56 of file libtc.h.

5.5.1.3 typedef TC_FLOAT(* FloatFuncString) (TC_STRING)

Definition at line 51 of file libtc.h.

5.5.1.4 typedef TC_INT(* IntFuncNoParams) ()

Definition at line 43 of file libtc.h.

5.5.1.5 typedef TC_INT(* IntFuncString) (TC_STRING)

Definition at line 44 of file libtc.h.

5.5.1.6 typedef TC_INT(* IntFuncStringInt) (TC_STRING, TC_INT)

Definition at line 46 of file libtc.h.

5.5.1.7 typedef TC_INT(* IntFuncStringIntIntP) (TC_STRING, TC_INT, TC_INT *)

Definition at line 47 of file libtc.h.

5.5.1.8 typedef TC_INT(* IntFuncStringIntPStringIntFloatP) (TC_STRING, TC_INT *, TC_STRING, TC_INT, TC_FLOAT *)

Definition at line 49 of file libtc.h.

5.5.1.9 typedef TC_INT(* IntFuncStringIntPStringStrLenFloatP) (TC_STRING, TC_INT *, TC_STRING, TC_STRING_LENGTH, TC_FLOAT *)

Definition at line 48 of file libtc.h.

5.5.1.10 typedef TC_INT(* IntFuncStringString) (TC_STRING, TC_STRING)

Definition at line 45 of file libtc.h.

5.5.1.11 typedef TC_STRING(* StringFuncString) (TC_STRING)

Definition at line 53 of file libtc.h.

5.5.1.12 typedef struct _tc_function_library tc_function_library

5.5.1.13 typedef void(* VoidFuncInt) (TC_INT)

Definition at line 27 of file libtc.h.

5.5.1.14 typedef void(* VoidFuncIntPIntPIntPIntP) (TC_INT *, TC_INT *, TC_INT *, TC_INT *)

Definition at line 40 of file libtc.h.

5.5.1.15 typedef void(* VoidFuncNoParams) ()

Definition at line 24 of file libtc.h.

5.5.1.16 typedef void(* VoidFuncString) (TC_STRING)

Definition at line 26 of file libtc.h.

5.5.1.17 typedef void(* VoidFuncStringFloat) (TC_STRING, TC_FLOAT)

Definition at line 30 of file libtc.h.

5.5.1.18 typedef void(* VoidFuncStringFloatP) (TC_STRING, TC_FLOAT *)

Definition at line 31 of file libtc.h.

5.5.1.19 typedef void(* VoidFuncStringFloatPFloatP) (TC_STRING, TC_FLOAT *, TC_FLOAT *)

Definition at line 37 of file libtc.h.

5.5.1.20 typedef void(* VoidFuncStringInt) (TC_STRING, TC_INT)

Definition at line 32 of file libtc.h.

5.5.1.21 typedef void(* VoidFuncStringIntInt) (TC_STRING, TC_INT, TC_INT)

Definition at line 36 of file libtc.h.

5.5.1.22 typedef void(* VoidFuncStringIntString) (TC_STRING, TC_INT, TC_STRING)

Definition at line 38 of file libtc.h.

5.5.1.23 typedef void(* VoidFuncStringString) (TC_STRING, TC_STRING)

Definition at line 29 of file libtc.h.

5.5.1.24 typedef void(* VoidFuncStringStringFloat) (TC_STRING, TC_STRING, TC_FLOAT)

Definition at line 35 of file libtc.h.

5.5.1.25 typedef void(* VoidFuncStringStringInt) (TC_STRING, TC_STRING, TC_INT)

Definition at line 34 of file libtc.h.

5.5.1.26 typedef void(* VoidFuncStringStringIntIntFloatString) (TC_STRING, TC_STRING, TC_INT, TC_INT, TC_FLOAT, TC_STRING)

Definition at line 41 of file libtc.h.

5.5.2 Function Documentation

5.5.2.1 int importFunctions (TCHANDLE *tcHandle,* tc_function_library * *tc,* char * *message*)

Definition at line 23 of file libtc.c.

5.6 ReadMe.txt File Reference

5.7 ReadMe.txt File Reference

5.8 tc_data_defs.h File Reference

This graph shows which files directly or indirectly include this file:

Classes

- struct tc_conditions_as_arrays_of_strings
- struct tc_elements_strings
- struct tc_components_strings
- struct tc_species_strings
- struct tc_phases_strings
- struct tc_constituents_strings
- struct tc_databases_strings
- struct tc_reference_strings

Macros

- #define TCHANDLE void*
- #define TC_NWSG 4000000
- #define TC_NWSE 500000
- #define TC_STRLEN_SPECIES 25
- #define TC_STRLEN_PHASES 25
- #define TC_STRLEN_ELEMENTS 3
- #define TC_STRLEN_COMPONENTS 25
- #define TC_STRLEN_CONSTITUENTS 25
- #define TC_STRLEN_DATABASE 9
- #define TC_STRLEN_STOICHIOMETRY 81
- #define TC_STRLEN_MAX 256
- #define TC_STRLEN_PATH_MAX 512
- #define TC_STRLEN_REFERENCE 1024
- #define TC_MAX_NR_OF_ELEMENTS 40

- #define TC_MAX_NR_OF_SPECIES 5000
- #define TC_MAX_NR_OF_SUBLATTICES 10
- #define TC_MAX_NR_OF_CONSTITUENTS 200
- #define TC_MAX_NR_OF_CONST_PER_SUBLATTICE 200
- #define TC_MAX_NR_OF_CONST_PER_SUBLATTICE_IN_IDEAL_GAS 5000
- #define TC_MAX_NR_OF_DATABASES 130
- #define TC_MAX_NR_OF_AXES 5
- #define TC_MAX_NR_OF_PHASES 4000 /∗ check ITDBPX in tdbmax.inc ∗/
- #define TC_EPS 1.00E-8
- #define TC_VARS
- #define true 1
- #define false 0
- #define DllExport __declspec(dllexport)
- #define TCFuncExport extern DllExport
- #define INTEGER_FUNC extern TC_INT
- #define INTEGER_FUNC_WIN INTEGER_FUNC
- #define INTEGER_FUNC_GNU INTEGER_FUNC
- #define VOID_FUNC_WIN extern void
- #define BOOL_FUNC_WIN extern TC_BOOL
- #define FLOAT_FUNC_WIN extern TC_FLOAT

Typedefs

- typedef long TC_INT
- typedef long pointer
- typedef double TC_FLOAT
- typedef TC_INT TC_BOOL
- typedef char ∗ TC_STRING
- typedef long TC_STRING_LENGTH
- typedef struct tc_conditions_as_arrays_of_strings tc_conditions_as_arrays_of_strings
- typedef struct tc_elements_strings tc_elements_strings
- typedef struct tc_components_strings tc_components_strings
- typedef struct tc_species_strings tc_species_strings
- typedef struct tc_phases_strings tc_phases_strings
- typedef struct tc_constituents_strings tc_constituents_strings
- typedef struct tc_databases_strings tc_databases_strings
- typedef struct tc_reference_strings tc_reference_strings
- typedef TC_INT TC_IARR[4]
- typedef char TC_LABEL_STRING[127]

5.8.1 Macro Definition Documentation

5.8.1.1 #define BOOL_FUNC_WIN extern TC_BOOL

Definition at line 196 of file tc_data_defs.h.

5.8.1.2 #define DllExport __declspec(dllexport)

Definition at line 170 of file tc_data_defs.h.

5.8.1.3 #define false 0

Definition at line 161 of file tc_data_defs.h.

5.8.1.4 #define FLOAT_FUNC_WIN extern TC_FLOAT

Definition at line 197 of file tc_data_defs.h.

5.8.1.5 #define INTEGER_FUNC extern TC_INT

Definition at line 191 of file tc_data_defs.h.

5.8.1.6 #define INTEGER_FUNC_GNU INTEGER_FUNC

Definition at line 194 of file tc_data_defs.h.

5.8.1.7 #define INTEGER_FUNC_WIN INTEGER_FUNC

Definition at line 193 of file tc_data_defs.h.

5.8.1.8 #define TC_EPS 1.00E-8

Definition at line 77 of file tc_data_defs.h.

5.8.1.9 #define TC_MAX_NR_OF_AXES 5

Definition at line 74 of file tc_data_defs.h.

5.8.1.10 #define TC_MAX_NR_OF_CONST_PER_SUBLATTICE 200

Definition at line 71 of file tc_data_defs.h.

5.8.1.11 #define TC_MAX_NR_OF_CONST_PER_SUBLATTICE_IN_IDEAL_GAS 5000

Definition at line 72 of file tc_data_defs.h.

5.8.1.12 #define TC_MAX_NR_OF_CONSTITUENTS 200

Definition at line 70 of file tc_data_defs.h.

5.8.1.13 #define TC_MAX_NR_OF_DATABASES 130

Definition at line 73 of file tc_data_defs.h.

5.8.1.14 #define TC_MAX_NR_OF_ELEMENTS 40

Definition at line 67 of file tc_data_defs.h.

5.8.1.15 #define TC_MAX_NR_OF_PHASES 4000 /∗ check ITDBPX in tdbmax.inc ∗/

Definition at line 76 of file tc_data_defs.h.

5.8.1.16 #define TC_MAX_NR_OF_SPECIES 5000

Definition at line 68 of file tc_data_defs.h.

5.8.1.17 #define TC_MAX_NR_OF_SUBLATTICES 10

Definition at line 69 of file tc_data_defs.h.

5.8.1.18 #define TC_NWSE 500000

Definition at line 54 of file tc_data_defs.h.

5.8.1.19 #define TC_NWSG 4000000

Definition at line 53 of file tc_data_defs.h.

5.8.1.20 #define TC_STRLEN_COMPONENTS 25

Definition at line 59 of file tc_data_defs.h.

5.8.1.21 #define TC_STRLEN_CONSTITUENTS 25

Definition at line 60 of file tc_data_defs.h.

5.8.1.22 #define TC_STRLEN_DATABASE 9

Definition at line 61 of file tc_data_defs.h.

5.8.1.23 #define TC_STRLEN_ELEMENTS 3

Definition at line 58 of file tc_data_defs.h.

5.8.1.24 #define TC_STRLEN_MAX 256

Definition at line 63 of file tc_data_defs.h.

5.8.1.25 #define TC_STRLEN_PATH_MAX 512

Definition at line 64 of file tc_data_defs.h.

5.8.1.26 #define TC_STRLEN_PHASES 25

Definition at line 57 of file tc_data_defs.h.

5.8.1.27 #define TC_STRLEN_REFERENCE 1024

Definition at line 65 of file tc_data_defs.h.

5.8.1.28 #define TC_STRLEN_SPECIES 25

Definition at line 56 of file tc_data_defs.h.

5.8.1.29 #define TC_STRLEN_STOICHIOMETRY 81

Definition at line 62 of file tc_data_defs.h.

5.8.1.30 #define TC_VARS

Definition at line 151 of file tc_data_defs.h.

5.8.1.31 #define TCFuncExport extern DllExport

Definition at line 182 of file tc_data_defs.h.

5.8.1.32 #define TCHANDLE void*

Definition at line 36 of file tc_data_defs.h.

5.8.1.33 #define true 1

Definition at line 157 of file tc_data_defs.h.

5.8.1.34 #define VOID_FUNC_WIN extern void

Definition at line 195 of file tc_data_defs.h.

5.8.2 Typedef Documentation

5.8.2.1 typedef long pointer

Definition at line 20 of file tc_data_defs.h.

5.8.2.2 typedef TC_INT TC_BOOL

Definition at line 41 of file tc_data_defs.h.

5.8.2.3 typedef struct tc_components_strings tc_components_strings

Definition at line 110 of file tc_data_defs.h.

5.8.2.4 typedef struct tc_conditions_as_arrays_of_strings tc_conditions_as_arrays_of_strings

Definition at line 97 of file tc_data_defs.h.

5.8.2.5 typedef struct tc_constituents_strings tc_constituents_strings

Definition at line 129 of file tc_data_defs.h.

5.8.2.6 typedef struct tc_databases_strings tc_databases_strings

Definition at line 135 of file tc_data_defs.h.

5.8.2.7 typedef struct tc_elements_strings tc_elements_strings

Definition at line 104 of file tc_data_defs.h.

5.8.2.8 typedef double TC_FLOAT

Definition at line 31 of file tc_data_defs.h.

5.8.2.9 typedef TC_INT TC_IARR[4]

Definition at line 147 of file tc_data_defs.h.

5.8.2.10 typedef long TC_INT

Definition at line 19 of file tc_data_defs.h.

5.8.2.11 typedef char TC_LABEL_STRING[127]

Definition at line 148 of file tc_data_defs.h.

5.8.2.12 typedef struct tc_phases_strings tc_phases_strings

Definition at line 123 of file tc_data_defs.h.

5.8.2.13 typedef struct tc_reference_strings tc_reference_strings

Definition at line 141 of file tc_data_defs.h.

5.8.2.14 typedef struct tc_species_strings tc_species_strings

Definition at line 116 of file tc_data_defs.h.

5.8.2.15 typedef char* TC_STRING

Definition at line 42 of file tc_data_defs.h.

5.8.2.16 typedef long TC_STRING_LENGTH

Definition at line 50 of file tc_data_defs.h.

5.9 tcapi.h File Reference

```
#include "tc_data_defs.h"
```

Include dependency graph for tcapi.h: This graph shows which files directly or indirectly include this file:

Functions

- void tc_append_database (TC_STRING name)
- TC_BOOL tc_check_license (TC_STRING name, TC_STRING message, TC_STRING_LENGTH strlen_↩ message)
- TC_STRING tc_component_status (TC_STRING component_name)
- void tc_compute_equilibrium ()
- void tc_create_new_equilibrium (TC_INT equilibrium)
- TC_INT tc_database (TC_STRING datan, TC_INT linelen)
- void tc_define_components (TC_STRING component_name, TC_INT strlen, TC_INT number_of_↩ components)
- TC_INT tc_degrees_of_freedom ()
- void tc_deinit ()
- void tc_delete_condition (TC_STRING condition)
- void tc_delete_symbol (TC_STRING symbol)
- TC_INT tc_element (TC_STRING elements, TC_INT linelen)
- void tc_element_reject (TC_STRING element_name)
- void tc_element_select (TC_STRING element_name)
- void tc_enter_ges5_parameter (TC_STRING parameter, TC_STRING expression)
- void tc_enter_symbol (TC_STRING symbol, TC_STRING type, TC_INT argument_type, TC_INT integer_↩ argument, TC_FLOAT double_argument, TC_STRING string_argument)
- TC_BOOL tc_error (TC_INT *error_number, TC_STRING message, TC_INT strlen)
- void tc_ges5 (TC_STRING command)
- void tc_get_data ()
- void tc_get_derivatives (TC_STRING phase_name, TC_FLOAT *arr1, TC_FLOAT *arr2)
- void tc_get_ges5_parameter (TC_STRING parameter, TC_STRING expression, TC_INT strlenExpression)
- TC_FLOAT tc_get_value (TC_STRING symbol)
- TC_INT tc_init_root ()
- TC_INT tc_init_root3 (TC_STRING tmppath, TC_STRING tcpath)
- TC_INT tc_list_component (TC_STRING component_name, TC_INT strlen)
- TC_INT tc_list_conditions (TC_STRING conditions, TC_INT strlen)
- TC_INT tc_list_phase (TC_STRING phase_name, TC_INT strlen)
- TC_INT tc_list_species (TC_STRING species_name, TC_INT strlen)
- TC_INT tc_list_symbols (TC_STRING symbols, TC_INT strlen, TC_INT *type)
- TC_INT tc_nr_of_constituents_in_phase (TC_STRING phase_name)
- void tc_open_database (TC_STRING name)
- TC_INT tc_phase (TC_STRING phases, TC_INT linelen)
- TC_INT tc_phase_all_constituents (TC_STRING phase_name, TC_INT *constituent_array, TC_STRIN↩ G element_array, TC_INT strLenElem, TC_FLOAT *number_of_sites)
- TC_INT tc_phase_constituents (TC_STRING phase_name, TC_INT *constituent_array, TC_STRIN↩ G element_array, TC_INT strLenElem, TC_FLOAT *number_of_sites)
- void tc_phase_reject (TC_STRING phase_name)
- void tc_phase_select (TC_STRING phase_name)
- TC_STRING tc_phase_status (TC_STRING phase_name)
- TC_INT tc_phase_structure (TC_STRING phase_name, TC_INT *constituent_array, TC_STRING species↩ _array, TC_STRING_LENGTH strLenSpecies, TC_FLOAT *number_of_sites)
- void tc_poly3 (TC_STRING command)
- void tc_put_sitefractions (TC_STRING phase_name, TC_FLOAT *sfarr)
- void tc_read_poly3_file (TC_STRING filename)
- void tc_reject_constituent (TC_STRING phase_name, TC_INT sublattice, TC_STRING constituent)
- void tc_reset_error ()
- void tc_restore_constituent (TC_STRING phase_name, TC_INT sublattice, TC_STRING constituent)
- void tc_save_poly3_file (TC_STRING filename)
- void tc_select_equilibrium (TC_INT equilibrium)
- void tc_set_component_status (TC_STRING component_name, TC_STRING status)

- void tc_set_condition (TC_STRING condition, TC_FLOAT value)
- void tc_set_license_code (TC_INT code)
- void tc_set_minimization_option (TC_INT *global_flag, TC_INT *max_gridpoints, TC_INT *frequency, TC_INT *mesh_flag)
- void tc_set_phase_addition (TC_STRING phase_name, TC_FLOAT addition)
- void tc_set_phase_status (TC_STRING phase_name, TC_STRING status, TC_FLOAT value)
- void tc_set_start_value (TC_STRING state_variable, TC_FLOAT starting_value)
- TC_STRING tc_species_status (TC_STRING species_name)
- void tc_version (TC_STRING str, TC_INT str_len)

5.9.1 Function Documentation

5.9.1.1 void tc_append_database (TC_STRING *name*)

Appends the named database

5.9.1.2 TC_BOOL tc_check_license (TC_STRING *name*, TC_STRING *message*, TC_STRING_LENGTH *strlen_message*)

Returns true if the checked license is available and valid, currently accepts: TC_DLL, TC_GUI and TC_TC4U

5.9.1.3 TC_STRING tc_component_status (TC_STRING *component_name*)

Returns the status of "component_name" where status may be one of "ENTERED" or "SUSPENDED"

5.9.1.4 void tc_compute_equilibrium ( )

Computes the equilibrium in POLY-3 using the currently set conditions

5.9.1.5 void tc_create_new_equilibrium (TC_INT *equilibrium*)

Creates a new equilibrium in POLY-3 with number "equilibrium number".

5.9.1.6 TC_INT tc_database (TC_STRING *datan*, TC_INT *linelen*)

Returns the number of databases in the system and their names in "datan"

5.9.1.7 void tc_define_components (TC_STRING *component_name*, TC_INT *strlen*, TC_INT *number_of_components*)

Redefines the components in the system to the components in "component_name".

5.9.1.8 TC_INT tc_degrees_of_freedom ( )

Returns the degrees of freedom in the system. This must be zero in order to perform an equilibrium calculation.

5.9.1.9 void tc_deinit ( )

5.9.1.10 void tc_delete_condition (TC_STRING *condition*)

Deletes the condition for the expression in "condition".

5.9.1.11 void tc_delete_symbol (TC_STRING *symbol*)

Deletes a symbol in the system.

5.9.1.12 TC_INT tc_element (TC_STRING *elements,* TC_INT *linelen*)

Returns the number of elements in the database and their names in "elements"

5.9.1.13 void tc_element_reject (TC_STRING *element_name*)

Rejects "element_name" in the currently selected database.

5.9.1.14 void tc_element_select (TC_STRING *element_name*)

Selects "element_name" in the currently selected database.

5.9.1.15 void tc_enter_ges5_parameter (TC_STRING *parameter,* TC_STRING *expression*)

Enters a parameter expression

5.9.1.16 void tc_enter_symbol (TC_STRING *symbol,* TC_STRING *type,* TC_INT *argument_type,* TC_INT *integer_argument,* TC_FLOAT *double_argument,* TC_STRING *string_argument*)

Enters a symbol in the system, the symbol type may be one of "CONSTANT", "VARIABLE", "FUNCTION" or "T↩ ABLE", "argument_type" defines which of the following arguments will be used, 1 indicates the integer argument, 2 the double argument and 3 the string argument.

5.9.1.17 TC_BOOL tc_error (TC_INT * *error_number,* TC_STRING *message,* TC_INT *strlen*)

Returns true if an error has been set, returning the error number in "error_number" and its corresponding message in "message"

5.9.1.18 void tc_ges5 (TC_STRING *command*)

Sends a command to the GES5 module as defined in the argument "command"

5.9.1.19 void tc_get_data ( )

Executes the database command "GET_DATA"

5.9.1.20 void tc_get_derivatives (TC_STRING *phase_name,* TC_FLOAT * *arr1,* TC_FLOAT * *arr2*)

Returns Gm and the first derivatives with respect to site-fractions in "arr1" and the second derivatives in "arr2" as GM.Y1.Y1, GM.Y1.Y2, GM.Y2.Y2, GM.Y1.Y3, GM.Y2.Y3, GM.YN.YN

5.9.1.21 void tc_get_ges5_parameter (TC_STRING *parameter,* TC_STRING *expression,* TC_INT *strlenExpression*)

Retrieves the expression of a parameter name

5.9.1.22 TC_FLOAT tc_get_value (TC_STRING *symbol*)

Retrieves the symbol or state variable value from the POLY-3 module.

5.9.1.23 TC_INT tc_init_root ( )

Initializes the Thermo-Calc system. This function (or tc_init_root3) must be called prior to anything else.

5.9.1.24 TC_INT tc_init_root3 (TC_STRING *tmppath,* TC_STRING *tcpath*)

Initializes the Thermo-Calc system. This function (or tc_init_root) must be called prior to anything else. tmppath Path to directory for log file tcpath Path to Thermo-Calc installation (Used to find databases)

5.9.1.25 TC_INT tc_list_component (TC_STRING *component_name,* TC_INT *strlen*)

Returns the number of components in the system and their names in "component_name".

5.9.1.26 TC_INT tc_list_conditions (TC_STRING *conditions,* TC_INT *strlen*)

Returns the number of conditions set and their values in "conditions"

5.9.1.27 TC_INT tc_list_phase (TC_STRING *phase_name,* TC_INT *strlen*)

Returns the number of phases in the system and their names in "phase_name".

5.9.1.28 TC_INT tc_list_species (TC_STRING *species_name,* TC_INT *strlen*)

Returns the number of species in the system and their names in "species_name".

5.9.1.29 TC_INT tc_list_symbols (TC_STRING *symbols,* TC_INT *strlen,* TC_INT * *type*)

Returns the number of defined symbols in the system with their expression and value in "symbols" and their corresponding type in "type of symbol", where the type may be one of 1="CONSTANT", 2="VARIABLE" 3="FUNCTION" 4="TABLE"

5.9.1.30 TC_INT tc_nr_of_constituents_in_phase (TC_STRING *phase_name*)

5.9.1.31 void tc_open_database (TC_STRING *name*)

Opens the named database "name_of_database"

5.9.1.32 TC_INT tc_phase (TC_STRING *phases,* TC_INT *linelen*)

Returns the number of phases in the system with the selected elements. NOTE: the routine returns the number of all available phases.

5.9.1.33 TC_INT tc_phase_all_constituents (TC_STRING *phase_name,* TC_INT * *constituent_array,* TC_STRING *element_array,* TC_INT *strLenElem,* TC_FLOAT * *number_of_sites*)

Returns the number of sublattices in the phase (including phases with the status SUSPENDED), the number of constituents on each sublattice in "constituent_array", the name of the selected species on each sublattice one after each other in "element_array" and the "number_of_sites" on each sublattice.

5.9.1.34 TC_INT tc_phase_constituents (TC_STRING *phase_name,* TC_INT * *constituent_array,* TC_STRING *element_array,* TC_INT *strLenElem,* TC_FLOAT * *number_of_sites*)

Returns the number of sublattices in the phase, the number of constituents on each sublattice in "constituent_array", the name of the selected species on each sublattice one after each other in "element_array" and the "number of sites" on each sublattice.

5.9.1.35 void tc_phase_reject (TC_STRING *phase_name*)

Rejects the phase in "phase_name"

5.9.1.36 void tc_phase_select (TC_STRING *phase_name*)

Selects the phase in "phase_name"

5.9.1.37 TC_STRING tc_phase_status (TC_STRING *phase_name*)

Returns the status of "phase_name" where status may be one of "FIXED", "SUSPENDED" or "ENTERED"

5.9.1.38 TC_INT tc_phase_structure (TC_STRING *phase_name,* TC_INT * *constituent_array,* TC_STRING *species_array,* TC_STRING_LENGTH *strLenSpecies,* TC_FLOAT * *number_of_sites*)

Returns the number of sublattices in the phase, the number of constituents on each sublattice in "constituent_↩ array", the name of the species on each sublattice one after each other in "species_array" and the number of sites in "number of sites".

5.9.1.39 void tc_poly3 (TC_STRING *command*)

Sends a command to POLY-3 module as defined in the argument "command"

5.9.1.40 void tc_put_sitefractions (TC_STRING *phase_name,* TC_FLOAT * *sfarr*)

5.9.1.41 void tc_read_poly3_file (TC_STRING *filename*)

Loads the workspace from file "filename" in POLY-3.

5.9.1.42 void tc_reject_constituent (TC_STRING *phase_name,* TC_INT *sublattice,* TC_STRING *constituent*)

Rejects the constituent "constituent" on sublattice "sublattiice" from phase "phase_name".

5.9.1.43 void tc_reset_error ( )

Resets the error if an error has been set

5.9.1.44 void tc_restore_constituent (TC_STRING *phase_name,* TC_INT *sublattice,* TC_STRING *constituent*)

Restores the constituent "constituent" on sublattice "sublattice" from phase "phase_name".

5.9.1.45 void tc_save_poly3_file (TC_STRING *filename*)

Stores/overwrites the current workspace in POLY-3 on the file "filename".

5.9.1.46 void tc_select_equilibrium (TC_INT *equilibrium*)

Selects an equilibrium in POLY-3 with number "equilibrium".

5.9.1.47 void tc_set_component_status (TC_STRING *component_name,* TC_STRING *status*)

Sets the status of "component_name" to "status" to one of "ENTERED" or "SUSPENDED"

5.9.1.48 void tc_set_condition (TC_STRING *condition,* TC_FLOAT *value*)

Sets a condition for the expression in "condition" to value in "value".

5.9.1.49 void tc_set_license_code (TC_INT *code*)

5.9.1.50 void tc_set_minimization_option (TC_INT * *global_flag,* TC_INT * *max_gridpoints,* TC_INT * *frequency,* TC_INT * *mesh_flag*)

Sets parameters for global minimization

5.9.1.51 void tc_set_phase_addition (TC_STRING *phase_name,* TC_FLOAT *addition*)

Sets the addition "addition" to the Gibbs

5.9.1.52 void tc_set_phase_status (TC_STRING *phase_name,* TC_STRING *status,* TC_FLOAT *value*)

Sets the status of "phase_name" to "status" to one of "FIXED", "SUSPENDED", DORMANT" or "ENTERED".

5.9.1.53 void tc_set_start_value (TC_STRING *state_variable,* TC_FLOAT *starting_value*)

Sets a starting value for the "state_variable" to "start_value".

5.9.1.54 TC_STRING tc_species_status (TC_STRING *species_name*)

Returns the status of "species_name" where status may be one of "ENTERED" or "SUSPENDED"

5.9.1.55 void tc_version (TC_STRING *str,* TC_INT *str_len*)

Returns the version of Thermo-Calc in "str"

5.10 tcExamples.c File Reference

```
#include <stdio.h>
#include "tcExamples.h"
#include "tcutils.h"
```

Include dependency graph for tcExamples.c:

Typedefs

- typedef char str8[8]
- typedef str8 strvect[100]
- typedef TC_INT ivect[50]
- typedef TC_FLOAT rvect[50]

Functions

- void example1 (tc_function_library *tc)
- void example2 (tc_function_library *tc)
- void example3 (tc_function_library *tc)

5.10.1 Typedef Documentation

5.10.1.1 typedef TC_INT ivect[50]

Definition at line 288 of file tcExamples.c.

5.10.1.2 typedef TC_FLOAT rvect[50]

Definition at line 289 of file tcExamples.c.

5.10.1.3 typedef char str8[8]

Definition at line 286 of file tcExamples.c.

5.10.1.4 typedef str8 strvect[100]

Definition at line 287 of file tcExamples.c.

5.10.2 Function Documentation

5.10.2.1 void example1 (tc_function_library * *tc*)

tcExamples.h

Definition at line 9 of file tcExamples.c.

5.10.2.2 void example2 (tc_function_library * *tc*)

Definition at line 201 of file tcExamples.c.

5.10.2.3 void example3 (tc_function_library * *tc*)

Definition at line 291 of file tcExamples.c.

5.11 tcExamples.h File Reference

```
#include "libtc.h"
```
Include dependency graph for tcExamples.h: This graph shows which files directly or indirectly include this file:

Functions

- void example1 (tc_function_library *tc)
- void example2 (tc_function_library *tc)
- void example3 (tc_function_library *tc)

5.11.1 Function Documentation

5.11.1.1 void example1 (tc_function_library * *tc*)

tcExamples.h

Definition at line 9 of file tcExamples.c.

5.11.1.2 void example2 (tc_function_library * *tc*)

Definition at line 201 of file tcExamples.c.

5.11.1.3 void example3 (tc_function_library * *tc*)

Definition at line 291 of file tcExamples.c.

5.12 tcMain.c File Reference

```
#include <stdio.h>
#include <string.h>
#include "tcExamples.h"
#include "tcutils.h"
```
Include dependency graph for tcMain.c:

Macros

- #define TC_API_LIBRARY_NAME "libtcapi-linux-ia32-gfortran-7.5-SNAPSHOT.so"

Functions

- TCHANDLE loadTCLibraryInCurrentDir ()
- int importLibThermoCalc (tc_function_library *tc, char *message)
- int main (int argc, char *argv[ ])

5.12.1 Macro Definition Documentation

5.12.1.1 #define TC_API_LIBRARY_NAME "libtcapi-linux-ia32-gfortran-7.5-SNAPSHOT.so"

Definition at line 28 of file tcMain.c.

5.12.2 Function Documentation

5.12.2.1 int importLibThermoCalc (tc_function_library * tc, char * message)

Definition at line 66 of file tcMain.c.

5.12.2.2 TCHANDLE loadTCLibraryInCurrentDir ( )

Definition at line 38 of file tcMain.c.

5.12.2.3 int main (int argc, char * argv[ ])

Definition at line 100 of file tcMain.c.

5.13 tcutils.c File Reference

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "tcutils.h"
```

Include dependency graph for tcutils.c:

Functions

- void getThermoCalcEnvironmentPath (char *pathBuffer)
- void getTempEnvironmentPath (char *pathBuffer)

5.13.1 Function Documentation

5.13.1.1 void getTempEnvironmentPath (char * pathBuffer)

Get path to temp directory If it can't find it - default to current working directory

Definition at line 29 of file tcutils.c.

5.13.1.2 void getThermoCalcEnvironmentPath (char * pathBuffer)

Get path from the values of Thermo-Calc environment variables TC61_HOME TCPATH Older versions and fallback

Definition at line 12 of file tcutils.c.

5.14 tcutils.h File Reference

```
#include <unistd.h>
```

Include dependency graph for tcutils.h: This graph shows which files directly or indirectly include this file:

Macros

- #define getCurrentWorkingDir getcwd
- #define TCPATH "TCPATH"
- #define TC61_HOME "TC61_HOME"
- #define TEMP "TMPDIR"
- #define SLASH "/"

Functions

- void getThermoCalcEnvironmentPath (char *pathBuffer)
- void getTempEnvironmentPath (char *pathBuffer)

5.14.1 Macro Definition Documentation

5.14.1.1 #define getCurrentWorkingDir getcwd

Definition at line 6 of file tcutils.h.

5.14.1.2 #define SLASH "/"

Definition at line 21 of file tcutils.h.

5.14.1.3 #define TC61_HOME "TC61_HOME"

Definition at line 10 of file tcutils.h.

5.14.1.4 #define TCPATH "TCPATH"

Definition at line 9 of file tcutils.h.

5.14.1.5 #define TEMP "TMPDIR"

Definition at line 15 of file tcutils.h.

5.14.2 Function Documentation

5.14.2.1 void getTempEnvironmentPath (char * *pathBuffer*)

Get path to temp directory If it can't find it - default to current working directory

Definition at line 29 of file tcutils.c.

5.14.2.2 void getThermoCalcEnvironmentPath (char * *pathBuffer*)

Get path from the values of Thermo-Calc environment variables TC61_HOME TCPATH Older versions and fallback

Definition at line 12 of file tcutils.c.

Index

_tc_function_library, 9
  tc_append_database, 10
  tc_check_license, 10
  tc_component_status, 10
  tc_compute_equilibrium, 10
  tc_create_new_equilibrium, 10
  tc_database, 10
  tc_define_components, 11
  tc_degrees_of_freedom, 11
  tc_deinit, 11
  tc_delete_condition, 11
  tc_delete_symbol, 11
  tc_element, 11
  tc_element_reject, 11
  tc_element_select, 11
  tc_enter_ges5_parameter, 11
  tc_enter_symbol, 11
  tc_error, 11
  tc_ges5, 11
  tc_get_data, 12
  tc_get_derivatives, 12
  tc_get_ges5_parameter, 12
  tc_get_value, 12
  tc_init_root, 12
  tc_init_root3, 12
  tc_list_component, 12
  tc_list_conditions, 12
  tc_list_phase, 12
  tc_list_species, 12
  tc_list_symbols, 12
  tc_nr_of_constituents_in_phase, 12
  tc_open_database, 13
  tc_phase, 13
  tc_phase_all_constituents, 13
  tc_phase_constituents, 13
  tc_phase_reject, 13
  tc_phase_select, 13
  tc_phase_status, 13
  tc_phase_structure, 13
  tc_poly3, 13
  tc_put_sitefractions, 13
  tc_read_poly3_file, 13
  tc_reject_constituent, 13
  tc_reset_error, 14
  tc_restore_constituent, 14
  tc_save_poly3_file, 14
  tc_select_equilibrium, 14
  tc_set_component_status, 14
  tc_set_condition, 14
  tc_set_license_code, 14
  tc_set_minimization_option, 14
  tc_set_phase_addition, 14
  tc_set_phase_status, 14
  tc_set_start_value, 14
  tc_species_status, 14
  tc_version, 15

BOOL_FUNC_WIN
  tc_data_defs.h, 25
BoolFuncIntPStringInt
  libtc.h, 22
BoolFuncStringStringStrLen
  libtc.h, 22

component
  tc_components_strings, 15
condition
  tc_conditions_as_arrays_of_strings, 16
constituent
  tc_constituents_strings, 16

database
  tc_databases_strings, 16
DllExport
  tc_data_defs.h, 25

element
  tc_elements_strings, 17
example1
  tcExamples.c, 36
  tcExamples.h, 37
example1.c, 19
example2
  tcExamples.c, 36
  tcExamples.h, 37
example2.c, 19
  main, 19
example3
  tcExamples.c, 36
  tcExamples.h, 37
example3.c, 19
  ivect, 20
  main, 20
  rvect, 20
  str8, 20
  strvect, 20

FLOAT_FUNC_WIN
  tc_data_defs.h, 25
false

tc_data_defs.h, 25
FloatFuncString
libtc.h, 22

getCurrentWorkingDir
tcutils.h, 39
getTempEnvironmentPath
tcutils.c, 38
tcutils.h, 39
getThermoCalcEnvironmentPath
tcutils.c, 38
tcutils.h, 39

INTEGER_FUNC
tc_data_defs.h, 26
INTEGER_FUNC_GNU
tc_data_defs.h, 26
INTEGER_FUNC_WIN
tc_data_defs.h, 26
importFunctions
libtc.c, 20
libtc.h, 24
importLibThermoCalc
tcMain.c, 38
IntFuncNoParams
libtc.h, 22
IntFuncString
libtc.h, 22
IntFuncStringInt
libtc.h, 22
IntFuncStringIntIntP
libtc.h, 22
IntFuncStringIntPStringIntFloatP
libtc.h, 22
IntFuncStringIntPStringStrLenFloatP
libtc.h, 22
IntFuncStringString
libtc.h, 22
ivect
example3.c, 20
tcExamples.c, 36

libtc.c, 20
importFunctions, 20
tcloadfunc, 20
libtc.h, 21
BoolFuncIntPStringInt, 22
BoolFuncStringStringStrLen, 22
FloatFuncString, 22
importFunctions, 24
IntFuncNoParams, 22
IntFuncString, 22
IntFuncStringInt, 22
IntFuncStringIntIntP, 22
IntFuncStringIntPStringIntFloatP, 22
IntFuncStringIntPStringStrLenFloatP, 22
IntFuncStringString, 22
StringFuncString, 22
tc_function_library, 22
VoidFuncInt, 23
VoidFuncIntPIntPIntPIntP, 23
VoidFuncNoParams, 23
VoidFuncString, 23
VoidFuncStringFloat, 23
VoidFuncStringFloatP, 23
VoidFuncStringFloatPFloatP, 23
VoidFuncStringInt, 23
VoidFuncStringIntInt, 23
VoidFuncStringIntString, 23
VoidFuncStringString, 23
VoidFuncStringStringFloat, 23
VoidFuncStringStringInt, 23
VoidFuncStringStringIntIntFloatString, 24
loadTCLibraryInCurrentDir
tcMain.c, 38

main
example2.c, 19
example3.c, 20
tcMain.c, 38

phase
tc_phases_strings, 17
pointer
tc_data_defs.h, 28

ReadMe.txt, 24
reference
tc_reference_strings, 18
rvect
example3.c, 20
tcExamples.c, 36

SLASH
tcutils.h, 39
specie
tc_species_strings, 18
str8
example3.c, 20
tcExamples.c, 36
StringFuncString
libtc.h, 22
strvect
example3.c, 20
tcExamples.c, 36

TC61_HOME
tcutils.h, 39
TC_API_LIBRARY_NAME
tcMain.c, 38
TC_BOOL
tc_data_defs.h, 28
TC_EPS
tc_data_defs.h, 26
TC_FLOAT
tc_data_defs.h, 29
TC_IARR
tc_data_defs.h, 29

TC_INT
tc_data_defs.h, 29
TC_LABEL_STRING
tc_data_defs.h, 29
TC_MAX_NR_OF_AXES
tc_data_defs.h, 26
TC_MAX_NR_OF_CONST_PER_SUBLATTICE
tc_data_defs.h, 26
TC_MAX_NR_OF_CONST_PER_SUBLATTICE_IN_↩
IDEAL_GAS
tc_data_defs.h, 26
TC_MAX_NR_OF_CONSTITUENTS
tc_data_defs.h, 26
TC_MAX_NR_OF_DATABASES
tc_data_defs.h, 26
TC_MAX_NR_OF_ELEMENTS
tc_data_defs.h, 26
TC_MAX_NR_OF_PHASES
tc_data_defs.h, 26
TC_MAX_NR_OF_SPECIES
tc_data_defs.h, 26
TC_MAX_NR_OF_SUBLATTICES
tc_data_defs.h, 27
TC_NWSE
tc_data_defs.h, 27
TC_NWSG
tc_data_defs.h, 27
TC_STRING
tc_data_defs.h, 29
TC_STRING_LENGTH
tc_data_defs.h, 29
TC_STRLEN_COMPONENTS
tc_data_defs.h, 27
TC_STRLEN_CONSTITUENTS
tc_data_defs.h, 27
TC_STRLEN_DATABASE
tc_data_defs.h, 27
TC_STRLEN_ELEMENTS
tc_data_defs.h, 27
TC_STRLEN_MAX
tc_data_defs.h, 27
TC_STRLEN_PATH_MAX
tc_data_defs.h, 27
TC_STRLEN_PHASES
tc_data_defs.h, 27
TC_STRLEN_REFERENCE
tc_data_defs.h, 27
TC_STRLEN_SPECIES
tc_data_defs.h, 27
TC_STRLEN_STOICHIOMETRY
tc_data_defs.h, 28
TC_VARS
tc_data_defs.h, 28
TCFuncExport
tc_data_defs.h, 28
TCHANDLE
tc_data_defs.h, 28
TCPATH
tcutils.h, 39
TEMP
tcutils.h, 39
tc_append_database
_tc_function_library, 10
tcapi.h, 31
tc_check_license
_tc_function_library, 10
tcapi.h, 31
tc_component_status
_tc_function_library, 10
tcapi.h, 31
tc_components_strings, 15
component, 15
tc_data_defs.h, 28
tc_compute_equilibrium
_tc_function_library, 10
tcapi.h, 31
tc_conditions_as_arrays_of_strings, 15
condition, 16
tc_data_defs.h, 28
tc_constituents_strings, 16
constituent, 16
tc_data_defs.h, 28
tc_create_new_equilibrium
_tc_function_library, 10
tcapi.h, 31
tc_data_defs.h, 24
BOOL_FUNC_WIN, 25
DllExport, 25
FLOAT_FUNC_WIN, 25
false, 25
INTEGER_FUNC, 26
INTEGER_FUNC_GNU, 26
INTEGER_FUNC_WIN, 26
pointer, 28
TC_BOOL, 28
TC_EPS, 26
TC_FLOAT, 29
TC_IARR, 29
TC_INT, 29
TC_LABEL_STRING, 29
TC_MAX_NR_OF_AXES, 26
TC_MAX_NR_OF_CONST_PER_SUBLATTICE,
26
TC_MAX_NR_OF_CONST_PER_SUBLATTICE↩
_IN_IDEAL_GAS, 26
TC_MAX_NR_OF_CONSTITUENTS, 26
TC_MAX_NR_OF_DATABASES, 26
TC_MAX_NR_OF_ELEMENTS, 26
TC_MAX_NR_OF_PHASES, 26
TC_MAX_NR_OF_SPECIES, 26
TC_MAX_NR_OF_SUBLATTICES, 27
TC_NWSE, 27
TC_NWSG, 27
TC_STRING, 29
TC_STRING_LENGTH, 29
TC_STRLEN_COMPONENTS, 27

TC_STRLEN_CONSTITUENTS, 27
TC_STRLEN_DATABASE, 27
TC_STRLEN_ELEMENTS, 27
TC_STRLEN_MAX, 27
TC_STRLEN_PATH_MAX, 27
TC_STRLEN_PHASES, 27
TC_STRLEN_REFERENCE, 27
TC_STRLEN_SPECIES, 27
TC_STRLEN_STOICHIOMETRY, 28
TC_VARS, 28
TCFuncExport, 28
TCHANDLE, 28
tc_components_strings, 28
tc_conditions_as_arrays_of_strings, 28
tc_constituents_strings, 28
tc_databases_strings, 28
tc_elements_strings, 29
tc_phases_strings, 29
tc_reference_strings, 29
tc_species_strings, 29
true, 28
VOID_FUNC_WIN, 28
tc_database
_tc_function_library, 10
tcapi.h, 31
tc_databases_strings, 16
database, 16
tc_data_defs.h, 28
tc_define_components
_tc_function_library, 11
tcapi.h, 31
tc_degrees_of_freedom
_tc_function_library, 11
tcapi.h, 31
tc_deinit
_tc_function_library, 11
tcapi.h, 31
tc_delete_condition
_tc_function_library, 11
tcapi.h, 31
tc_delete_symbol
_tc_function_library, 11
tcapi.h, 31
tc_element
_tc_function_library, 11
tcapi.h, 32
tc_element_reject
_tc_function_library, 11
tcapi.h, 32
tc_element_select
_tc_function_library, 11
tcapi.h, 32
tc_elements_strings, 17
element, 17
tc_data_defs.h, 29
tc_enter_ges5_parameter
_tc_function_library, 11
tcapi.h, 32
tc_enter_symbol
_tc_function_library, 11
tcapi.h, 32
tc_error
_tc_function_library, 11
tcapi.h, 32
tc_function_library
libtc.h, 22
tc_ges5
_tc_function_library, 11
tcapi.h, 32
tc_get_data
_tc_function_library, 12
tcapi.h, 32
tc_get_derivatives
_tc_function_library, 12
tcapi.h, 32
tc_get_ges5_parameter
_tc_function_library, 12
tcapi.h, 32
tc_get_value
_tc_function_library, 12
tcapi.h, 32
tc_init_root
_tc_function_library, 12
tcapi.h, 33
tc_init_root3
_tc_function_library, 12
tcapi.h, 33
tc_list_component
_tc_function_library, 12
tcapi.h, 33
tc_list_conditions
_tc_function_library, 12
tcapi.h, 33
tc_list_phase
_tc_function_library, 12
tcapi.h, 33
tc_list_species
_tc_function_library, 12
tcapi.h, 33
tc_list_symbols
_tc_function_library, 12
tcapi.h, 33
tc_nr_of_constituents_in_phase
_tc_function_library, 12
tcapi.h, 33
tc_open_database
_tc_function_library, 13
tcapi.h, 33
tc_phase
_tc_function_library, 13
tcapi.h, 33
tc_phase_all_constituents
_tc_function_library, 13
tcapi.h, 33
tc_phase_constituents
_tc_function_library, 13

tcapi.h, 34
tc_phase_reject
_tc_function_library, 13
tcapi.h, 34
tc_phase_select
_tc_function_library, 13
tcapi.h, 34
tc_phase_status
_tc_function_library, 13
tcapi.h, 34
tc_phase_structure
_tc_function_library, 13
tcapi.h, 34
tc_phases_strings, 17
phase, 17
tc_data_defs.h, 29
tc_poly3
_tc_function_library, 13
tcapi.h, 34
tc_put_sitefractions
_tc_function_library, 13
tcapi.h, 34
tc_read_poly3_file
_tc_function_library, 13
tcapi.h, 34
tc_reference_strings, 17
reference, 18
tc_data_defs.h, 29
tc_reject_constituent
_tc_function_library, 13
tcapi.h, 34
tc_reset_error
_tc_function_library, 14
tcapi.h, 34
tc_restore_constituent
_tc_function_library, 14
tcapi.h, 34
tc_save_poly3_file
_tc_function_library, 14
tcapi.h, 35
tc_select_equilibrium
_tc_function_library, 14
tcapi.h, 35
tc_set_component_status
_tc_function_library, 14
tcapi.h, 35
tc_set_condition
_tc_function_library, 14
tcapi.h, 35
tc_set_license_code
_tc_function_library, 14
tcapi.h, 35
tc_set_minimization_option
_tc_function_library, 14
tcapi.h, 35
tc_set_phase_addition
_tc_function_library, 14
tcapi.h, 35
tc_set_phase_status
_tc_function_library, 14
tcapi.h, 35
tc_set_start_value
_tc_function_library, 14
tcapi.h, 35
tc_species_status
_tc_function_library, 14
tcapi.h, 35
tc_species_strings, 18
specie, 18
tc_data_defs.h, 29
tc_version
_tc_function_library, 15
tcapi.h, 35
tcExamples.c, 36
example1, 36
example2, 36
example3, 36
ivect, 36
rvect, 36
str8, 36
strvect, 36
tcExamples.h, 37
example1, 37
example2, 37
example3, 37
tcMain.c, 37
importLibThermoCalc, 38
loadTCLibraryInCurrentDir, 38
main, 38
TC_API_LIBRARY_NAME, 38
tcapi.h, 29
tc_append_database, 31
tc_check_license, 31
tc_component_status, 31
tc_compute_equilibrium, 31
tc_create_new_equilibrium, 31
tc_database, 31
tc_define_components, 31
tc_degrees_of_freedom, 31
tc_deinit, 31
tc_delete_condition, 31
tc_delete_symbol, 31
tc_element, 32
tc_element_reject, 32
tc_element_select, 32
tc_enter_ges5_parameter, 32
tc_enter_symbol, 32
tc_error, 32
tc_ges5, 32
tc_get_data, 32
tc_get_derivatives, 32
tc_get_ges5_parameter, 32
tc_get_value, 32
tc_init_root, 33
tc_init_root3, 33
tc_list_component, 33

tc_list_conditions, 33
tc_list_phase, 33
tc_list_species, 33
tc_list_symbols, 33
tc_nr_of_constituents_in_phase, 33
tc_open_database, 33
tc_phase, 33
tc_phase_all_constituents, 33
tc_phase_constituents, 34
tc_phase_reject, 34
tc_phase_select, 34
tc_phase_status, 34
tc_phase_structure, 34
tc_poly3, 34
tc_put_sitefractions, 34
tc_read_poly3_file, 34
tc_reject_constituent, 34
tc_reset_error, 34
tc_restore_constituent, 34
tc_save_poly3_file, 35
tc_select_equilibrium, 35
tc_set_component_status, 35
tc_set_condition, 35
tc_set_license_code, 35
tc_set_minimization_option, 35
tc_set_phase_addition, 35
tc_set_phase_status, 35
tc_set_start_value, 35
tc_species_status, 35
tc_version, 35
tcloadfunc
libtc.c, 20
tcutils.c, 38
getTempEnvironmentPath, 38
getThermoCalcEnvironmentPath, 38
tcutils.h, 38
getCurrentWorkingDir, 39
getTempEnvironmentPath, 39
getThermoCalcEnvironmentPath, 39
SLASH, 39
TC61_HOME, 39
TCPATH, 39
TEMP, 39
true
tc_data_defs.h, 28

VOID_FUNC_WIN
tc_data_defs.h, 28
VoidFuncInt
libtc.h, 23
VoidFuncIntPIntPIntPIntP
libtc.h, 23
VoidFuncNoParams
libtc.h, 23
VoidFuncString
libtc.h, 23
VoidFuncStringFloat
libtc.h, 23
VoidFuncStringFloatP
libtc.h, 23
VoidFuncStringFloatPFloatP
libtc.h, 23
VoidFuncStringInt
libtc.h, 23
VoidFuncStringIntInt
libtc.h, 23
VoidFuncStringIntString
libtc.h, 23
VoidFuncStringString
libtc.h, 23
VoidFuncStringStringFloat
libtc.h, 23
VoidFuncStringStringInt
libtc.h, 23
VoidFuncStringStringIntIntFloatString
libtc.h, 24