

TQ-Interface

SDK Programmer's Guide

Thermo-Calc 2022b



Copyright 2022 Thermo-Calc Software AB. All rights reserved.

Information in this document is subject to change without notice. The software or database described in this document is furnished under a license agreement or nondisclosure agreement. The software or database may be used or copied only in accordance with the terms of those agreements. You can [read more on our website](#).

Thermo-Calc Software AB

Råsundavägen 18, SE-169 67 Solna, Sweden

+46 8 545 959 30

www.thermocalc.com

Contents

| | |
|---|-----------|
| TQ-Interface | 1 |
| The TQ-Interface | 1 |
| <i>Introduction to the TQ-Interface</i> | 2 |
| <i>The Subroutines and Functions</i> | 4 |
| Installing and Using the TQ-Interface | 5 |
| <i>Installing the TQ-Interface and Examples</i> | 6 |
| <i>Using this Guide</i> | 8 |
| <i>Programming Languages</i> | 9 |
| <i>Basic Concepts</i> | 11 |
| <i>Naming Components, Phases and Constituents</i> | 12 |
| <i>About Adaptive Interpolation Schemes</i> | 14 |
| Initialization Subroutines | 15 |
| <i>Units for TQSSU and TQGSU</i> | 16 |
| <i>Legal Input/Output Options for TQSIO and TQGIO</i> | 17 |
| <i>TQINI3</i> | 18 |
| <i>TQINI</i> | 19 |
| <i>TQSIO</i> | 20 |
| <i>TQGIO</i> | 21 |
| <i>TQRFIL</i> | 22 |
| <i>TQSSU</i> | 23 |
| <i>TQGSU</i> | 24 |
| <i>TQSAME</i> | 25 |

| | |
|---|-----------|
| <i>TQGVER</i> | 26 |
| System Data Manipulation Subroutines | 27 |
| <i>Legal Component Status</i> | 28 |
| <i>Legal Phase Status</i> | 29 |
| <i>TQGNC</i> | 30 |
| <i>TQSCOM</i> | 31 |
| <i>TQGCOM</i> | 33 |
| <i>TQGSCI</i> | 34 |
| <i>TQGNP</i> | 35 |
| <i>TQGPN</i> | 36 |
| <i>TQGPI</i> | 37 |
| <i>TQGPCN</i> | 38 |
| <i>TQGPCI</i> | 39 |
| <i>TQGCCF</i> | 40 |
| <i>TQGPCS</i> | 41 |
| <i>TQGNPC</i> | 42 |
| <i>TQCSSC</i> | 43 |
| <i>TQGSSC</i> | 44 |
| <i>TQCSP</i> | 45 |
| <i>TQGSP</i> | 46 |
| <i>TQSETR</i> | 47 |
| <i>TQPACS</i> | 48 |
| <i>TQSGA</i> | 49 |
| <i>TQGGA</i> | 50 |

Condition, Stream and Segment Subroutines 51

| | |
|---|----|
| <i>Possible State Variables to Set Conditions in TQSETC</i> | 52 |
| TQSETC | 53 |
| TQREMC | 55 |
| TQSCURC | 56 |
| TQREMAC | 57 |
| TQRESTC | 58 |
| TQCSTM | 59 |
| TQSSC | 60 |
| TQSSIC | 61 |
| TQDSTM | 63 |
| TQNSEG | 64 |
| TQSSEG | 65 |

Calculations and Results Subroutines 66

| | |
|--|----|
| <i>State Variables Available for TQGETV and TQGET1</i> | 67 |
| TQCE | 70 |
| TQCEG | 72 |
| TQGETV and TQGET1 | 73 |
| TQGMU | 76 |
| TQGGM | 77 |
| TQGPD | 78 |
| TQGDF2 | 80 |
| TQGSE | 82 |

Miscellaneous Subroutines 84

| | |
|-------------------------|-----|
| TQLS | 85 |
| TQLC | 86 |
| TQLE | 87 |
| TQFASV | 88 |
| TQKEEP_CS_NUMBERS | 89 |
| TQSDMC | 90 |
| TQSSPC | 91 |
| TQSSV | 92 |
| TQPINI | 93 |
| TQSNL | 94 |
| TQSMNG | 96 |
| TQSECO | 97 |
| ST1ERR | 98 |
| ST2ERR | 99 |
| SG1ERR or TQG1ERR | 100 |
| SG2ERR or TQG2ERR | 101 |
| SG3ERR or TQG3ERR | 102 |
| RESERR or TQRSERR | 103 |
| TQSP3F | 104 |

Extra Subroutines-Phase Properties 105

| | |
|-------------------------------|-----|
| TQGMA, TQGMB and TQGM C | 106 |
| TQGMDY | 107 |
| TQGMOB | 108 |
| TQSTP | 109 |

| | | | |
|---|------------|--|------------|
| <i>TQSYF</i> | 110 | <i>TQIPS_READ_IPS_DATA_FROM_FILE</i> ... | 139 |
| <i>TQGSSPI</i> | 111 | <i>TQIPS_GET_MEMORY_USAGE</i> | 140 |
| <i>TQCMOBA and TQCMOBB</i> | 112 | Composition Set Reordering | |
| <i>TQDGY</i> | 113 | Routines | 141 |
| <i>TQGPHP</i> | 114 | <i>TQROINIT</i> | 142 |
| <i>TQX2Y</i> | 115 | <i>TQSETRX</i> | 143 |
| <i>TQGM</i> | 116 | <i>TQORDER</i> | 144 |
| Database Subroutines | 118 | <i>TQLROX</i> | 145 |
| <i>TQGDBN</i> | 119 | Compiler Settings | 146 |
| <i>TQOPDB</i> | 120 | <i>Compiling FORTRAN Code</i> | 147 |
| <i>TQLIDE</i> | 121 | <i>Compiling C code</i> | 149 |
| <i>TQAPDB</i> | 122 | | |
| <i>TQDEFEL</i> | 123 | | |
| <i>TQREJEL</i> | 124 | | |
| <i>TQREJPH</i> | 125 | | |
| <i>TQRESPH</i> | 126 | | |
| <i>TQLISPH</i> | 127 | | |
| <i>TQLISSF</i> | 128 | | |
| <i>TQGDAT</i> | 129 | | |
| <i>TQREJSY</i> | 130 | | |
| Adaptive Interpolation Schemes .. | 131 | | |
| <i>TQIPS_INIT_TOP</i> | 132 | | |
| <i>TQIPS_INIT_BRANCH</i> | 133 | | |
| <i>TQIPS_INIT_FUNCTION</i> | 136 | | |
| <i>TQIPS_GET_VALUE</i> | 137 | | |
| <i>TQIPS_WRITE_IPS_DATA_TO_FILE</i> | 138 | | |

The TQ-Interface

In this section:

Introduction to the TQ-Interface 2

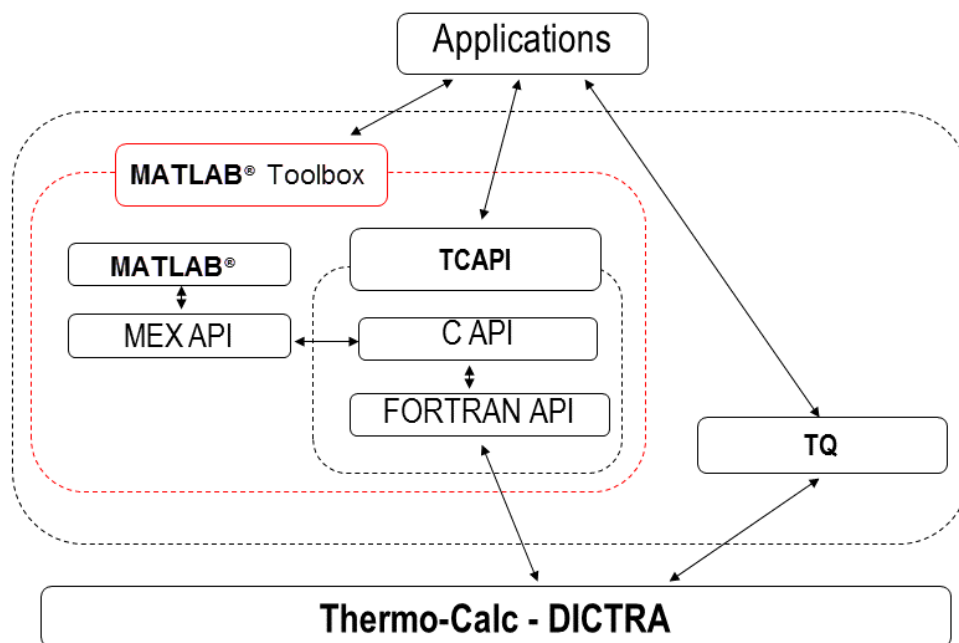
The Subroutines and Functions 4

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.



["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 15](#). 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 27](#). 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 51](#). 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 66](#). For example, calculate equilibriums, get molar Gibbs energy values, and calculate interfacial energy between a matrix phase and a precipitate phase.
5. ["Miscellaneous Subroutines" on page 84](#). 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 105](#). 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 118](#) For example, get lists of database names, reject a selected element and get data from the selected database.
- ["Adaptive Interpolation Schemes" on page 131](#). 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 141](#). 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 | 6 |
| Using this Guide | 8 |
| Programming Languages | 9 |
| Basic Concepts | 11 |
| Naming Components, Phases and Constituents | 12 |
| About Adaptive Interpolation Schemes | 14 |

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, Metallurgical Thermochemistry, 1979, Page 211. |
| TQEX07 | This sample program calculates the A3 temperature of a steel and determines the influence of |

| Example Name | Description |
|--------------|--|
| | 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 minimization 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. |
| |  <p>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.</p> |
| MPExample1 | This is an MPI (Message Passing Interface) example that calculates the Gibbs energy for a composition grid C with a density of "npoin" at 1273K in the Mn-Ni-Fe system. |
| 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.

 ["Programming Languages" on the next page](#)

Note the following conventions to distinguish between the programming languages.

- Routines starting with **TQXXX**, for example, *TQGDAT*, are in the Fortran interface
- Routines starting with **tq_**xxxx, for example *tq_gdat*, 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.

TQGDAT  Subroutine name

| Fortran | TQGDAT(IWSG, IWSE) | Subroutine name (commands) |
|-------------|---|-------------------------------|
| C-interface | tq_gdat(TC_INT* iwsg, TC_INT* iwse) | C-procedure (definitions) |
| Full name: | Get Data. | |
| Purpose: | Get data for the defined system from the chosen database. | |
| Arguments | Description and purpose: E.g. GET = get data | |
| Name | Type | Value set on call or returned |
| IWSG | Integer array | Workspace |
| IWSE | Integer array | Workspace |

The commands match the string order in the first two rows and based on if you are using Fortran or C-interface

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 **tgroot.h** which should be included in the procedures using the library calls in C.
- The **tgroot.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 146).

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. |
| 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:

- "TQGC0M" on page 33 returns the total number of components and all component names
- "TQGSCI" on page 34 returns the index of one component name
- "TQSC0M" on page 31 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 43, thus leaving gaps in the component list because suspending one component does not change the sequential numbering. The logical function "TQGSSC" on page 44 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 35](#) for the total number of phases
- "TQGPN" on [page 36](#) for the name of a phase
- "TQGPI" on [page 37](#) for the name of its index
- "TQGNPC" on [page 42](#) for the number of phase constituents



Phases can be suspended or set dormant by "TQCSP" on [page 45](#), thus leaving gaps in the list because suspending one phase does not change the sequential numbering. The logical function "TQGSP" on [page 46](#) 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 42](#) for the number of constituents of a phase
- "TQGPCN" on [page 38](#) (or its index "TQGPCI" on [page 39](#)) for the name of a phase constituent
- "TQGPCS" on [page 41](#) 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 and Höglund (2015): "A Scheme for More Efficient Usage of CALPHAD Data in Simulations', *Calphad*, 50, 1–5.

Initialization Subroutines

| Purpose | Subroutine |
|--|---------------------|
| Initialize TQ-Interface with user-specified database and temporary directories | "TQINI3" on page 18 |
| Initialize TQ-Interface | "TQINI" on page 19 |
| Set input/output option | "TQSIO" on page 20 |
| Get input/output option | "TQGIO" on page 21 |
| Read thermodynamic data file | "TQRFIL" on page 22 |
| Set unit for a system quantity | "TQSSU" on page 23 |
| Get unit for a system quantity | "TQGSU" on page 24 |
| Check if the system is the same | "TQSAME" on page 25 |
| Retrieve information about the TQ-library | "TQGVER" on page 26 |



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 |
| | 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 TQSI0 and TQG10

| 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 | <div>Path to the directory holding the data directory, which in turn contains the databases.</div> <div> See the examples collection for a default value for this parameter.</div> |
| temp_path | Character*256 | <div>Path to the directory for temporary and log file output. This directory has to be writable by the user who runs the application.</div> <div> See the examples collection for a default value for this parameter.</div> |
| 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. |
| 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 17 |
| 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. | |
| 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 17. |
| 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: | <p>The default set of components is supplied by the thermodynamic input file. The thermodynamic data file should contain at least the following information.</p> <ul style="list-style-type: none">• 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 <p>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.</p> | |
| | <div><div></div><div><p>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.</p></div></div> | |
| 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 16 . | |
| 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 |
| 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 16. | |
| 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. | |
| 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 page 30 |
| Set system components | "TQSCOM" on page 31 |
| Get system components | "TQGCOM" on page 33 |
| Get system component index | "TQGSCI" on page 34 |
| Get number of phases | "TQGNP" on page 35 |
| Get phase name | "TQGPN" on page 36 |
| Get phase index | "TQGPI" on page 37 |
| Get phase constituent name | "TQGPCN" on page 38 |
| Get phase constituent index | "TQGPCI" on page 39 |
| Get component chemical formula | "TQGCCF" on page 40 |
| Get phase constituent stoichiometry | "TQGPCS" on page 41 |
| Get number of phase constituents | "TQGNPC" on page 42 |
| Change the status of components, phases, and component reference states | |
| Change status of system component | "TQCSSC" on page 43 |
| Get status of system component | "TQGSSC" on page 44* |
| Change status of phase | "TQCSP" on page 45 |
| Get status of phase | "TQGSP" on page 46* |
| Set reference state | "TQSETR" on page 47 |
| Add a composition set to a phase | "TQPACS" on page 48 |
| | * Logical function |
| Contributions to the Gibbs energy of a phase | |
| Set Gibbs energy addition | "TQSGA" on page 49 |
| Get Gibbs energy addition | "TQGGA" on page 50 |

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. |
| SPECIAL | The specified component(s) are not included in summations for mole or mass fractions. It only works for component(s). |

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 43. | |
| Comments: | <div><ul style="list-style-type: none">• 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 34.</div> <div><h3>Example</h3><p>For example, to transform from the components A, B, C to A₂B, B₄CC₂ use the following values of STOI:</p><pre>A2B2.0 1.0 0.0 B4C 0.0 4.0 1.0 C2 0.0 0.0 2.0</pre></div> | |
| 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* iwsq,TC_INT* iwse); | |
| Full name: | Get System Component. | |
| Purpose: | Get components of a system | |
| Comments: | <p>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.</p> <p> "TQGSCI" on the next page</p> | |
| 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* np,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 |
| 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 the application 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* iws,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 . | |
| 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 |
| 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. | |
| | <div><div></div><div>The dimension of ELNAM and STOI is NEL (number of elements in the component).</div></div> | |
| 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* iws,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 33. | |
| 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 78. | |
| 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, SUSPENDED and SPECIAL | |
| |  "Legal Component Status" on page 28 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. |
| 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 28.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: | <p>The legal values for STATUS are given in "Legal Phase Status" on page 29.</p> <p>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.</p> <p>For FIXED phase the exact amount of the phase must be given. Note that the amount is in number of mole formula units.</p> <p>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.</p> | |
| 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 mole formula units. |
| 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 29. 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 as mole formula units. |
| 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 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 (J/(mol formula unit)) |
| IWSG | Integer array | Workspace |
| IWSE | Integer array | Workspace |

TQGGA

| | | |
|-------------|---|--|
| Fortran | TQGA (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 (J/(mol formula unit)). |
| 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 53 |
| Remove condition | "TQREMC" on page 55 |
| Save current conditions | "TQSCURC" on page 56 |
| Remove all conditions | "TQREMAC" on page 57 |
| Restore saved conditions | "TQRESTC" on page 58 |
| Create stream | "TQCSTM" on page 59 |
| Set stream constituent amount | "TQSSC" on page 60 |
| Set stream invariant state variable | "TQSSIC" on page 61 |
| Delete stream | "TQDSTM" on page 63 |
| Create new equilibrium segment | "TQNSEG" on page 64 |
| Select equilibrium segment | "TQSSEG" on page 65 |

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 phase constituent |
| V | | | Volume | of the whole system |
| G | | | Gibbs energy | of the whole system |
| 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 |

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.

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.

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 the previous page. 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 the previous page. 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. |
| NUMCON | Integer | Returned as an identification of the |

| | | |
|--------------------|--|------------|
| 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); | |
| | | 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. | |
| Comments: | "TQSETC" on page 53 and "TQCSTM" on page 59 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 page 58. | |
| 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. | |
| 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 | | |
| 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 22. Note that when several segments are used, an 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* iwsq,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 70 |
| Calculate global equilibrium | "TQCEG" on page 72 |
| Get equilibrium property values (TQGET) and Get one value (TQGET1) | "TQGETV and TQGET1" on page 73 |
| Get chemical potential value. It is a double precision function. | "TQGMU " on page 76 |
| Get molar Gibbs energy value. It is a double precision function. | "TQGGM" on page 77 |
| Get phase data | "TQGPD" on page 78 |
| Get driving force and local equilibrium compositions for ortho- or para-equilibrium phase transformation | "TQGDF2" on page 80 |
| Get interfacial energy between a matrix phase and a precipitate phase | "TQGSE " on page 82 |

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 |
| QF | Yes | | Phase stability function | Negative when phase composition is inside a spinodal, otherwise positive. Can be used to find out if an equilibrium is within the miscibility gap for a solution phase. Cannot be used as a condition. |
| 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 |
| 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 |

| STAVAR | INDEXP | INDEXC | Meaning | Comments |
|---|--------|--------|------------------------|-------------------------------------|
| 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 |
| * 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 | $\text{m}^2\text{mol}^{-1}\text{s}^{-1}$ |
| LOGM(phase,J) | 10log of the mobility coefficient | $\log_{10} (\text{m}^2\text{mol}^{-1}\text{s}^{-1})$ |
| DT(phase,J) | Tracer diffusion coefficient where J=diffusing species | m^2/s |
| LOGDT(phase,J) | 10log of the tracer diffusion coefficient | $\log_{10} (\text{m}^2\text{s}^{-1})$ |
| 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 | $\log_{10} (\text{m}^2\text{s}^{-1})$ |

| STAVAR | Meaning | Unit |
|--------------------|--|-------------------|
| 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 |

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 53. 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 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 TQGP1('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: | <p>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</p> <div> See Example 13.</div> | |
| | <p>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.</p> | |
| 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: | <div><div></div><div>With TQGETV an array of values can be returned; with TQGET1 a single value only.</div></div> | |
| 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 67 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 |

| | | |
|--------------------|--|-----------|
| 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); | |
| IWSG | Integer array | Workspace |
| 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)
```

Get the derivative X(LIQUID,CR).X(FE). Where the indices describe the property before the "dot".

```
CALL TQGPI(INDEXP, 'LIQUID', IWSG, IWSE)
CALL TQGSCI(INDEXC, 'CR', IWSG, IWSE)
CALL TQGET1('X.X(FE)', INDEXP, INDEXC, VAL, 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 |
| 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: | <p>With this subroutine the application program can determine the structure of the phase and the fraction of the constituents and other things. Note that 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.</p> <p>The constituents of a phase are numbered sequentially from 1 for the first constituent on the first sublattice, to NPCON (See "TQGNPC" on page 42) 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.</p> | |
| 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) |

| 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); | |
| 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: | <p>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.</p> <p> See Example 11.</p> | |
| Comments: | For ortho-equilibrium calculations, XPREC can be inputs or outputs, depending 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. However, ± 3 can be used only 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 |

| | | |
|--------------------|---|---|
| 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* iws,TC_INT* iwse); | |
| 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 of atoms if MODE = ± 1 , ± 2 and J/mole of substitutional atoms if MODE = ± 3 |
| 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 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, U, VOLM,VOLP, IWSG, IWSE) | |
| C-interface | tq_gse (TC_INT imatr,TC_INT iprec,TC_INT imc,TC_FLOAT temp,TC_FLOAT* u,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: | <div><div><p>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 <i>Interfacial Energy</i> model included with the Property Model Calculator and Precipitation Module (TC-PRISMA).</p><p>For systems with interstitial elements note the following:</p><ul style="list-style-type: none">The composition array must contain so-called <i>u-fractions</i>.<p> Search for "u-fraction variable" in this PDF or press F1 to search the help.</p><ul style="list-style-type: none">The molar volumes of the matrix and precipitate should be with respect to substitutional elements. This can be achieved by first setting the component status to 'SPECIAL' for the interstitial elements with TQCSSC, and then retrieve the correct molar volume with TQGET1('VM',...).</div><div><div></div><div>An equilibrium calculation is not required prior to using this function.</div></div></div> | |
| Arguments | | |
| 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 |
| U | Double precision array | Set overall alloy composition in u-fraction |
| VOLM | Double precision | Set molar volume of matrix phase with |

| | | |
|---------------------|---|---|
| Fortran | TQGSE (IMATR, IPREC, IMC,TEMP, U, VOLM,VOLP, IWSG, IWSE) | |
| C-interface | tq_gse (TC_INT imatr,TC_INT iprec,TC_INT imc,TC_FLOAT temp,TC_FLOAT* u,TC_FLOAT volm,TC_FLOAT volp,TC_INT* iwsg,TC_INT* iwse); | |
| | | respect to substitutional elements |
| VOLP | Double precision | Set molar volume of precipitate phase with respect to substitutional elements |
| IWSG | Integer array | Workspace |
| IWSE | Integer array | Workspace |
| Return Value | | |
| TQGSE | Double precision | The interfacial energy in J/m ² |

Miscellaneous Subroutines

| Purpose | Subroutine |
|--------------------------------------|-----------------------------------|
| List status | "TQLS" on the next page |
| List conditions | "TQLC" on page 86 |
| List equilibrium | "TQLE" on page 87 |
| Force automatic start values | "TQFASV" on page 88 |
| Keep composition set numbers | "TQKEEP_CS_NUMBERS" on page 89 |
| Set default major constituent | "TQSDMC" on page 90 |
| Set start phase constitution | "TQSSPC" on page 91 |
| Set start value of a state variable | "TQSSV" on page 92 |
| Reinitiate the calculation workspace | "TQPINI" on page 93 |
| Set numerical limits | "TQSNL" on page 94 |
| Set maximum number of grid points | "TQSMNG" on page 96 |
| Set equilibrium calculation options | "TQSECO" on page 97 |
| Set error code and give message | "ST1ERR" on page 98 |
| Set error code | "ST2ERR" on page 99 |
| Get error code and give message | "SG1ERR or TQG1ERR" on page 100 * |
| Get error code | "SG2ERR or TQG2ERR" on page 101 * |
| Get error code and message | "SG3ERR or TQG3ERR" on page 102 * |
| Reset error code and message | "RESERR or TQRSERR" on page 103 |
| Save a POLY-3 file | "TQSP3F" on page 104 |
| | * 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 |

TQKEEP_CS_NUMBERS

| | | |
|-------------|---|--|
| Fortran | TQKEEP_CS_NUMBERS(IWSE, KEEP) | |
| C-interface | tq_keep_cs_numbers(TC_INT* iwse, TC_BOOL* keep); | |
| Full name: | Keep Composition Set Numbers. | |
| Purpose: | To prevent composition set ID-numbers from switching between consecutive equilibrium calculations. | |
| Comments: | This subroutine turns on/off the functionality to keep the composition set numbers from the previous equilibrium calculations. By default the setting is off. Once turned on, it affects all subsequent equilibrium calculations until explicitly turned off again. | |
| Arguments | | |
| Name | Type | Value set on call or returned |
| IWSE | Integer array | Workspace |
| KEEP | Logical | TRUE will turn the functionality on, FALSE off |

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 Gibbs (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. |
| 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 52 |
| 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_STRING adg,TC_INT* iwsq,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. |
| ADG | Character*1 | <p>Approximate driving force for metastable phases. Y is the default. Enter N to change the default as required and based on the options described below.</p> <p>This setting involves the convergence of metastable phases and affects their driving forces. It can also have an effect on when an equilibrium is considered successful.</p> <p>The default is to allow an equilibrium with metastable phases that have not converged, as long as the stable phases have converged. This is efficient but often causes approximate values of the driving forces for the metastable phases.</p> <p>If you change the default, it enforces metastable phases to converge. This gives accurate driving forces for metastable phases as well as stable phases. It can however take a slightly longer time, and if metastable phases do not converge it causes the equilibrium calculation to fail.</p> |
| IWSG | Integer array | Workspace |

| | | |
|-------------|--|-----------|
| Fortran | TQSNL(MAXIT, ACC, YMIN, ADG, IWSG, IWSE) | |
| C-interface | tq_snl(TC_INT maxit,TC_FLOAT acc,TC_FLOAT ymin,TC_STRING adg,TC_INT* iwsg,TC_INT* iwse); | |
| 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 Hessian 0 = 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: | <div>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 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.</div> <div><div></div><div>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.</div></div> | |
| 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 |
| 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: | <p>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.</p> <p><i>Example</i></p> <pre>LOGICAL SG2ERR ... CALL TQCE(' ',IWSG,IWSE) IF(SG2ERR(IERR)) GOTO 900 ... 900 RETURN</pre> | |
| 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. |
| 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 107 |
| Get mobility of a species in a phase. | "TQGMOB" on page 108 |
| Set temperature and pressure for TQGMC, TQGMDY, and TQGMOB. | "TQSTP" on page 109 |
| Set site fractions for TQGMB, TQGMC, TQGMDY, and TQGMOB. | "TQSYF" on page 110 |
| Get index of a system species. | "TQGSSPI" on page 111 |
| Check if Mobility data is available Method A and B | "TQCMOBA and TQCMOBB" on page 112† |
| Get 1st and 2nd partial derivative of Gibbs energy w.r.t. site fractions.* | "TQDGY" on page 113 |
| Get constitutional properties of a phase.* | "TQGPHP" on page 114 |
| Convert mole fraction to site fraction for phases with no internal degree of freedom.* | "TQX2Y" on page 115 |
| Convert 1st partial derivative of Gibbs energy w.r.t. site fractions to that w.r.t. mole fractions.* | "TQGM DX" on page 116 |
| *in SI unit for one mole of formula unit. | † Logical function. |


TQGMA, TQGMB and TQGM C

| | | |
|-------------|---|--|
| 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* iwsq,TC_INT* iwse);tq_gmb(TC_INT indexp,TC_FLOAT* tp,TC_FLOAT* val,TC_INT* iwsq,TC_INT* iwse); tq_gmc(TC_INT indexp,TC_FLOAT* val,TC_INT* iwsq,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: | <div>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.</div> <div> This subroutine is demonstrated in Example 9.</div> | |
| 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, TQDGY, 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: | <p>This subroutine is used before calling TQGMB, TQGMC, TQGMDY, TQDGY, and TQGMOB.</p> <p> See Example 9.</p> | |
| Arguments | | |
| Name | Type | Value set on call or returned |
| 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 108.  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 |


TQDGY

| | | |
|-------------|---|---|
| Fortran | TQDGY(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. | |
| 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=I+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: | <p>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.</p> <p> See Example 10.</p> | |
| 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 $\geq 4 \cdot \text{NCNV}$ |
| WORK | Double precision array | Return values needed in X to Y conversion, array size $\geq (\text{NE}+1) \cdot \text{NCNV}$ |
| 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 input. <div> See Example 10.</div> | |
| 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: | <div>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.</div> <div> For the use of this subroutine together with "TQGPHP" on page 114 and "TQX2Y" on the previous page, see Example 10.</div> | |
| 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 |

| | | |
|--------------------|--|---|
| Fortran | TQGMDX(IP, NE, NCV, NC, IWORK, WORK, YF, VARR, GM, DGDX, XF, IWSG, IWSE) | |
| C-interface | tq_gmdx(TC_INT indexp,TC_INT ne,TC_INT ncv,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); | |
| GM | Double precision | Return Gibbs energy |
| 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" on the next page |
| Open or switch to a database | "TQOPDB" on page 120 |
| List database elements | "TQLIDE" on page 121 |
| Append a database | "TQAPDB" on page 122 |
| Select an element | "TQDEFEL" on page 123 |
| Reject a selected element | "TQREJEL" on page 124 |
| Reject a phase or all phases | "TQREJPH" on page 125 |
| Restore a phase | "TQRESPH" on page 126 |
| List phases related to the selected element(s) | "TQLISPH" on page 127 |
| List retained phases for the selected element(s) | "TQLISSF" on page 128 |
| Get data from the selected database | "TQGDAT" on page 129 |
| Reject defined system and reinitiate workspace | "TQREJSY" on page 130 |

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: <code>tc_initd.tdb</code> . | |
| Comments: | IERR = 1001 is Failed to find the initiation file. | |
| Arguments | | |
| 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 | tc_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(PHNAME, IWSG, IWSE) | |
| C-interface | tq_rejph(TC_STRING phase,TC_INT* iws,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 |
| PHNAME | Character*24 | Set to a phase name |
| | | <div><div></div><div>If * is used then all phases are rejected</div></div> |
| IWSG | Integer array | Workspace |
| IWSE | Integer array | Workspace |

TQRESPH

| | | |
|-------------|---|--|
| Fortran | TQRESPH(PHNAME, 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 |
| PHNAME | Character*24 | Set to a phase name <div> If * is used then all phases are rejected</div> |
| 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 19 or "TQINI3" on page 18 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



"About Adaptive Interpolation Schemes" on page 14

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 19](#) and thermodynamic information must be loaded, usually with the ["TQRFIL" on page 22](#) 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" on the next page |
| Initiate a branch in the interpolation scheme | "TQIPS_INIT_BRANCH" on page 133 |
| Define a function or state variable to be interpolated | "TQIPS_INIT_FUNCTION" on page 136 |
| Retrieve the interpolated value | "TQIPS_GET_VALUE" on page 137 |
| Write the data of the interpolation scheme to file. | "TQIPS_WRITE_IPS_DATA_TO_FILE" on page 138 |
| Read interpolation scheme data from file. | "TQIPS_READ_IPS_DATA_FROM_FILE" on page 139 |
| Get statistics on the usage of the interpolation scheme. | "TQIPS_GET_MEMORY_USAGE" on page 140 |

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. | |
| 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 discretization_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. | |
| 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. |

| | | |
|--------------------|--|---|
| 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 discretization_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) | |
| 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, 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 |

| | | |
|--------------------|--|--|
| 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 discretization_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) | |
| | | interpolation scheme for this branch (value < 1.0). If a value larger than 1.0 is 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. |
| 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. |
| 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. 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, IUSEDLOTS, 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 IUSEDLOTS/ISLOTS |
| ISLOTS | Integer | The total number of data slots allocated |
| IUSEDLOTS | 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" on the next page |
| Set ideal composition in this phase | "TQSETRX" on page 143 |
| Reorder CS in current EQ | "TQORDER" on page 144 |
| List content of IWSR set by user. | "TQLROX" on page 145 |

TQROINIT

| | | |
|-------------|---|---|
| Fortran | TQROINIT(NWSR, IWSR, IWSG, IWSE) | |
| C-interface | tq_roinit(TC_INT nwsr, TC_INT* iwsr,TC_INT* iwsr, 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 | | |
| 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



"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

There is different OS support for Windows and Linux as shown below.

Windows: Visual Studio 2010, Intel FORTRAN Composer 16

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
```



If you are using a newer compiler than the one supported, you might need to use hyphen (-) instead of underscore (_) when specifying these compiler switches:

/integer_size /real_size /double_size would then be /integer-size /real-size /double-size.

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.

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
```