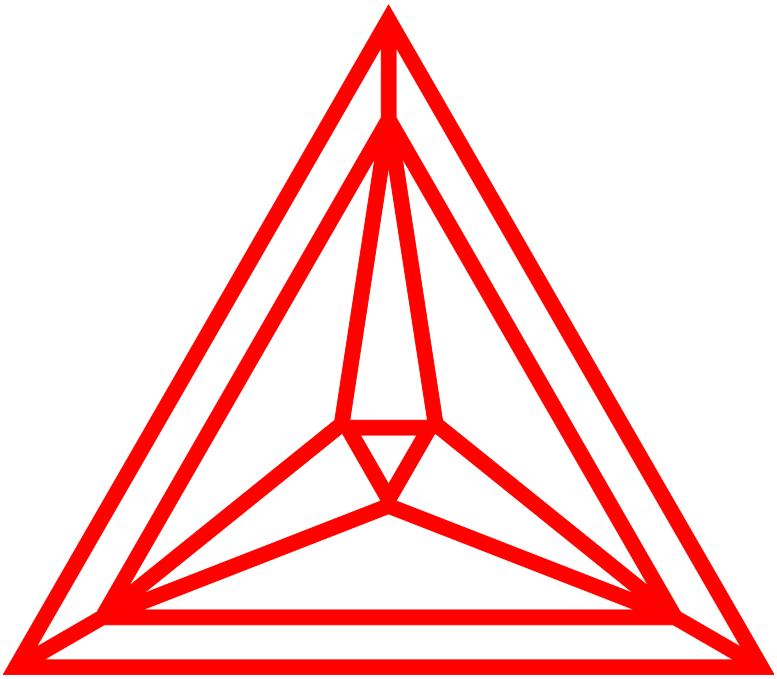
Thermodynamic Calculation Interface

TQ-Interface

(Version 8.0)

Programmer's Guide



Thermo-Calc Software AB

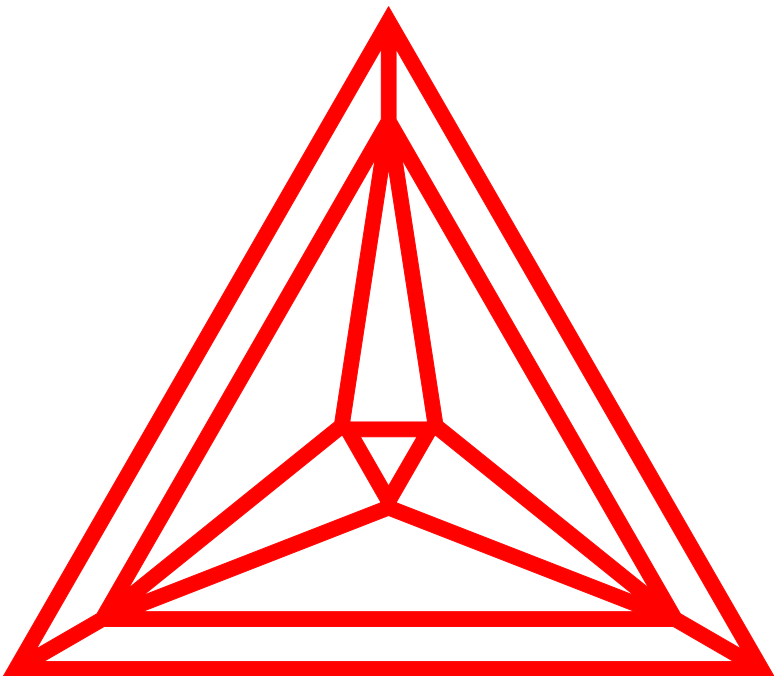
2013

Thermodynamic Calculation Interface

TQ-Interface

(Version 8.0)

Programmer's Guide



Copyright:

Disclaimers:

Acknowledgement of Copyright and Trademark Names:

CONTENT

1. INTRODUCTION 1

2. USE OF TQ INTERFACE 3

2.1 How to install and run TQ interface 3
2.2 How to use this guide 3
2.8 From version 7.0 to 8.0 3

3. BASIC CONCEPTS 4

3.1 Names 4
3.2 Components 5
3.3 Phases 5
3.3.1 Phase constituents 5
3.4 Programming interfaces 6
3.4.1 Fortran 6
3.4.2 C-interface 6
3.5 Adaptive interpolation scheme 7

4. DESCRIPTION OF TQ SUBROUTINES 8

Table 1. Initialization subroutines 9
Table 2. Possible units used by TQSSU and TQGSU 11
Table 3. Legal input/output options used by TQSIO and TQGIO 12
Table 4. Subroutines for identifying components, phases, and constituents 13
Table 5. Subroutines for changing the status of components and phases 15
Table 6. Subroutines for adding a contribution to the Gibbs energy of a phase 17
Table 7. Legal status for components 17
Table 8. Legal status for phases 17
Table 9. Subroutines for setting conditions, streams, or segments for 18
Table 10. Possible state variables to be used for setting conditions in TQSETC 20
Table 11. Subroutines and functions for doing calculation and getting results 21
Table 12. State variable that can be used in TQGETV and TQGET1 23
Table 13. Additional variables that can be used in TQGETV and TQGET1 24
Table 14. Subroutines and functions for debugging and error handling 24
Table 15. Subroutines for speedy retrieval of important properties of a phase 26
Table 16. Subroutines for retrieval of data from databases 28
Table 17. Subroutines for using the adaptive interpolation scheme 30
Table 18. Subroutines for reordering composition sets in TQ 32
4.1 Initialization Subroutines 33
4.1.1 TQINI(NWSG, NWSE, IWSG, IWSE) 33
4.1.2 TQSIO(OPTION, IVAL) 34
4.1.3 TQGIO(OPTION, IVAL) 34
4.1.4 TQRFIL(FILE, IWSG, IWSE) 34
4.1.6 TQGSU(QUANT, UNIT, IWSG, IWSE) 36
4.1.7 TQSAME(ICODE, IWSG, IWSE) 36
4.1.8 TQGVER(VERS, LNKDAT, OSNAME, BUILD, CMPLER) 36
4.2 System Subroutines 38
4.2.1 TQGNC (NCOM, IWSG, IWSE) 38
4.2.2 TQSCOM(NCOM, NAMES, STOI, IWSG, IWSE) 38
4.2.3 TQGCOM(NCOM, NAMES, IWSG, IWSE) 39
4.2.4 TQGSCI(INDEXC, NAME, IWSG, IWSE) 39
4.2.5 TQGNP (NPH, IWSG, IWSE) 40
4.2.6 TQGPN (INDEXP, NAME, IWSG, IWSE) 40
4.2.7 TQGPI (INDEXP, NAME, IWSG, IWSE) 40
4.2.8 TQGPCN(INDEXP, INDEXC, NAME, IWSG, IWSE) 41
4.2.9 TQGPCI(INDEXP, INDEXC, NAME, IWSG, IWSE) 41
4.2.10 TQGCCF(INDEXC, NEL, ELNAM, STOI, MMASS, IWSG, IWSE) 42

4.2.11 TQGPCS (INDEXP, INDEXC, STOI, MMASS, IWSG, IWSE) *42*
4.2.12 TQGNPC(INDEXP, NPCON, IWSG, IWSE) *43*
4.2.13 TQCSSC (INDEXC, STATUS, IWSG, IWSE) *43*
4.2.14 LOGICAL FUNCTION TQGSSC (INDEXC, IWSG, IWSE) *44*
4.2.15 TQCSP (INDEXP, STATUS, VAL, IWSG, IWSE) *44*
4.2.16 LOGICAL FUNCTION TQGSP (INDEXP, STATUS, VAL, IWSG, IWSE) *45*
4.2.17 TQSETR (INDEXC, INDEXP, TEMP, PRES, IWSG, IWSE) *45*
4.2.18 TQPACS (INDEXP, IWSG, IWSE) *46*
4.2.19 TQSGA (INDEXP, VALUE, IWSG, IWSE) *47*
4.2.20 TQGGA (INDEXP, VALUE, IWSG, IWSE) *48*
4.3 CONDITION, STREAM, AND SEGMENT SUBROUTINES 48
4.3.1 TQSETC(STAVAR, INDEXP, INDEXC, VAL, NUMCON, IWSG, IWSE) *48*
4.3.2 TQREMC(NUMCON, IWSG, IWSE) *49*
4.3.3 TQSCURC(IWSG, IWSE) *50*
4.3.4 TQREMAC(IWSG, IWSE) *50*
4.3.5 TQRESTC(IWSG, IWSE) *50*
4.3.6 TQCSTM(IDENT, TEMP, PRESS, IWSG, IWSE) *51*
4.3.7 TQSSC(IDENT, INDEXP, INDEXC, VALUE, NUMIN, IWSG, IWSE) *51*
4.3.8 TQSSIC(STAVAR, VALUE, IWSG, IWSE) *52*
4.3.9 TQDSTM(IDENT, IWSG, IWSE) *53*
4.3.10 TQNSEG(ID, IWSG, IWSE) *53*
4.3.11 TQSSEG(ID, IWSG, IWSE) *54*
4.4 CALCULATION AND RESULTS SUBROUTINES 55
4.4.1 TQCE(TARGET, INDEXP, INDEXC, VALUE, IWSG, IWSE) *55*
4.4.2 TQCEG(IWSG, IWSE) *56*
4.4.3 TQGETV(STAVAR, INDEXP, INDEXC, NUMBER, VALAR, IWSG, IWSE) *56*
4.4.4 TQGET1(STAVAR, INDEXP, INDEXC, VAL, IWSG, IWSE) *56*
4.4.5 DOUBLE PRECISION FUNCTION TQGMU (INDEXC, IWSG, IWSE) *58*
4.4.6 DOUBLE PRECISION FUNCTION TQGGM (INDEXP, IWSG, IWSE) *58*
4.4.7 TQGPD (INDEXP, NSUB, NSCON, SITES, YFRAC, EXTRA, IWSG, IWSE) *58*
4.4.8 TQGDF (IMATR, IPREC, NPH, NCOM, XMATR, XPREC, TEMP, DF, IWSG, IWSE) *60*
4.4.9 TQGDF2 (MODE, IMATR, IPREC, NIE, IIE, XMATR, TEMP, DF, XPREC, ... *61*
4.5 TROUBLESHOOTING SUBROUTINES 63
4.5.1 TQLS(IWSG, IWSE) *63*
4.5.2 TQLC(IWSG, IWSE) *63*
4.5.3 TQLE(IWSG, IWSE) *63*
4.5.4 TQFASV(IWSG, IWSE) *64*
4.5.5 TQSDMC(INDEXP, IWSG, IWSE) *64*
4.5.6 TQSSPC(INDEXP, YF, IWSG, IWSE) *64*
4.5.7 TQSSV(STAVAR, IP, IC, VALUE, IWSG, IWSE) *65*
4.5.8 TQPINI(IWSG, IWSE) *65*
4.5.9 TQSNL(MAXIT, ACC, YMIN, ADG, IWSG, IWSE) *66*
4.5.10 TQSMNG(NGP, IWSG, IWSE) *66*
4.5.11 TQSECO(IPDH, ICSS, IWSG, IWSE) *67*
4.5.12 ST1ERR(IERR, SUBR, MESS) *67*
4.5.13 ST2ERR(IERR, SUBR, MESS) *68*
4.5.14 LOGICAL FUNCTION SG1ERR or TQG1ERR(IERR) *68*
4.5.15 LOGICAL FUNCTION SG2ERR or TQG2ERR(IERR) *69*
4.5.16 LOGICAL FUNCTION SG3ERR or TQG3ERR(IERR, SUBR, MESS) *69*
4.5.17 RESERR (or TQRSERR) *70*
4.5.18 TQSP3F(FILE, IWSG, IWSE *70*
4.6 EXTRA SUBROUTINES 71
4.6.1 TQGMA(INDEXP, TP, YF, VAL, IWSG, IWSE) *71*
4.6.2 TQGMB(INDEXP, TP, VAL, IWSG, IWSE) *71*
4.6.3 TQGMC(INDEXP, VAL, IWSG, IWSE) *71*
4.6.4 TQGMDY(INDEXP, VARR, IWSG, IWSE) *72*
4.6.5 TQGMOB(INDEXP, ISP, VAL, IWSG, IWSE) *72*
4.6.6 TQSTP(TP, IWSG, IWSE) *73*

4.6.7 TQSYF(INDEXP, YF, IWSG, IWSE) *73*
4.6.8 TQGSSPI(SPN, ISP, IWSG, IWSE) *73*
4.6.9 TQCMOBA(INDEXP, ISP, IWSG, IWSE) *74*
4.6.10 TQCMOBB(INDEXP, IWSG, IWSE) *74*
4.6.11 TQDGYY(INDEXP, VARR1, VARR2, IWSG, IWSE) *74*
4.6.12 TQGPHP(INDEXP, NE, NCNV, NC, IWORK, WORK, IWSG, IWSE) *75*
4.6.13 TQX2Y(INDEXP, NE, NCNV, NC, IWORK, WORK, XF, YF, IWSG, IWSE) *76*
4.6.14 TQGMDX(IP, NE, NCNV, NC, IWORK, WORK, YF, VARR, GM, DGDX, ... *76*
4.7 DATABASE SUBROUTINES (SEE EXAMPLE 12) 78
4.7.1 TQGDBN(DB_ARR, N, IWSG, IWSE) *78*
4.7.2 TQOPDB(TDB, IWSG, IWSE) *78*
4.7.3 TQLIDE(EL_ARR, N, IWSG, IWSE) *78*
4.7.4 TQAPDB(TDB, IWSG, IWSE) *79*
4.7.5 TQDEFEL(ELNAM, IWSG, IWSE) *79*
4.7.6 TQREJEL(ELNAM, IWSG, IWSE) *79*
4.7.7 TQRESPH(PHNAM, IWSG, IWSE) *80*
4.7.8 TQREJPH(PHNAM, IWSG, IWSE) *80*
4.7.9 TQLISPH(PH_ARR, N, IWSG, IWSE) *81*
4.7.10 TQLISSF(PH_ARR, N, IWSG, IWSE) *81*
4.7.11 TQGDAT(IWSG, IWSE) *81*
4.7.12 TQREJS(IWSG, IWSE) *82*
4.8 INTERPOLATION SCHEME 83
4.8.1 TQIPS_INIT_TOP(NELC, NPHC, PHASESTR, IERR, IWSG, IWSE) *83*
4.8.2 TQIPS_INIT_BRANCH(TISCOND, TISCNST, PISCOND, PISCNST, IDEPEL, ... *83*
NSTEP, IPHSTA, T, TMIN, TMAX, P, PMIN, PMAX, RMEMFR, PHAMNT, ... *83*
XMIN, XMAX, IBRANCH, IERR IWSG, IWSE) *83*
4.8.3 TQIPS_INIT_FUNCTION(STRING, IBRANCH, IERR, IWSG, IWSE) *85*
4.8.4 TQIPS_GET_VALUE(IBRANCH, NOSCHEME, ARR, ... *85*
4.9 COMPOSITION SET REORDERING ROUTINES 87
4.9.1 TQROINIT(NWSR, IWSR, IWSG, IWSE) *87*
4.9.2 TQSETRX(PHASE, X, IWSR, IWSG, IWSE) *87*
4.9.3 TQORDER(IWSR, IWSG, IWSE) *87*
4.9.4 TQLROX(IWSR, IWSG, IWSE) *88*

6. APPENDIX 1 COMPILER SETTINGS 89

6.1 COMPILING FORTRAN CODE 89
6.1.1 Windows *89*
6.1.1.1 Visual Studio 2010, Intel Fortran Composer 12 89
6.1.1.1.1 32-bit configuration 89
6.1.1.1.2 64-bit configuration 89
6.1.1.2 Visual C++ 6.0, Compaq Visual Fortran 6.6C (not supported) 89
6.1.1.1.2.1 32-bit configuration 89
6.1.2 Linux *90*
6.1.2.1 GNU compiler version 4.4 90
6.1.2.1.1 32-bit configuration 90
6.1.2.1.2 64-bit configuration 90
6.1.2.2 Intel fortran compiler 90
6.1.2.2.1 32-bit configuration 90
6.1.2.2.2 64-bit configuration 90
6.2 COMPILING C CODE 91
6.2.1 Windows *91*
6.2.1.1 Visual Studio 2010 91
6.2.1.1.1 32-bit configuration 91
6.2.1.1.2 64-bit configuration 91
6.2.1.2 Visual Studio 6 91
6.2.1.1.2.1 32-bit configuration 91
6.2.2 Linux *91*
6.2.2.1 GNU compiler version 4.4 91
6.2.2.1.1 32-bit configuration 91
6.2.2.1.2 64-bit configuration 91

1. Introduction

TQ is an application programming interface of Thermo-Calc, a general software package for multi-component phase equilibrium calculations. TQ is intended for application programmers to write application programs using the Thermo-Calc kernel (*Figure 1*). 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. The thermodynamic properties and phase equilibrium data that can be obtained by using TQ 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 DICTRA mobility databases into the workspace, one can also obtain assessed mobility or diffusivity data via the TQ interface.

The TQ interface can be applied not only to calculate equilibrium state but also to predict metastable or non-equilibrium state by changing the status of the phases under consideration.

The FORTRAN subroutines and functions available in the TQ interface can be classified into six categories according to their purposes:

1) Initialization — Initializing the workspace, reading the thermodynamic data files, setting units for thermodynamic quantities, and selecting the input and output options.

2) System data manipulation — Identifying system components, phases, and constituents and changing their status.

3) Setting conditions, streams, and segments — Defining conditions for an equilibrium calculation.

4) Performing calculation.

5) Obtaining results.

6) Troubleshooting — Debugging and Error handling.

Essentially, only subroutines from the first, third, fourth and fifth categories are required for using this interface. In the simplest case, only one or two subroutines are needed from each category.

Same as the Thermo-Calc software, the TQ interface is available for both Linux platforms. It is supplied in the form of DLLs (Dynamically Linked Libraries) so that there is no necesscity to recompile existing application programs when a new version of TQ interface is available.

TQ is written in FORTRAN because many other software packages for scientific calculations are still developed in this language. *However, the actual computer language to be used 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 subroutines with appropriate calling conventions.*

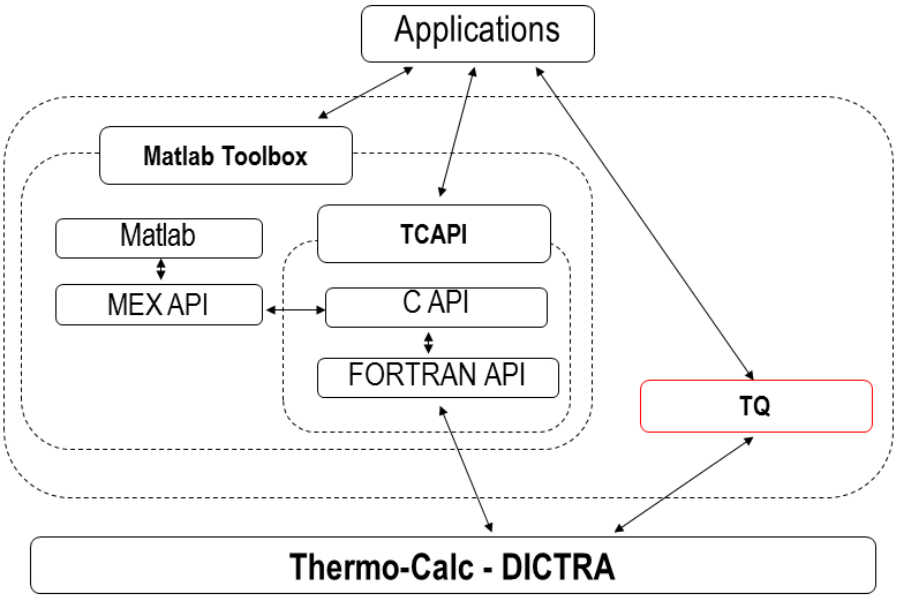


Figure 1. Interfacing with the Thermo-Calc Engine

2. Use of TQ Interface

2.1 How to install and run TQ interface

The TQ interface is only commercially available when it is formally purchased together with the Thermo-Calc software/database package. It requires a license key that is generated and provided by Thermo-Calc Software AB based on the hardware information of the installation computer/server provided by the user.

The installation of TQ is straightforward, run the installation program and follow the instructions. It is necessary to have the Thermo-Calc 3.0 software/database package installed on the same computer or on a server in order to run the TQ. For both Windows and Linux platforms, the TQ interface is supplied as a dynamically linked library. **All the examples referenced in this document are provided in the SDK installation directory.**

2.2 How to use this guide

Those who are not familiar with the Thermo-Calc software should start from Chapter 3, which defines some basic concepts used in TQ. Chapter 4 gives a full description of the subroutines included in TQ. It is written as a reference manual. Chapter 4 gives you an overview of the steps to program with the TQ interface. Several simple application examples are given in the installed SDK folder.

An experienced Thermo-Calc user can start by simply copying a suitable example and making it to work for his/her own problems by changing, adding, or deleting some callings to TQ subroutines and functions.

It is strongly recommended that one should compile and link at least one or two examples and make sure the linked executables can be run successfully.

2.8 From version 7.0 to 8.0

- Change of default compilation switches on Windows fortran compiler (see appendix).

- ifort compiler support on Linux.

- POLY-3 workspace is now also used when calling all TQ-Interface routines.

- Addition of some utility routines:

TQSP3F	- to save POLY-3 workspace file (for debugging purposes).
TQPACS	- Add a composition set to a phase.
TQGVER	- to obtain version and build information of the library.
TQROINIT	- Initialize IWSR workspace for reordering of composition sets

in TQ.

TQSETRX - Set ideal composition in a phase.
TQOREDER - ReOrder CS in current EQ.
TQLROX - List content of IWSR set by user.

- Addition of an adaptive interpolation scheme to rapidly obtain basic thermodynamic properties:

TQIPS_INIT_TOP	- initializes the interpolation scheme
TQIPS_INIT_BRANCH	- initializes a set of specific conditions where the interpolation is to be performed.
TQIPS_INIT_FUNCTION	- initializes a specific function or state variable that is to evaluated.
TQIPS_GET_VALUE	- returns the value(s) defined in TQIPS_INIT_FUNCTION

- Addition of a C programming interface matching the fortran routines.

3. Basic Concepts

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

A component is a system wide entity, sometimes this is emphasized by calling it a *system component*. A component has a unique name and some thermodynamic properties are associated with it, for example, 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 a lot of 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. These clusters may be modeled as constituents.

3.1 Names

A name of a component, phase or constituent must start with a letter A-Z or a-z and contain only letters, digits and the special characters underscore “_”, period “.”, parentheses “(” and “)”, plus “+”, minus “-”, and slash “/”. A name can be maximum 24 characters long.

3.2 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 like in h2o, c2h2cl_cis, au3cu_cvm1, my-favorite-component. Several subroutines are available for obtaining information about the components and to manipulate them. TQGCOM returns the total number of components and all component names, TQGSCI returns the index of one component name, TQSCOM makes it possible to re-define the components. The component index is used in most subroutines for defining conditions, etc.

Note that components can be suspended by TQCSSC, thus leaving "holes" in the component list because suspending one component will not change the sequential numbering. The logical function TQGSSC can be used to check if a specific component is suspended or not.

3.3 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 3.3.1.

The total number of phases is returned by TQGNP; the name of a phase by TQGPN or its index by TQGPI. The number of phase constituents is returned by TQGNPC.

Note that phases can be suspended or set dormant by TQCSP, thus leaving "holes" in the list because suspending one phase will not change the sequential numbering. The logical function TQGSP can be used to check if a specific phase is suspended or not.

3.3.1 Phase constituents

The TQ interface maintains a list of the constituents of each phase. 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 will go from the first constituent in the first sub-lattice over all sub-lattices to the last constituent in the last sub-lattice.

The number of constituents of a phase is returned by TQGNPC; the name of a phase constituent by TQGPCN or its index by TQGPCI. The stoichiometry of a constituent expressed in terms of the components is returned by TQGPCS.

3.4 Programming interfaces

For convienience the TQ-library offers the possibility program with the FORTRAN, C or Python programming languages.

3.4.1 Fortran

No special consideration need to be taken into account when interfacing the TQ-library with a main program written in FORTRAN as the main core of the TQ-library is written in FORTRAN. All parameters are by default passed to routines by reference with the exception for strings that are passed by descriptor.

3.4.2 C-interface

As the the default parameter passing mechanism in C is by value and not by reference, some consideration must be taken how parameters that are updated in the TQ-interface that are passed into the library.

The C-interface acts as a translation layer inbetween the calling C-program and the underlaying fortran TQ-library.

The C procedures are defined in the include file “tqroot.h” which should be included in the procedures using these library calls in C. The “tqroot.h” also includes the file “tc_data_defs.h” where the datatypes are defined. It should be noted that the definition of some of these datatype vary depending on which platform and compiler is used, it is therefore necessary provide the definitions for these to be correctly set (see appendix of compiler setttings).

The commonly used definitions in the C-interface are:

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 defininig the length of the string.

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

3.5 Adaptive interpolation scheme

A general dynamic interpolation scheme has been implemented in the TQ-library. This scheme gives the possibility at a slight cost of accuracy 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. Provided that 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 the scheme automatically extends the interpolation matrix. By 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. In the case that the memory requirements for extending the interpolation exceeds the available memory, the nodes in matrix that have less frequently used will be removed, thereby freeing memory.

For each set of condition values within a branch a unique identifying number is calculated, this number is then used to find the correct position in the interpolation matrix via a hash table.

In order to perform a simulation using the scheme, the TQ-library must be initialized in the normal manner using the routine TQINIT and thermodynamic information must be loaded, normally using the TQRFIL 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.

4. Description of TQ Subroutines

The subroutines and functions defined in the TQ interface can be divided into five categories according to their purposes:

- Initialization subroutines (see Table 1)
  - ⇒ Initializing the TQ workspace.
  - ⇒ Reading a thermodynamic data file into the workspace.
  - ⇒ Re-setting default units for temperature, pressure, energy, mass, and volume. A list of possible units is shown in Table 2.
  - ⇒ Changing the program input and output units. The legal options for program input/output are listed in Table 3.
- System subroutines (see Table 4-6)
  - ⇒ Getting number, names, and indexes of the system components, phases, and constituents of each phase.
  - ⇒ Re-define the system components.
  - ⇒ Changing the status of components and phases. The legal status is listed in Table 7 for components and Table 8 for phases.
  - ⇒ Changing the reference state of the system.
  - ⇒ Adding a contribution to the Gibbs energy of a phase.
- Condition, stream, and segment subroutines (see Table 9)
  - ⇒ Setting conditions for a thermodynamic equilibrium calculation. Possible state variables as conditions are listed in Table 10.
  - ⇒ Setting streams. 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.
  - ⇒ Setting a new equilibrium segment. Different sets of equilibrium conditions can be defined for the same system in different segments.
- Calculation subroutines (see Table 11)
  - ⇒ Performing calculation.
  - ⇒ Obtaining results. A list of state variables that can be used for getting results is given in Table 12.
- Troubleshooting subroutines (see Table 13)
  - ⇒ Getting information directly from the Thermo-Calc for debugging.
  - ⇒ Re-setting start values for an equilibrium calculation.
  - ⇒ Getting error number and message.

Table 1. Initialization subroutines

Subroutine	*Purpose*
TQINI3(DATABASE_PATH,TEMP_PATH,NWSG,NWSE,IWSG,IWSE)	Initialize TQ interface with user-specified database and temporary directories.
TQINI(NWSG,NWSE,IWSG,IWSE)	Initialize TQ interface
TQSIO(OPTION,IVAL)	Set input/output option
TQGIO(OPTION,IVAL)	Get input/output option
TQRFIL(FILE,IWSG,IWSE)	Read thermodynamic data file
TQSSU(QUANT,UNIT,IWSG,IWSE)	Set unit for a system quantity
TQGSU(QUANT,UNIT,IWSG,IWSE)	Get unit for a system quantity
TQSAME(ICODE,IWSG,IWSE)	Get unit for a system quantity
TQGVER(VERS,LNKDAT,OSNAME,BUILD,CMPLER)	Retrieve information about the TQ-library

C-procedure	*Purpose*
voidtq_ini3(TC_STRING database_path, TC_STRING temp_path, TC_INT nwsq, TC_INT nwse, TC_INT iwsq, TC_INT* iwse);*	Initialize TQ interface with user-specified database and temporary directories.
void tq_ini(TC_INT nwsq, TC_INT nwse, TC_INT iwsq, TC_INT* iwse);*	Initialize TQ interface
void tq_sio(TC_STRING option, TC_INT ival);	Set input/output option
void tq_qio(TC_STRING option, TC_INT ival);*	Get input/output option
void tq_rfil(TC_STRING file, TC_INT iwsq, TC_INT* iwse);*	Read thermodynamic data file
void tq_ssu(TC_STRING quant, TC_STRING unit, TC_INT iwsq, TC_INT* iwse);*	Set unit for a system quantity
void tq_qsu(TC_STRING quant, TC_STRING unit, TC_STRING_LENGTH strlen_unit, TC_INT iwsq, TC_INT* iwse);*	Get unit for a system quantity
void tq_same TC_INT icode, TC_INT* iwsq, TC_INT* iwse);*	Check if the system is same
void tq_qver(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);	Retrieve information about the TQ-library

Table 2. Possible units used by TQSSU and TQGSU

Quantity	*Unit*	*Comment*
Temperature	K	Kelvin (default)
	C	Celcius. Calculated as K-273.15
	F	Fahrenheit
Volume	M3	Cubic meter (default)
	L	Liter. Calculated as 0.001 M3
	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

Table 3. Legal input/output options used by TQSIO and TQGIO

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

Table 4. Subroutines for identifying components, phases, and constituents

Subroutine	*Purpose*
TQGNC(NCOM,IWSG,IWSE)	Get number of system components
TQSCOM(NCOM,NAMES,STOI,IWSG,IWSE)	Set system components
TQGCOM(NCOM,NAMES,IWSG,IWSE)	Get system components
TQGSCI(INDEXC,NAME,IWSG,IWSE)	Get system component index
TQGNP(NPH,IWSG,IWSE)	Get number of phases
TQGPN(INDEXP,NAME,IWSG,IWSE)	Get phase name
TQGPI(INDEXP,NAME,IWSG,IWSE)	Get phase index
TQGPCN(INDEXP,INDEXC,NAME,IWSG,IWSE)	Get phase constituent name
TQGPCI(INDEXP,INDEXC,NAME,IWSG,IWSE)	Get phase constituent index
TQGCCF(INDEXC,NEL,ELNAM,STOI,MMASS,IWSG, IWSE)	Get component chemical formula
TQGPCS(INDEXP,INDEXC,STOI,MMASS,IWSG,IWSE)	Get phase constituent stoichiometry
TQGNPC(INDEXP,NPCON,IWSG,IWSE)	Get number of phase constituents

C-procedure	*Purpose*
void tq_qnc(TC_INT ncom, TC_INT* iwsq, TC_INT* iwse);*	Get number of system components
void tq_scom(TC_INT num, tc_components_strings components, TC_FLOAT* stoi, TC_INT* iwsq, TC_INT* iwse);*	Set system components
void tq_qcom(TC_INT num, tc_components_strings* components, TC_INT* iwsq, TC_INT* iwse);*	Get system components
void tq_qsci(TC_INT index, TC_STRING component, TC_INT* iwsq, TC_INT* iwse);*	Get system component index
void tq_qnp(TC_INT nph, TC_INT* iwsq, TC_INT* iwse);*	Get number of phases
void tq_qpn(TC_INT index, TC_STRING phase, TC_STRING_LENGTH strlen_phase, TC_INT iwsq, TC_INT* iwse);*	Get phase name
void tq_qpi(TC_INT index, TC_STRING phase, TC_INT* iwsq, TC_INT* iwse);*	Get phase index
void tq_qpcn(TC_INT indexp, TC_INT indexc, TC_STRING name, TC_STRING_LENGTH strlen_name, TC_INT iwsq, TC_INT* iwse);*	Get phase constituent name
void tq_qpci(TC_INT indexp, TC_INT indexc, TC_STRING name, TC_INT* iwsq, TC_INT* iwse);*	Get phase constituent index

void tq_gccf(TC_INT indexc, TC_INT nel, tc_elements_strings* elname, TC_FLOAT* stoi, TC_FLOAT* mmass, TC_INT* iwsq, TC_INT* iwse);*	Get component chemical formula
void tq_gpcs(TC_INT indexp, TC_INT indexc, TC_FLOAT stoi, TC_FLOAT* mmass, TC_INT* iwsq, TC_INT* iwse);*	Get phase constituent stoichiometry
void tq_gnpc(TC_INT indexp, TC_INT npcon, TC_INT* iwsq, TC_INT* iwse);*	Get number of phase constituents

Table 5. Subroutines for changing the status of components and phases as well as the reference state of components

Subroutine	*Purpose*
TQCSSC(INDEXC,STATUS,IWSG,IWSE)	Change status of system component
TQGSSC(INDEXC,IWSG,IWSE)[*]	Get status of system component
TQCSP(INDEXP,STATUS,VAL,IWSG,IWSE)	Change status of phase
TQGSP(INDEXP,STATUS,VAL,IWSG,IWSE)[*]	Get status of phase
TQSETR(INDEXC,INDEXP,TEMP,PRES,IWSG, IWSE)	Set reference state
TQPACS(INDEXP,IWSG,IWSE)	Add a composition set to a phase

[*] ***Logical function.***

C-procedure	*Purpose*
void tq_cssc(TC_INT index, TC_STRING status, TC_INT iwsq, TC_INT* iwse);*	Change status of system component
TC_BOOL tq_gssc(TC_INT index, TC_STRING status, TC_STRING_LENGTH strlen_status, TC_INT iwsq, TC_INT* iwse);*	Get status of system component

void tq_csp(TC_INT index, ***TC_STRING status,*** ***TC_FLOAT amount,*** ***TC_INT* iwsq,*** ***TC_INT* iwse);***	Change status of phase
TC_BOOL tq_qsp(TC_INT index, ***TC_STRING status,*** ***TC_STRING_LENGTH*** ***strlen_status,*** ***TC_FLOAT* amount,*** ***TC_INT* iwsq,*** ***TC_INT* iwse);***	Get status of phase
void tq_setr(TC_INT indexc, ***TC_INT indexp,*** ***TC_FLOAT temp,*** ***TC_FLOAT press,*** ***TC_INT* iwsq,*** ***TC_INT* iwse);***	Set reference state
void tq_pacs(TC_INT indexp, ***TC_INT* iwsq,*** ***TC_INT* iwse);***	Add a composition set to a phase

Table 6. Subroutines for adding a contribution to the Gibbs energy of a phase

Subroutine	Purpose
TQSGA(INDEXP,VALUE,IWSG,IWSE) TQGGA(INDEXP,VALUE,IWSG,IWSE)	Set Gibbs energy addition Get Gibbs energy addition

C-procedure		Purpose
void tq_sga(	TC_INT indexp, TC_FLOAT value, TC_INT* iwsg, TC_INT* iwse);	Set Gibbs energy addition
void tq_gga(	TC_INT indexp, TC_FLOAT* value, TC_INT* iwsg, TC_INT* iwse);	Get Gibbs energy addition

Table 7. Legal status for components

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.

Table 8. Legal status for phases

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.

Table 9. Subroutines for setting conditions, streams, or segments for equilibrium calculations

Subroutine	*Purpose*
TQSETC(STAVAR,INDEXP,INDEXC,VAL,NUMCON,IWSG, IWSE)	Set condition
TQREMC(NUMCON,IWSG,IWSE)	Remove condition
TQSCURC(IWSG,IWSE)	Save current conditions
TQREMAC(IWSG,IWSE)	Remove all conditions
TQRESTC(IWSG,IWSE)	Restore saved conditions
TQCSTM(INDENT,TEMP,PRESS,IWSG,IWSE)	Create stream
TQSSC(IDENT,INDEXP,INDEXC,VALUE,NUMIN,IWSG, IWSE)	Set stream constituent amount
TQSSIC(STAVAR,VALUE,IWSG,IWSE)	Set stream invariant state variable
TQDSTM(IDENT,IWSG,IWSE)	Delete stream
TQNSEG(ID,IWSG,IWSE)	Create new equilibrium segment
TQSSEG(ID,IWSG,IWSE)	Select equilibrium segment

C-procedure	*Purpose*
void tq_setc(TC_STRING condition, TC_INT i1, TC_INT i2, TC_FLOAT value, TC_INT icond, TC_INT* iwsq, TC_INT* iwse);*	Set condition
void tq_remc(TC_INT index, TC_INT iwsq, TC_INT* iwse);*	Remove condition
void tq_scurc(TC_INT iwsq, TC_INT* iwse);*	Save current conditions
void tq_remac(TC_INT iwsq, TC_INT* iwse);*	Remove all conditions
void tq_restc(TC_INT iwsq, TC_INT* iwse);*	Restore saved conditions
void tq_cstm(TC_STRING stream, TC_FLOAT temp, TC_FLOAT press, TC_INT iwsq, TC_INT* iwse);*	Create stream
void tq_ssc(TC_STRING stream, TC_INT iph, TC_INT icmp, TC_FLOAT value, TC_INT icond, TC_INT iwsq, TC_INT* iwse);*	Set stream constituent amount
void tq_ssic(TC_STRING stavar, TC_FLOAT value, TC_INT iwsq, TC_INT* iwse);*	Set stream invariant state variable
void tq_dstm(TC_STRING stream, TC_INT iwsq, TC_INT* iwse);*	Delete stream
void tq_nseq(TC_STRING id, TC_INT iwsq, TC_INT* iwse);*	Create new equilibrium segment
void tq_sseq(TC_STRING id, TC_INT iwsq, TC_INT* iwse);*	Select equilibrium segment

Table 10. Possible state variables to be used for setting conditions in TQSETC

STAVAR	*INDEXP*	*INDEXC*	*Meaning*	*Comments*
T			Temperature	of the whole system
P			Pressure	of the whole system
MU	*note 1*	Yes	Chemical potential	of a system component
MUC	Yes	Yes	Chemical potential	of a phase constituent
AC	*note 1*	Yes	Activity	of a system component
ACC	Yes	Yes	Activity	of a system constituent
V			Volume	of the whole system
G			Gibbs energy	of the whole system
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
MP	*note 2*		Mass	of a phase
IN	Yes	Yes	Input amount	in moles of phase constituents
IM	Yes	Yes	Input amount	in mass units of phase constituents
X		Yes	Mole fraction	of a system component
W		Yes	Mass (Weight) fraction	of a system component
X%		Yes	Mole percent	of a system component
W%		Yes	Mass (Weight) fraction	of a system component

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

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

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

Table 11. Subroutines and functions for doing calculation and getting results

Subroutine	*Purpose*
TQCE(TARGET,INDEXP,INDEXC,VALUE,IWSG, IWSE)	Calculate equilibrium
TQCEG(IWSG,IWSE)	Calculate global equilibrium
TQGETV(STAVAR,INDEXP,INDEXC,NUMBER, VALAR,IWSG,IWSE)	Get equilibrium property values
TQGET1(STAVAR,INDEXP,INDEXC,VAL, IWSG,IWSE)	Get one value
TQGMU(INDEXC, IWSG,IWSE)[*]	Get chemical potential value
TQGGM(INDEXP, IWSG,IWSE)[*]	Get molar Gibbs energy value
TQGPD(INDEXP,NSUB,NSCON,SITES,YFRAC, EXTRA,IWSG,IWSE)	Get phase data
TQGDF(IMATR,IPREC,NPH,NCOM,XMATR, XPREC,TEMP,DF,IWSG,IWSE)	Get driving force for phase transformation. Obsolete
TQGDF2(MODE,IMATR,IPREC,NI,INI,XMATR, TEMP,DF,XPREC,XEM,XEP,MUI,IWSG, IWSE)	Get driving force and local equilibrium compositions for ortho- or para-equilibrium phase transformation

[*] ***Double precision function.***

Subroutine	*Purpose*
void tq_ce(TC_STRING var, TC_INT i1, TC_INT i2, TC_FLOAT value, TC_INT iwsq, TC_INT* iwse);*	Calculate equilibrium
void tq_ceq(TC_INT iwsq, TC_INT* iwse);*	Calculate global equilibrium
void tq_getv(TC_STRING var, TC_INT indexp, TC_INT indexc, TC_INT nvals, TC_FLOAT values, TC_INT* iwsq, TC_INT* iwse);*	Get equilibrium property values
void tq_get1(TC_STRING var, TC_INT indexp, TC_INT indexc, TC_FLOAT value, TC_INT* iwsq, TC_INT* iwse);*	Get one value
TC_FLOAT tq_gmu(TC_INT icmp, TC_INT iwsq, TC_INT* iwse);*	Get chemical potential value
TC_FLOAT tq_ggm(TC_INT iph, TC_INT iwsq, TC_INT* iwse);*	Get molar Gibbs energy value

void tq_gpd(	***TC_INT indexp,*** ***TC_INT* nsub,*** ***TC_INT* nscon,*** ***TC_FLOAT* sites,*** ***TC_FLOAT* yfrac,*** ***TC_FLOAT* extra,*** ***TC_INT* iwsq,*** ***TC_INT* iwse);***	Get phase data
void tq_gdf(	***TC_INT imatr,*** ***TC_INT iprec,*** ***TC_INT nph,*** ***TC_INT ncom,*** ***TC_FLOAT* xmatr,*** ***TC_FLOAT* xprec,*** ***TC_FLOAT* temp,*** ***TC_FLOAT* df,*** ***TC_INT* iwsq,*** ***TC_INT* iwse);***	Get driving force for phase transformation. Obsolete
void 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* iwsq,*** ***TC_INT* iwse);***	Get driving force and local equilibrium compositions for ortho- or para-equilibrium phase transformation

Table 12. State variable that can be used in TQGETV and TQGET1

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

***** One 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 in order to get a value which is calculated with respect to the reference state specified by calling TQSETR.

Table 13. Additional variables that can be used in TQGETV and TQGET1

STAVAR	IND EXP	IND EXC	Meaning	Unit
M(phase,J)			Mobility coefficient where J=diffusing specie	m^2/s
LOGM(phase,J)			^{10}log of the mobility coefficient	m^2/s
DT(phase,J)			Tracer diffusion coefficient where J=diffusing specie	m^2/s
LOGDT(phase,J)			^{10}log of the tracer diffusion coefficient	m^2/s
DC(phase,J,K,N)			Chemical diffusion coefficient where K=gradient specie, and N=reference specie	m^2/s
LOGDC(phase,J,K,N)			^{10}log of the chemical diffusion coefficient	m^2/s
DI(phase,J,K,N)			Intrinsic diffusion coefficient	m^2/s
LOGDI(phase,J,K,N)			^{10}log of the intrinsic diffusion coefficient	m^2/s
QC(phase,J,K,N)			$Q=R(ln(DC\{T_1\})-ln(DC\{T_1+\varepsilon\}))/(1/(T_1+\varepsilon)-1/T_1)$	J/mol
QT(phase,J)			$Q=R(ln(DT\{T_1\})-ln(DT\{T_1+\varepsilon\}))/(1/(T_1+\varepsilon)-1/T_1)$	J/mol
QI(phase,J,K,N)			$Q=R(ln(DI\{T_1\})-ln(DI\{T_1+\varepsilon\}))/(1/(T_1+\varepsilon)-1/T_1)$	J/mol
FC(phase,J,K,N)			$D0=\exp(\ln(DC\{T_1\})+Q/R/T_1)$	m^2/s
FI(phase,J,K,N)			$D0=\exp(\ln(DI\{T_1\})+Q/R/T_1)$	m^2/s
FT(phase,J)			$D0=\exp(\ln(DT\{T_1\})+Q/R/T_1)$	m^2/s

Table 14. Subroutines and functions for debugging and error handling

Subroutine	Purpose
TQLS(IWSG,IWSE)	List status
TQLC(IWSG,IWSE)	List conditions
TQLE(IWSG,IWSE)	List equilibrium
TQFASV(IWSG,IWSE)	Force automatic start values
TQSDMC(INDEXP,IWSG,IWSE)	Set default major constituent
TQSSPC(INDEXP,YF,IWSG,IWSE)	Set start phase constitution
TQSSV(STAVAR,IP,IC,VALUE,IWSG,IWSE)	Set start value of a state variable
TQPINI(IWSG,IWSE)	Reinitiate the calculation workspace
TQSNL(MAXIT,ACC,YMIN,ADG,IWSG,IWSE)	Set numerical limits
TQSMNG(NGP,IWSG,IWSE)	Set maximum number of grid points
TQSECO(IPDH,ICSS,,IWSG,IWSE)	Set equilibrium calculation options
ST1ERR(IERR,SUBR,MESS)	Set error code and give message
ST2ERR(IERR,SUBR,MESS)	Set error code
SG1ERR(IERR)*	Get error code and give message
SG2ERR(IERR)*	Get error code
SG3ERR(IERR,SUBR,MESS)*	Get error code and message
RESERR	Reset error code and message
TQSP3F(FILE,IWSG,IWSE)	Save a POLY-3 file

**** Logical function.***

C-procedure	*Purpose*
void tq_ls(TC_INT* iwsq, TC_INT* iwse);	List status
void tq_lc(TC_INT* iwsq, TC_INT* iwse);	List conditions
void tq_le(TC_INT* iwsq, TC_INT* iwse);	List equilibrium
void tq_le(TC_INT* iwsq, TC_INT* iwse);	Force automatic start values
void tq_sdmc(TC_INT indexp, TC_INT* iwsq, TC_INT* iwse);	Set default major constituent
void tq_sspc(TC_INT indexp, TC_FLOAT* vf, TC_INT* iwsq, TC_INT* iwse);	Set start phase constitution
void tq_ssv(TC_STRING stavar, TC_INT ip, TC_INT ic, TC_FLOAT value, TC_INT* iwsq, TC_INT* iwse);	Set start value of a state variable
void tq_pini(TC_INT* iwsq, TC_INT* iwse);	Reinitiate the calculation workspace
void tq_snl(TC_INT maxit, TC_FLOAT acc, TC_FLOAT ymin, TC_FLOAT adq, TC_INT* iwsq, TC_INT* iwse);	Set numerical limits
void tq_smng(TC_INT ngp, TC_INT* iwsq, TC_INT* iwse);	Set maximum number of grid points
void tq_seco(TC_INT ipdh, TC_INT icss, TC_INT* iwsq, TC_INT* iwse);	Set equilibrium calculation options
void tq_st1err(TC_INT ierr, TC_STRING subr, TC_STRING mess);	Set error code and give message
void tq_st2err(TC_INT ierr, TC_STRING subr, TC_STRING mess);	Set error code
TC_BOOL tq_sq1err(TC_INT* ierr);	Get error code and give message
TC_BOOL tq_sq2err(TC_INT* ierr);	Get error code

C-procedure	*Purpose*
TC_BOOL tq_sq3err(TC_INT* ierr, TC_STRING subr, TC_STRING_LENGTH strlen_subr, TC_STRING mess, TC_STRING_LENGTH strlen_mess);	Get error code and message
void tq_reserr( );	Reset error code and message
void tq_sp3f(TC_STRING filename, TC_INT* iwsq, TC_INT* iwse);	Save a POLY-3 file

Table 15. Subroutines for speedy retrieval of important properties of a phase

Subroutine	*Purpose*
TQGMA(INDEXP,TP,YF,VAL,IWSG,IWSE)	Get Gibbs energy of a phase*
TQGMB(INDEXP,TP,VAL,IWSG,IWSE)	Get Gibbs energy of a phase*
TQGMC(INDEXP,VAL,IWSG,IWSE)	Get Gibbs energy of a phase*
TQGMDY(INDEXP,VARR,IWSG,IWSE)	Get 1st partial derivative of Gibbs energy w.r.t. site fractions.*
TQDGYY(INDEXP,VARR1,VARR2,IWSG, IWSE)	Get 1st and 2nd partial derivative of Gibbs energy w.r.t. site fractions.*
TQGPHP(INDEXP,NE,NCNV,NC,IWORK, WORK,IWSG,IWSE)	Get constitutional properties of a phase.*
TQX2Y(INDEXP,NE,NCNV,NC,IWORK, WORK,X,YF,IWSG,IWSE)	Convert mole fraction to site fraction for phases with no internal degree of freedom.*
TQGMDX(INDEXP,NE,NCNV,NC,IWORK, WORK,YF,DGDY,GM,DGDX,X,IWSG, IWSE)	Convert 1st partial derivative of Gibbs energy w.r.t. site fractions to that w.r.t. mole fractions.*
TQGMOB(INDEXP,ISP,VAL,IWSG,IWSE)	Get mobility of a species in a phase
TQSTP(TPARR,IWSG,IWSE)	Set temperature and pressure for TQGMC, TQGMDY, and TQGMOB
TQSYF(INDEXP,YF,IWSG,IWSE)	Set site fractions for TQGMB, TQGMC, TQGMDY, and TQGMOB
TQGSSPI(SPN,ISP,IWSG,IWSE)	Get index of a system species
TQCMOBA(INDEXP,ISP,IWSG,IWSE)†	Check if mobility of a species in a phase available
TQCMOBB(INDEXP,IWSG,IWSE)†	Check if mobility data for a phase available

*in SI unit for one mole of formula unit
† Logical function.

C-procedure	*Purpose*
void tq_gma(TC_INT indexp, TC_FLOAT* tp, TC_FLOAT* yf, TC_FLOAT* varr, TC_INT* iwsq, TC_INT* iwse);	Get Gibbs energy of a phase*
void tq_gmb(TC_INT indexp, TC_FLOAT* tp, TC_FLOAT* varr, TC_INT* iwsq, TC_INT* iwse);	Get Gibbs energy of a phase*
void tq_gmc(TC_INT indexp, TC_FLOAT* varr, TC_INT* iwsq, TC_INT* iwse);	Get Gibbs energy of a phase*
void tq_gmdy(TC_INT indexp, TC_FLOAT* varr, TC_INT* iwsq, TC_INT* iwse);	Get 1st partial derivative of Gibbs energy w.r.t. site fractions.*
void tq_dgyy(TC_INT indexp, TC_FLOAT* varr1, TC_FLOAT* varr2, TC_INT* iwsq, TC_INT* iwse);	Get 1st and 2nd partial derivative of Gibbs energy w.r.t. site fractions.*
void tq_gphp(TC_INT indexp, TC_INT* ne, TC_INT* ncnv, TC_INT* nc, TC_INT* iwork, TC_FLOAT* work, TC_INT* iwsq, TC_INT* iwse);	Get constitutional properties of a phase.*
void 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* iwsq, TC_INT* iwse);	Convert mole fraction to site fraction for phases with no internal degree of freedom.*
void 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* qm, TC_FLOAT* dqdx, TC_FLOAT* xf, TC_INT* iwsq,	Convert 1st partial derivative of Gibbs energy w.r.t. site fractions to that w.r.t. mole fractions.*

TC_INT iwse);*	
void tq_gmob(TC_INT indexp, *TC_INT isp,* *TC_FLOAT* val,* *TC_INT* iwsq,* *TC_INT* iwse);*	Get mobility of a species in a phase
void tq_stp(TC_FLOAT tp,* *TC_INT* iwsq,* *TC_INT* iwse);*	Set temperature and pressure for TQGMC, TQGMDY, and TQGMOB
void tq_syf(TC_INT indexp, *TC_FLOAT* yf,* *TC_INT* iwsq,* *TC_INT* iwse);*	Set site fractions for TQGMB, TQGMC, TQGMDY, and TQGMOB
void tq_gsspi(TC_STRING name, *TC_INT* index,* *TC_INT* iwsq,* *TC_INT* iwse);*	Get index of a system species
TC_BOOL tq_cmoba(TC_INT indexp, *TC_INT* iwsq,* *TC_INT* iwse);*	Check if mobility of a species in a phase available
TC_BOOL tq_cmobb(TC_INT indexp, *TC_INT* iwsq,* *TC_INT* iwse);*	Check if mobility data for a phase available

Table 16. Subroutines for retrieval of data from databases

Subroutine	*Purpose*
TQGDBN(DB_ARR,N,IWSG,IWSE)	Get lists of database names
TQOPDB(TDB,IWSG,IWSE)	Open or switch to a database
TQLIDE(EL_ARR,N,IWSG,IWSE)	List database elements
TQAPDB(TDB,IWSG,IWSE)	Append a database
TQDEFEL(ELNAM,IWSG,IWSE)	Select an element
TQREJEL(ELNAM,IWSG,IWSE)	Reject a selected element
TQREJPH(PHNAM,IWSG,IWSE)	Reject a phase or all phases
TQRESPH(PHNAM,IWSG,IWSE)	Restore a phase
TQLISPH(PH_ARR,N,IWSG,IWSE)	List phases related to the selected element(s)
TQLISSF(PH_ARR,N,IWSG,IWSE)	List retained phases for the selected element(s)
TQGDAT(IWSG,IWSE)	Get data from the selected database
TQREJS(IWSG,IWSE)	Reject defined system and reinitiate workspace

C-procedure	*Purpose*
void tq_qdbn(tc_databases_strings databases, TC_INT* n, TC_INT* iwsq, TC_INT* iwse);*	Get lists of database names
void tq_opdb(TC_STRING database, TC_INT iwsq, TC_INT* iwse);*	Open or switch to a database
void tq_lide(tc_elements_strings elements, TC_INT* num, TC_INT* iwsq, TC_INT* iwse);*	List database elements
void tq_apdb(TC_STRING database, TC_INT iwsq, TC_INT* iwse);*	Append a database
void tq_defel(TC_STRING element, TC_INT iwsq, TC_INT* iwse);*	Select an element
void tq_rejel(TC_STRING element, TC_INT iwsq, TC_INT* iwse);*	Reject a selected element
void tq_rejph(TC_STRING phase, TC_INT iwsq, TC_INT* iwse);*	Reject a phase or all phases
void tq_resph(TC_STRING phase, TC_INT iwsq, TC_INT* iwse);*	Restore a phase
void tq_lisph(tc_phases_strings phases, TC_INT* num, TC_INT* iwsq, TC_INT* iwse);*	List phases related to the selected element(s)
void tq_lissf(tc_phases_strings phases, TC_INT* num, TC_INT* iwsq, TC_INT* iwse);*	List retained phases for the selected element(s)
void tq_qdat(TC_INT iwsq, TC_INT* iwse);*	Get data from the selected database
void tq_rejsy(TC_INT iwsq, TC_INT* iwse);*	Reject defined system and reinitiate workspace

Table 17. Subroutines for using the adaptive interpolation scheme

Subroutine	*Purpose*
TQIPS_INIT_TOP(IERR, IWSG, IWSE)	Initiates the interpolation scheme
TQIPS_INIT_BRANCH(TISCOND, TISCNST, PISCOND, PISCNST, IDEPEL, NSTEP, IPHSTA, T, TMIN, TMAX, P, PMIN, PMAX, RMEMFRAC, PHAMNT, XMIN, XMAX, IBRANCH, IERR, IWSG, IWSE)	Initiates a branch in the interpolation scheme
TQIPS_INIT_FUNCTION(STRING, BRANCH, IERR IWSG, IWSE)	Defines a function or statevariable to be interpolated
TQIPS_GET_VALUE(IBRANCH, NOSCHEME, ARR, RESULT, IERR, ISHORT IWSG, IWSE)	Retrieves the interpolated value

C-procedure	*Purpose*
void tq_ips_init_top(*TC_INT* iwsq,* *TC_INT* iwse,* *TC_INT* err);*	Initiates the interpolation scheme
void tq_ips_init_branch(*TC_INT* iwsq,* *TC_INT* iwse,* *TC_BOOL t_is_condition,* *TC_BOOL t_is_constant,* *TC_BOOL p_is_condition,* *TC_BOOL p_is_constant,* *TC_BOOL* independent_elements,* *TC_INT dicretization_type,* *TC_INT nr_of_steps,* *TC_INT* state_of_phases,* *TC_FLOAT t,* *TC_FLOAT tmin,* *TC_FLOAT tmax,* *TC_FLOAT p,* *TC_FLOAT pmin,* *TC_FLOAT pmax,* *TC_FLOAT memory_fraction,* *TC_FLOAT* amount_of_phases,* *TC_FLOAT* xmin,* *TC_FLOAT* xmax,* *TC_INT* branch_nr,* *TC_INT* err);*	Initiates a branch in the interpolation scheme
void tq_ips_init_function(*TC_INT* iwsq,* *TC_INT* iwse,* *TC_STRING function_string,* *TC_INT branch_nr,* *TC_INT* err);*	Defines a function or statevariable to be interpolated
void tq_ips_get_value(*TC_INT* iwsq,* *TC_INT* iwse,* *TC_INT branch_nr,* *TC_INT noscheme,* *TC_FLOAT* variable_values,* *TC_FLOAT* function_values,* *TC_INT* err,* *TC_INT* shortcut);*	Retrieves the interpolated value

Table 18. Subroutines for reordering composition sets in TQ

Subroutine	*Purpose*
TQROINIT(NWSR,IWSR,IWSG,IWSE)	Initialize IWSR workspace for reordering of CS in TQ.
TQSETRX(PHASE, X,IWSR,IWSG,IWSE)	Set ideal composition in this phase
TQORDER(IWSR,IWSG,IWSE)	ReOrder CS in current EQ
TQLROX(IWSR,IWSG,IWSE)	List content of IWSR set by user.

C-procedure	*Purpose*
void tq_roinit(TC_INT nwsr, TC_INT* iwsr, TC_INT* iwsg, TC_INT* iwse);	Initialize IWSR workspace for reordering of CS in TQ.
void tq_setrx(TC_STRING phase, TC_FLOAT* x, TC_INT* iwsr, TC_INT* iwsg, TC_INT* iwse);	Set ideal composition in this phase
void tq_order(TC_INT* iwsr, TC_INT* iwsg, TC_INT* iwse);	ReOrder CS in current EQ
void tq_lrox(TC_INT* iwsr, TC_INT* iwsg, TC_INT* iwse);	List content of IWSR set by user.

4.1 Initialization Subroutines

4.1.0 TQINI3(DATABASE_PATH, TEMP_PATH, NWSG, NWSE, IWSG, IWSE)

Full name: Initialize TQ interface with user-specified database and temporary directories. If a GES file will be used (i.e. no databases will be opened) the directories can be empty strings.

Purpose: With this subroutine the application program initializes the Thermo-Calc package for thermodynamic calculations. This or TQINI must be called before using any other subroutines in the TQ interface.

Arguments:

Name	Type	Value set on call or returned
database_path	Character*256	Path to the directory that holds the data directory, which in turn contains the databases.
temp_path	Character*256	Path to the directory for temporary and log file output.
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.

C-interface:

```
tq_ini3( TC_STRING database_path,
         TC_STRING temp_path,
         TC_INT nwsg,
         TC_INT nwse,
         TC_INT* iwsg,
         TC_INT* iwse);
```

4.1.1 TQINI(NWSG, NWSE, IWSG, 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.

C-interface:

```
tq_ini( TC_INT nwsg,
        TC_INT nwse,
        TC_INT* iwsg,
        TC_INT* iwse);
```

4.1.2 TQSIO(OPTION, 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.

Arguments:

Name	Type	Value set on call or returned
OPTION	Character*8	Set to a value given in Table 3.
IVAL	Integer	Set to an internal value.

C-interface:
```
tq_sio( TC_STRING option,
        TC_INT ival);
```

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.

4.1.3 TQGIO(OPTION, IVAL)

Full name: Get Input/output Unit.

Purpose: Obtain a value of Input/Output option.

Arguments:

Name	Type	Value set on call or returned
OPTION	Character*8	Set to a value given in Table 3.
IVAL	Integer	Return the current internal value.

C-interface:
```
tq_gio( TC_STRING option,
        TC_INT* ival);
```

Comments: OPTION is a character identifying the Input/Output option. IVAL is an integer where its current internal value is returned.

4.1.4 TQRFIL(FILE, IWSG, IWSE)

Full name: Read File.

Purpose: Read a thermodynamic data file in the Thermo-Calc format.

Arguments:

Name	Type	Value set on call or returned

FILE	Character*60	Legal file name.
IWSG	Integer array	Workspace.
IWSE	Integer array	Workspace

C-interface:
```
tq_rfil( TC_STRING file,
         TC_INT* iwsg,
         TC_INT* iwse);
```

Comments: The default set of components is supplied by the thermodynamic input file.

The thermodynamic data file should contain at least the following information.

System	⇒ name of the elements
	⇒ molecular mass for elements
	⇒ list of phases
Phase	⇒ list of constituents
	⇒ type of solution model (if not fixed composition)
	⇒ thermodynamic model parameters
Constituents	⇒ name
	⇒ chemical formula (stoichiometric matrix)
	⇒ molecular mass
	⇒ thermodynamic properties

All this data are not necessarily stored separately, for example the molecular weight for a constituent can be calculated from the masses of the elements.

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 the Thermo-Calc and then stored in a GES file by using the `save` command in the Gibbs-Energy-System module inside the Thermo-Calc. When the GES file has been read into the workspace by this subroutine it is possible to manipulate data by changing components and status for components or phases.

4.1.5 TQSSU(QUANT, UNIT, IWSG, IWSE)

Full name: Set System Unit.

Purpose: Set the unit for a quantity (like mass, volume, etc.).

Arguments:

Name	Type	Value set on call or returned
QUANT	Character*60	Set to a legal quantity as listed in Table 2.
UNIT	Character*60	Set to a legal unit, see below in Table 2.
IWSG	Integer array	Workspace.
IWSE	Integer array	Workspace.

C-interface:
```
tq_ssu( TC_STRING quant,
        TC_STRING unit,
        TC_INT* iwsg,
        TC_INT* iwse);
```

Comments: Default units are SI unless changes are made by this subroutine. The legal quantities and units are listed in Table 2

4.1.6 TQGSU(QUANT, UNIT, IWSG, IWSE)

Full name: Get System Unit.

Purpose: To find what units the TQ interface is currently using for a system quantity.

Arguments:

Name	Type	Value set on call or returned
QUANT	Character*60	Set to a legal quantity listed in Table 2.
UNIT	Character*60	Return the current unit.
IWSG	Integer array	Workspace.
IWSE	Integer array	Workspace.

C-interface:

```
tq_gsu( TC_STRING quant,
        TC_STRING unit,
        TC_STRING_LENGTH strlen_unit,
        TC_INT* iwsg,
        TC_INT* iwse);
```

Comments: The legal quantities and units are listed in Table 2.

4.1.7 TQSAME(ICODE, IWSG, IWSE)

Full name: Same System.

Purpose: With this subroutine 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.

Arguments:

Name	Type	Value set on call or returned
ICODE	Integer	Returns an internal code.
IWSG	Integer array	Workspace.
IWSE	Integer array	Workspace.

C-interface:

```
tq_same(    TC_INT* icode,
            TC_INT* iwsg,
            TC_INT* iwse);
```

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.

4.1.8 TQGVER(VERS, LNKDAT, OSNAME, BUILD, CMPLER)

Full name: Get version.

Purpose: With this subroutine the application program can obtain information about the used 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-libray 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-libray was built.

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);
```

Comments:

4.2 System Subroutines

4.2.1 TQGNC (NCOM, IWSG, IWSE)

Full name: Get Number of components.

Purpose: With this subroutine 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.

C-interface:
```
tq_gnc( TC_INT* ncom,
        TC_INT* iwsg,
        TC_INT* iwse);
```

Comments:

4.2.2 TQSCOM(NCOM, NAMES, STOI, IWSG, 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.

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.
IWSG	Integer array	Workspace.
IWSE	Integer array	Workspace.

C-interface:
```
tq_scom(    TC_INT num,
            tc_components_strings* components,
            TC_FLOAT* stoi,
            TC_INT* iwsg,
            TC_INT* iwse);
```

Comments: The set of components must be linearly independent. The names of the new system components are given in NAMES. STOI is a matrix with dimension STOI(1:NCOM,1:NCOM) which gives the stoichiometry of the new components expressed in the old ones.

The default set for components is taken from in the input thermodynamic data file.

Legal values for the array elements in NAMES are constituent names.

The components will be 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.

Example: To transform from the components A, B, C to A2B, B4C C2 use the following values of STOI:

A2B	2.0	1.0	0.0
B4C	0.0	4.0	1.0
C2	0.0	0.0	2.0

4.2.3 TQGCOM(NCOM, NAMES, IWSG, IWSE)

Full name: Get System Component.

Purpose: Get components of a system

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.

C-interface:

```
tq_gcom(    TC_INT* num,
            tc_components_strings* components,
            TC_INT* iwsg,
            TC_INT* iwse);
```

Comments: The number of components are returned in NCOM and their names are returned in NAMES. They are returned in an internal sequential order of the TQ interface. In other subroutines one must in some cases use the index of a component rather than the name. See also TQGSCI.

4.2.4 TQGSCI(INDEXC, NAME, IWSG, IWSE)

Full name: Get System Component Index.

Purpose: Get index of a system component.

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.

C-interface:

```
tq_gsci( TC_INT* index,
         TC_STRING component,
         TC_INT* iwsg,
         TC_INT* iwse);
```

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. The application program may call TQGCOM only once and maintain itself a list of component names stored by indices.

4.2.5 TQGNP (NPH, IWSG, IWSE)

Full name: Get Number of Phases.

Purpose: With this subroutine application program can get numbers of phases.

Arguments:

Name	Type	Value set on call or returned
NPH	Integer	Return the number of phases.
IWSG	Integer array	Workspace.
IWSE	Integer array	Workspace.

C-interface:
```
tq_gnp( TC_INT* nph,
        TC_INT* iwsg,
        TC_INT* iwse);
```

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.

4.2.6 TQGPN (INDEXP, NAME, IWSG, IWSE)

Full name: Get Phase Name.

Purpose: With this subroutine application program can convert a phase index to the name of the phase.

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.

C-interface:
```
tq_gpn( TC_INT index,
        TC_STRING phase,
        TC_STRING_LENGTH strlen_phase,
        TC_INT* iwsg,
        TC_INT* iwse);
```

Comments: The conversion from phase name to phase index is done by TQGPI. 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.

4.2.7 TQGPI (INDEXP, NAME, IWSG, IWSE)

Full name: Get Phase Index.

Purpose: With this subroutine the application program can get the index of a named phase.

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.

C-interface:
```
tq_gpi( TC_INT* index,
        TC_STRING phase,
        TC_INT* iwsg,
        TC_INT* iwse);
```

Comments: The conversion from phase index to phase name is done by TQGPN.

4.2.8 TQGPCN(INDEXP, INDEXC, NAME, IWSG, IWSE)

Full name: Get Phase Constituent Name.

Purpose: With this subroutine application program can get the name of an indexed constituent.

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.

C-interface:
```
tq_gpcn(TC_INT indexp,
        TC_INT indexc,
        TC_STRING name,
        TC_STRING_LENGTH strlen_name,
        TC_INT* iwsg,
        TC_INT* iwse);
```

Comments: If the same species appear in more than one sublattice site of a phase, they will be 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.

4.2.9 TQGPCI(INDEXP, INDEXC, NAME, IWSG, IWSE)

Full name: Get Phase Constituent Index.

Purpose: With this subroutine application program can get the index of a constituent if its name is known.

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.

C-interface:
```
tq_gpci( TC_INT indexp,
         TC_INT* indexc,
         TC_STRING name,
         TC_INT* iwsg,
         TC_INT* iwse);
```

Comments: The opposite conversion is done by TQGPCN.

4.2.10 TQGCCF(INDEXC, NEL, ELNAM, STOI, MMASS, IWSG, IWSE)

Full name: Get Component Chemical Formula.

Purpose: With this subroutine application program can get the stoichiometry array for a system component in terms of element.

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.

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);
```

Comments: With this subroutine it is possible to 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.
Note: the dimension of ELNAM and STOI will be NEL (number of elements in the component).

4.2.11 TQGPCS (INDEXP, INDEXC, STOI, MMASS, IWSG, IWSE)

Full name: Get Phase Constituent Stoichiometry.

Purpose: With this subroutine application program can obtain the stoichiometry of a constituent expressed in the system components and also the molecular mass.

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.

C-interface:
```
tq_gpcs( TC_INT indexp,
         TC_INT indexc,
         TC_FLOAT* stoi,
         TC_FLOAT* mmass,
         TC_INT* iwsg,
         TC_INT* iwse);
```

Comments: This does not give the chemical formula in terms of elements for the constituent. The dimension of STOI will be NCOM (number of components) got by calling TQGCOM.

4.2.12 TQGNPC(INDEXP, NPCON, IWSG, IWSE)

Full name: Get Number of Phase Constituent.

Purpose: With this subroutine the number of constituents in a phase can be obtained.

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.

C-interface:
```
tq_gnpc(TC_INT indexp,
        TC_INT* npcon,
        TC_INT* iwsg,
        TC_INT* iwse);
```

Comments: To have also the names, fractions etc. of the constituents, use TQGPD.

4.2.13 TQCSSC (INDEXC, STATUS, IWSG, IWSE)

Full name: Change Status of System Component.

Purpose: With this subroutine the application program can change status for a system component.

Arguments:

Name	Type	Value set on call or returned
INDEXC	Integer	Set to a component index.
STATUS	Character*12	Set to the new status (see Table 7).
IWSG	Integer array	Workspace.
IWSE	Integer array	Workspace.

C-interface:
```
tq_cssc( TC_INT index,
         TC_STRING status,
         TC_INT* iwsg,
         TC_INT* iwse);
```

Comments: The legal values for STATUS are **ENTERED** and **SUSPENDED**. See also Table 7.

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.

4.2.14 LOGICAL FUNCTION TQGSSC (INDEXC, IWSG, 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**.

Arguments:

Name	Type	Value set on call or returned
INDEXC	Integer	Set to a component index.
IWSG	Integer array	Workspace.
IWSE	Integer array	Workspace.

C-interface:
```
tq_gssc( TC_INT index,
         TC_STRING status,
         TC_STRING_LENGTH strlen_status,
         TC_INT* iwsg,
         TC_INT* iwse);
```

Comments: The legal values for STATUS are given in Table 6.

4.2.15 TQCSP (INDEXP, STATUS, VAL, IWSG, IWSE)

Full name: Change Status of Phase.

Purpose: With this subroutine the application program can change status for a phase.

Arguments:

Name	Type	Value set on call or returned
INDEXP	Integer	Set to a phase index.
STATUS	Character*12	Set to the status code (see Table 8).
VAL	Double precision	Set to phase amount in number of moles.
IWSG	Integer array	Workspace.
IWSE	Integer array	Workspace.

C-interface:
```
tq_csp(  TC_INT index,
         TC_STRING status,
         TC_FLOAT amount,
         TC_INT* iwsg,
         TC_INT* iwse);
```

Comments: The legal values for STATUS are given in Table 8.

For **ENTERED** phase, VAL is provided as a start value. It is normally set to zero if the phase is not likely to be stable and one if expected to be stable

Setting a phase **SUSPENDED** or **DORMANT** is a way to calculate a metastable equilibrium if the phase would be stable. With the **DORMANT** status one can know if it would be stable or not. For these two statuses, VAL is irrelevant and may be simply put to zero.

For **FIXED** phase the exact amount of the phase must be given. Please note that the amount is in number of moles of atoms, which means that the vacancy in a sublattice phase will not be included. Therefore, for a vacancy-containing sublattice phase with **FIXED** status, it is impossible to set VAL to the total number of moles in the system.

Setting a phase **FIXED** will decrease 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.

4.2.16 LOGICAL FUNCTION TQGSP (INDEXP, STATUS, VAL, IWSG, 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.

Arguments:

Name	Type	Value set on call or returned
INDEXP	Integer	Set to a phase index.
STATUS	Character*12	Return the current status code.
VAL	Double precision	Return the phase amount.
IWSG	Integer array	Workspace.
IWSE	Integer array	Workspace.

C-interface:

```
tq_gsp( TC_INT index,
        TC_STRING status,
        TC_STRING_LENGTH strlen_status,
        TC_FLOAT* amount,
        TC_INT* iwsg,
        TC_INT* iwse);
```

Comments: The legal values for STATUS are listed for Table 8.

4.2.17 TQSETR (INDEXC, INDEXP, TEMP, PRES, IWSG, IWSE)

Full name: Set Reference State.

Purpose: With this subroutine the reference state of a system component can be reset.

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.

C-interface:
```
tq_setr( TC_INT indexc,
         TC_INT indexp,
         TC_FLOAT temp,
         TC_FLOAT press,
         TC_INT* iwsg,
         TC_INT* iwse);
```

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.

4.2.18 TQPACS (INDEXP, IWSG, IWSE)

Full name: Add composition set to phase.

Purpose: With this subroutine an additional composition set may be added 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.

C-interface:
```
tq_pacs( TC_INT indexp,
         TC_INT* iwsg,
         TC_INT* iwse);
```

Comments:

4.2.19 TQSGA (INDEXP, VALUE, IWSG, IWSE)

Full name: Set Gibbs energy Addition.

Purpose: With this subroutine an amount of extra contribution can be added to the Gibbs energy of a phase

Arguments:

Name	Type	Value set on call or returned
INDEXP	Integer	Set to a phase index.
VALUE	Double precision	Set to the value of extra contribution.
IWSG	Integer array	Workspace.
IWSE	Integer array	Workspace.

C-interface:

```
tq_gga( TC_INT indexp,
        TC_FLOAT* value,
        TC_INT* iwsg,
        TC_INT* iwse);
```

Comments: The extra contribution may be due to elastic strain energy, surface energy, etc.

4.2.20 TQGGA (INDEXP, VALUE, IWSG, IWSE)

Full name: Get Gibbs energy Addition.

Purpose: With this subroutine the contribution added to the Gibbs energy of a phase can be retrieved

Arguments:

Name	Type	Value set on call or returned
INDEXP	Integer	Set to a phase index.
VALUE	Double precision	Return the value of extra contribution.
IWSG	Integer array	Workspace.
IWSE	Integer array	Workspace.

C-interface:
```
tq_gga( TC_INT indexp,
        TC_FLOAT* value,
        TC_INT* iwsg,
        TC_INT* iwse);
```

4.3 Condition, Stream, and Segment Subroutines

4.3.1 TQSETC(STAVAR, INDEXP, INDEXC, VAL, NUMCON, IWSG, IWSE)

Full name: Set Condition.

Purpose: To set conditions for an equilibrium calculation.

Arguments:

Name	Type	Value set on call or returned
STAVAR	Character*8	Set as a state variable listed in Table 10.
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 condition.
IWSG	Integer array	Workspace.
IWSE	Integer array	Workspace.

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);
```

Comments: In STAVAR the mnemonic of the state variable must be given, see Table 10. 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 will be 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 will restore one degree of freedom.

Possible combinations of STAVAR and indices are listed in Table 10.

In the table it is shown that the same value of STAVAR may be used with or without 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 will make its best to provide relevant help for such cases.

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 Al_2O_3 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 H_2O 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)
```

4.3.2 TQREMC(NUMCON, IWSG, IWSE)

Full name: Remove Condition.

Purpose: Remove the condition numbered NUMCON.

Arguments:

Name	Type	Value set on call or returned
NUMCON	Integer	Set to a condition number.
IWSG	Integer array	Workspace.
IWSE	Integer array	Workspace.

C-interface:

```
tq_remc(    TC_INT numcon,
            TC_INT* iwsg,
            TC_INT* iwse);
```

Comments: TQSETC and TQCSTM 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.

4.3.3 TQSCURC(IWSG, IWSE)

Full name: Save Current Conditions.

Purpose: Save all conditions in case they need to be restored.

Arguments:

Name	Type	Value set on call or returned
IWSG	Integer array	Workspace.
IWSE	Integer array	Workspace.

C-interface:
```
tq_scurc(   TC_INT* iwsg,
            TC_INT* iwse);
```

Comments: The saved conditions can be restored if necessary by using TQRESTC.

4.3.4 TQREMAC(IWSG, IWSE)

Full name: Remove All Conditions.

Purpose: Remove all conditions.

Arguments:

Name	Type	Value set on call or returned
IWSG	Integer array	Workspace.
IWSE	Integer array	Workspace.

C-interface:
```
tq_remac(   TC_INT* iwsg,
            TC_INT* iwse);
```

Comments: TQREMAC provides the easiest way to remove all conditions. After calling TQREMAC, one can set completely new or restore previously saved conditions.

4.3.5 TQRESTC(IWSG, IWSE)

Full name: Restore Condition.

Purpose: Restore saved conditions.

Arguments:

Name	Type	Value set on call or returned
IWSG	Integer array	Workspace.
IWSE	Integer array	Workspace.

C-interface:
```
tq_restc(TC_INT* iwsg,
         TC_INT* iwse);
```

Comments: Before calling TQRESTC, one needs to remove all present conditions.

4.3.6 TQCSTM(IDENT, TEMP, PRESS, IWSG, 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.

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.

C-interface:

```
tq_cstm( TC_STRING stream,
         TC_FLOAT temp,
         TC_FLOAT press,
         TC_INT* iwsg,
         TC_INT* iwse);
```

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.

4.3.7 TQSSC(IDENT, INDEXP, INDEXC, VALUE, NUMIN, IWSG, IWSE)

Full name: Set Stream Constituent Amount.

Purpose: Set the amount of phase constituent in a stream.

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.

C-interface:

```
tq_ssc(  TC_STRING stream,
         TC_INT iph,
         TC_INT icmp,
         TC_FLOAT value,
         TC_INT icond,
         TC_INT* iwsg,
         TC_INT* iwse);
```

Comments: The last one will take effect if the amount of the same phase constituent have been set several times, i.e., the amount can not be set additively.

4.3.8 TQSSIC(STAVAR, VALUE, IWSG, IWSE)

Full name: Set Stream Invariant State Variable.

Purpose: To specify the invariant state variable for calculating the reaction of all streams.

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.

C-interface:

```
tq_ssic( TC_STRING stavar,
         TC_FLOAT value,
         TC_INT* iwsg,
         TC_INT* iwse);
```

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.

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
```

4.3.9 TQDSTM(IDENT, IWSG, IWSE)

Full name: Delete Stream.

Purpose: Delete all or a stream.

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.

C-interface:
```
tq_dstm(TC_STRING stream,
        TC_INT* iwsg,
        TC_INT* iwse);
```

Comments: Using an empty string as IDENT will remove all the streams entered.

4.3.10 TQNSEG(ID, IWSG, IWSE)

Full name: New Equilibrium Segment.

Purpose: With this subroutine 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.

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.

C-interface:
```
tq_nseg(TC_STRING id,
        TC_INT* iwsg,
        TC_INT* iwse);
```

Comments: TQNSEG does not read any thermodynamic file. This must have already been done with TQRFIL. Note that when several segments are used, an equilibrium should be computed when a segment is selected before any data is retrieved.

4.3.11 TQSSEG(ID, IWSG, IWSE)

Full name: Select Equilibrium.

Purpose: When the application program has created several equilibrium segments using TQNSEG, this subroutine makes it possible to select a current equilibria which the subroutine calls refer to.

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.

C-interface:

```
tq_sseg(TC_STRING id,
        TC_INT* iwsg,
        TC_INT* iwse);
```

Comment: Note that when several segments are used, an equilibrium should be computed when a segment is selected before any data is retrieved.

4.4 Calculation and Results Subroutines

4.4.1 TQCE(TARGET, INDEXP, INDEXC, VALUE, IWSG, IWSE)

Full name: Calculate Equilibrium.

Purpose: Calculate the equilibrium with current settings of conditions or streams.

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.

C-interface:

```
tq_ce(  TC_STRING var,
        TC_INT indexp,
        TC_INT indexc,
        TC_FLOAT value,
        TC_INT* iwsg,
        TC_INT* iwse);
```

Comments: Some software may need a TARGET specified for certain types of calculations. A TARGET is a state variable as specified in TQSETC or Table 10. 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.

Examples: Calculate enthalpy for an equilibrium gas mixture SO_3, SO_2 and O_2. Input SO_3 2%, O_2 10% and 88% SO_2.

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

NOTE: In this way a 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.

4.4.2 TQCEG(IWSG, IWSE)

Full name: Calculate Equilibrium Global.

Purpose: Calculate Equilibrium using Global Minimization Algorithm.

Arguments:

Name	Type	Value set on call or returned
IWSG	Integer array	Workspace.
IWSE	Integer array	Workspace.

C-interface:
```
tq_ceg( TC_INT* iwsg,
        TC_INT* iwse);
```

Comments: The use of global minimization algorithm is meant to avoid metastable or unstable equilibrium and to obtain truely stable equilibrium. This is mainly due to its ability to find automatically miscibility gap and create accordingly new composition sets. As a consequence, the number of phases may increase after calling TQCEG. The newly added phases (new composition sets of old phases) are always put in the end of the phase list. In this way, the indexes of old phases remain the same as before (See Example 13). The global minimization technique starts with discretizing the composition space and calculating Gibbs energy values at each grid point for each phase at a given temperature. This usually leads to a significant increase of computation time. Therefore, it is not recommended to use TQCEG in time-critical application programs. In the cases where phases involved are well known, for example, identifying the local equilibrium at a phase interface, it is absolutely not necessary to use TQCEG. If TQCEG is needed, irrelevant phases should better be rejected in the beginning when fetching thermodynamic data from a database.

4.4.3 TQGETV(STAVAR, INDEXP, INDEXC, NUMBER, VALAR, IWSG, IWSE)

4.4.4 TQGET1(STAVAR, INDEXP, INDEXC, VAL, IWSG, IWSE)

Full name: Get Values.

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.
- amount of the system, a phase or a constituent.

With TQGETV an array of values can be returned; with TQGET1 a single value only.

Arguments:

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

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);
```

Comments: If an equilibrium is not established, the error code will be set on return. In Table 12 an extended set of state variables is given, which can be used for obtaining values.

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

Get total mass of liquid phase

```
CALL TQGPI(INDEXP,'LIQUID',IWSG,IWSE)
CALL TQGET1('MP',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 H_2O in GAS

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

Get partial pressure of H_2O 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)
```

4.4.5 DOUBLE PRECISION FUNCTION TQGMU (INDEXC, IWSG, 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.

C-interface:
```
tq_gmu( TC_INT indexc,
        TC_INT* iwsg,
        TC_INT* iwse);
```

4.4.6 DOUBLE PRECISION FUNCTION TQGGM (INDEXP, IWSG, IWSE)

Full name: Get Molar Gibbs Energy.

Purpose: This function returns the molar Gibbs energy of a phase in a faster way.

Arguments:

Name	Type	Value set on call or returned
INDEXP	Integer	Set to a phase index
IWSG	Integer array	Workspace.
IWSE	Integer array	Workspace.

C-interface:
```
tq_ggm( TC_INT indexp,
        TC_INT* iwsg,
        TC_INT* iwse);
```

4.4.7 TQGPD (INDEXP, NSUB, NSCON, SITES, YFRAC, EXTRA, IWSG, IWSE)

Full name: Get Phase Data.

Purpose: With this subroutine the application program can get data for the constituents of a phase.

Arguments:

Name	Type	Value set on call or returned
INDEXP	Integer	Set to a phase index
NSUB	Integer	Return the number of sublattices.
NSCON	Integer array	Return the number of constituents on each sublattice.
SITES	Double precision array	Return the number of sites on each sublattice.
YFRAC	Double precision array	Return the fractions of the constituents.
EXTRA	Double precision array	Return some special values (see Comments)
IWSG	Integer array	Workspace.
IWSE	Integer array	Workspace.

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);

Comments: With this subroutine the application program can determine the structure of the phase and the fraction of the constituents and other things. Note that YFRAC is constituent fraction, not mole fractions.

A substitutional phase will have NSUB equal to 1, which is identical to no sublattice. That is true for the gas phase too. The maximum number of sublattices are 10.

The constituents of a phase are numbered sequentially from 1 for the first constituent on the first sublattice, to NPCON (See TQGNPC) 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.

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 all together.

```
      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
```

4.4.8 TQGDF (IMATR, IPREC, NPH, NCOM, XMATR, XPREC, TEMP, DF, IWSG, IWSE)

Full name: Get the driving force of phase transformation

Purpose: With this subroutine the application program can get data on the driving force for a phase transformation

Arguments:

Name	Type	Value set on call or returned
IMATR	Integer	Set index of matrix phase
IPREC	Integer	Set index of precipitate phase
NPH	Integer	Set total number of phases in the system
NCOM	Integer	Set total number of components in the system
XMATR	Double precision array	Set composition (in mole-fraction) array of the matrix phase
XPREC	Double precision array	Set composition (in mole-fraction) array of the precipitate phase
TEMP	Double precision	Set temperature in Kelvin
DF	Double precision	Return driving force in J/mol
IWSG	Integer array	Workspace.
IWSE	Integer array	Workspace.

C-interface:

```
tq_gdf( TC_INT imatr,
        TC_INT iprec,
        TC_INT nph,
        TC_INT ncom,
        TC_FLOAT* xmatr,
        TC_FLOAT* xprec,
        TC_FLOAT* temp,
        TC_FLOAT* df,
        TC_INT* iwsg,
        TC_INT* iwse);
```

Comments: 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.

4.4.9 TQGDF2 (MODE, IMATR, IPREC, NIE, IIE, XMATR, TEMP, DF, XPREC, XEM, XEP, MUI, IWSG, IWSE)

Full name: Get the driving force of nucleation and local equilibrium concentration for a phase transformation under para- or ortho-equilibrium condition.

Purpose: With this subroutine the application program can 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.

Arguments:

Name	Type	Value set on call or returned
MODE	Integer	Set type of output and type of composition to use (±1, ±2, and ±3 correspond to mole fraction, weight fraction, and U-fraction respectively. Obviously, ±3 has nothing to do with para-equilibrium calculations. If negative, calculate and output only driving force data. This will save the time for equilibrium calculation when you are not interested in local equilibrium concentrations)
IMATR	Integer	Set index of matrix phase
IPREC	Integer	Set index of precipitate phase
NIE	Integer	Set number of interstitial element(s). Zero implies no para-equilibrium calculation
IIE	Integer array	Set index of interstitial element(s), only relevant for para-equilibrium condition
XMATR	Double precision array	Set composition of matrix phase. Composition type depends on MODE
TEMP	Double precision	Set temperature in Kelvin
DF	Double precision	Return driving force in J/mol
XPREC	Double precision array	Return composition of the precipitate phase at the maximu 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.

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);
```

Comments: For ortho-equilibrium calculation, 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 supercedes that of the preceding subroutine TQGDF. A demonstration of this subroutine can be found in Example 11.

4.5 Troubleshooting Subroutines

4.5.1 TQLS(IWSG, IWSE)

Full name: List Status.

Purpose: Listing status of all components, phases, and species in a system.

Arguments:	Name	Type	Value set on call or returned
	IWSG	Integer array	Workspace.
	IWSE	Integer array	Workspace.

C-interface:
```
tq_ls(  TC_INT* iwsg,
        TC_INT* iwse);
```

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.

4.5.2 TQLC(IWSG, IWSE)

Full name: List Conditions.

Purpose: Listing conditions set for the current equilibrium calculation.

Arguments:	Name	Type	Value set on call or returned
	IWSG	Integer array	Workspace.
	IWSE	Integer array	Workspace.

C-interface:
```
tq_lc(  TC_INT* iwsg,
        TC_INT* iwse);
```

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.

4.5.3 TQLE(IWSG, IWSE)

Full name: List Equilibrium.

Purpose: Listing results from the most recent equilibrium calculation. The output will depend on the package used and the listing will appear on the current output unit.

Arguments:	Name	Type	Value set on call or returned
	IWSG	Integer array	Workspace.
	IWSE	Integer array	Workspace.

C-interface: tq_le(TC_INT* iwsg,
TC_INT* iwse);

Comments: If necessary, use this subroutine to check if an equilibrium calculation is successful. It should only be used for debugging purpose.

4.5.4 TQFASV(IWSG, IWSE)

Full name: Force Automatic Start Value.

Purpose: To force automatic start-values for all phases in a single equilibrium calculation.

Arguments:

Name	Type	Value set on call or returned
IWSG	Integer array	Workspace.
IWSE	Integer array	Workspace.

C-interface: tq_fasv(TC_INT* iwsg,
TC_INT* iwse);

Comments: It is not necessary unless the calculation fails.

4.5.5 TQSDMC(INDEXP, IWSG, IWSE)

Full name: Set Default Major Constituents.

Purpose: To set the major phase constituents to the default ones defined in the thermodynamic data file.

Arguments:

Name	Type	Value set on call or returned
INDEXP	Integer	Set as a phase index
IWSG	Integer array	Workspace.
IWSE	Integer array	Workspace.

C-interface: tq_sdmc(TC_INT indexp,
TC_INT* iwsg,
TC_INT* iwse);

Comments: Major constituents in a phase can be set in the GES module of Thermo-Calc and then saved into a thermodynamic data file for the use of this interface.

4.5.6 TQSSPC(INDEXP, YF, IWSG, IWSE)

Full name: Set Start Phase Constitution.

Purpose: To set start-values for the constitution of an individual 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.

C-interface:
```
tq_sspc( TC_INT indexp,
         TC_FLOAT* yf,
         TC_INT* iwsg,
         TC_INT* iwse);
```

Comments: It is not necessary unless the calculation fails, especially when involving a miscibility gap or an ordering phase.

4.5.7 TQSSV(STAVAR, IP, IC, VALUE, IWSG, IWSE)

Full name: Set Start Variable.

Purpose: To set start-value for a state variable.

Arguments:

Name	Type	Value set on call or returned
STAVAR	Character*8	Set as a state variable listed in Table 10.
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.

C-interface:
```
tq_ssv(  TC_STRING stavar,
         TC_INT ip,
         TC_INT ic,
         TC_FLOAT value,
         TC_INT* iwsg,
         TC_INT* iwse);
```

Comments: It is not necessary unless the calculation fails.

4.5.8 TQPINI(IWSG, IWSE)

Full name: Poly-3 reINItiation.

Purpose: Reinitiate the Poly-3 workspace in Thermo-Calc kernel.

Arguments:

Name	Type	Value set on call or returned
IWSG	Integer array	Workspace.
IWSE	Integer array	Workspace.

C-interface: tq_pini(TC_INT* iwsg,
TC_INT* iwse);

Comments: Preparing for a fresh calculation.

4.5.9 TQSNL(MAXIT, ACC, YMIN, ADG, IWSG, IWSE)

Full name: Set Numerical Limits

Purpose: To set the Numerical Limits to be used inside Poly-3.

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	Specify if the calculation should be forced to converge also for the meta stable phases. Legal options are Y or N, where Y means yes and N means no. N is default.
	IWSG	Integer array	Workspace.
	IWSE	Integer array	Workspace.

Cinterface: tq_snl(TC_INT maxit,
TC_FLOAT acc,
TC_FLOAT ymin,
TC_FLOAT adg,
TC_INT* iwsg,
TC_INT* iwse);

Comments: It is not necessary unless the calculation fails.

4.5.10 TQSMNG(NGP, IWSG, 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.

Arguments:	Name	Type	Value set on call or returned
	NGP	Integer	Number of grid points.
	IWSG	Integer array	Workspace.
	IWSE	Integer array	Workspace.

C-interface: tq_smng(TC_INT ngp,
TC_INT* iwsg,
TC_INT* iwse);

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.

4.5.11 TQSECO(IPDH, ICSS, IWSG, IWSE)

Full name: Set Equilibrium Calculation Option.

Purpose: To choose equilibrium calculation options.

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.

C-interface:

Comments: TQ starts with IPDH=1 and ICSS=1 by default.

4.5.12 ST1ERR(IERR, SUBR, 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.

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.

C-interface:
```
tq_st1err(      TC_INT ierr,
                TC_STRING subr,
                TC_STRING mess);
```

Comments: The error-handling routines are those defined by SGTE for use in the thermodynamic model package. Note that the error-handling is constructed in such a way that when a subroutine detects an error it cannot handle, it should first call an ST* subroutine to set an appropriate error code and then return to the calling subroutine. In that subroutine the error code should be tested, and 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 will be passed up to a higher level where it can be corrected or ignored. The normal subroutines to use are ST2ERR to

set the error code and SG2ERR to check it. The other subroutines are less used.

NOTE: The TQ subroutines will normally not clear the error code when they are 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.

4.5.13 ST2ERR(IERR, SUBR, MESS)

Full name: Set Error Code.

Purpose: This is called when an error occurs that cannot be handled by the current program unit. The program unit should return to the calling 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.

C-interface:

```
tq_st2err(      TC_INT ierr,
                TC_STRING subr,
                TC_STRING mess);
```

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.

4.5.14 LOGICAL FUNCTION SG1ERR or TQG1ERR(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 will also print the error message on the error unit.

Arguments:

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

C-interface:

```
tq_sg1err(      TC_INT* ierr);
```

Comments: This should be used 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.

4.5.15 LOGICAL FUNCTION SG2ERR or TQG2ERR(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 will not print the error message.

Arguments:

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

C-interface: tq_sg2err(TC_INT* ierr);

Comments: This should be used 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.

Examples:

```
      LOGICAL SG2ERR
      ...
      CALL TQCE(' ',IWSG,IWSE)
      IF(SG2ERR(IERR)) GOTO 900
      ...
900   RETURN
```

4.5.16 LOGICAL FUNCTION SG3ERR or TQG3ERR(IERR, SUBR, 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.

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.

C-interface: tq_sg3err(TC_INT* ierr,
TC_STRING subr,
TC_STRING_LENGTH strlen_subr,
TC_STRING mess,
TC_STRING_LENGTH strlen_mess);

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.

4.5.17 RESERR (or TQRSERR)

Full name: Reset Error Code and Message

Purpose: This subroutine will reset the error code. A subsequent call to the SG* functions will give no error.

Arguments: None

C-interface: tq_reserr( );

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 will continue to report the same error.

4.5.18 TQSP3F(FILE, IWSG, 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.

C-interface:
```
tq_sp3f( TC_STRING filename,
         TC_INT* iwsg,
         TC_INT* iwse);
```

Comments:

4.6 Extra Subroutines

4.6.1 TQGMA(INDEXP, TP, YF, VAL, IWSG, IWSE)

4.6.2 TQGMB(INDEXP, TP, VAL, IWSG, IWSE)

4.6.3 TQGMC(INDEXP, VAL, IWSG, 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 below in this Section.

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.

C-interface:

```
tq_gma( TC_INT indexp,
        TC_FLOAT* tp,
        TC_FLOAT* yf,
        TC_FLOAT* val,
        TC_INT* iwsg,
        TC_INT* iwse);

tq_gmb( TC_INT indexp,
        TC_FLOAT* tp,
        TC_FLOAT* val,
        TC_INT* iwsg,
        TC_INT* iwse);

tq_gmc( TC_INT indexp,
        TC_FLOAT* val,
        TC_INT* iwsg,
        TC_INT* iwse);
```

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.

4.6.4 TQGMDY(INDEXP, VARR, IWSG, 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 below in this Section.

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

C-interface:

```
tq_gmdy(    TC_INT indexp,
            TC_FLOAT* varr,
            TC_INT* iwsg,
            TC_INT* iwse);
```

Comments: The returned value is in J/mole of formula unit. Remember this subroutine requires no action of setting condition and calculating equilibrium. It is for getting the Gibbs energy and its 1st partial derivative w.r.t site fractions for a single phase with given temperature, pressure and atomic arrangements no matter if the phase is stable, metastable, or unstable in competition with other phases. The application program should take care of the phase stability or phase equilibrium if it is of interest. This subtrouine is demonstrated in Example 9.

4.6.5 TQGMOB(INDEXP, ISP, VAL, IWSG, 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 below in this Section.

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.

C-interface:

```
tq_gmob(    TC_INT indexp,
            TC_INT isp,
            TC_FLOAT* val,
            TC_INT* iwsg,
            TC_INT* iwse);
```

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.

4.6.6 TQSTP(TP, IWSG, IWSE)

Full name: Set Temperature and Pressure

Purpose: Setting temperature and pressure.

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.

C-interface:
```
tq_stp( TC_FLOAT* tp,
        TC_INT* iwsg,
        TC_INT* iwse);
```

Comments: This subroutine is used before calling TQGMC, TQGMDY, TQDGYY, and TQGMOB. See Example 9.

4.6.7 TQSYF(INDEXP, YF, IWSG, IWSE)

Full name: Set Site Fractions

Purpose: Setting site fractions for a phase.

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.

C-interface:
```
tq_syf( TC_INT indexp,
        TC_FLOAT* yf,
        TC_INT* iwsg,
        TC_INT* iwse);
```

Comments: This subroutine is used before calling TQGMB, TQGMC, TQGMDY, TQDGYY, and TQGMOB. See Example 9.

4.6.8 TQGSSPI(SPN, ISP, IWSG, IWSE)

Full name: Get System Species Index

Purpose: Getting index of a system species with given name.

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.

C-interface: tq_gsspi(TC_STRING name,
TC_INT* index,
TC_INT* iwsg,
TC_INT* iwse);

Comments: Useful if one want to use TQGMOB. See Example 9.

4.6.9 TQCMOBA(INDEXP, ISP, IWSG, IWSE)

4.6.10 TQCMOBB(INDEXP, IWSG, IWSE)

Full name: Check if Mobility data available – Method A and B

Purpose: Check if mobility data have been appended into thermodynamic data file.

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.

C-interface: tq_cmoba(TC_INT indexp,
TC_INT* iwsg,
TC_INT* iwse);

tq_cmobb(TC_INT indexp,
TC_INT* iwsg,
TC_INT* iwse);

Comments: No comments.

4.6.11 TQDGYY(INDEXP, VARR1, VARR2, IWSG, 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.

Arguments:

Name	Type	Value set on call or returned
INDEXP	Integer	Set to a phase index.
VARR1	Double precision array	Return values of Gibbs energy and its 1st

			partial derivatives w.r.t. site fractions in the index order of phase constituents.
	VARR2	Double precision array	Return values of 2nd partial derivatives of Gibbs energy w.r.t. site fractions in the index order of IR: IR=J+I*(I-1)/2, I>=J; IR=I+J*(J-1)/2, I<J, where I and J are row and column indexes of phase constituents, respectively.
	IWSG	Integer array	Workspace.
	IWSE	Integer array	Workspace.

C-interface:
```
tq_dgyy(TC_INT indexp,
        TC_FLOAT* varr1,
        TC_FLOAT* varr2,
        TC_INT* iwsg,
        TC_INT* iwse);
```

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.

4.6.12 TQGPHP(INDEXP, NE, NCNV, NC, IWORK, WORK, IWSG, 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.

Arguments:	Name	Type	Value set on call or returned
	INDEXP	Integer	Set to a phase index.
	NE	Integer	Return number of components
	NCNV	Integer	Return number of constituents without counting vacancies
	NC	Integer	Return number of constituents
	IWORK	Integer array	Return values needed in X to Y conversion, array size >= 4*NCNV
	WORK	Double precision array	Return values needed in X to Y conversion, array size >= (NE+1)*NCNV
	IWSG	Integer array	Workspace.
	IWSE	Integer array	Workspace.

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);
```

Comments: This subroutine is designed to spead up conversions of quantities involving mole fractions and site fractions in dynamic calculaitions 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 converning this phases. See Example 10.

4.6.13 TQX2Y(INDEXP, NE, NCNV, NC, IWORK, WORK, XF, YF, IWSG, IWSE)

Full name: Get Y-fraction given X-fraction.

Purpose: Converting mole fractions to site fractions in a phase without internal degree of freedom.

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.

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);
```

Comments: This subroutine uses the phase constitution properties obtained by TQGPHP as input. See Example 10.

4.6.14 TQGMDX(IP, NE, NCNV, NC, IWORK, WORK, YF, VARR, GM, DGDX, XF, IWSG, 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.

Arguments:

Name	Type	Value set on call or returned
IP	Integer	Set to a phase index.
NE	Integer	Set to number of components
NCNV	Integer	Set to number of constituents without counting vacancies
NC	Integer	Set to number of constituents
IWORK	Integer array	Set to values needed in X to Y conversion
WORK	Double precision array	Set to values needed in X to Y conversion

YF	Double precision array	Set to site fractions
VARR	Double precision array	Set to Gibbs energy and its first derivative with respect to site fractions
GM	Double precision	Return Gibbs energy
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.

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);
```

Comments: This subroutine uses the phase constitution properties obtained by TQGPHP as input. Note VARR obtained by calling TQGMDY is in unit of J/mole of formula unit. GM and DGDX in the present subroutine is in unit of J/mole of atoms. For the use of this subroutine together with TQGPHP and TQX2Y, please see Example 10.

4.7 Database Subroutines (See Example 12)

4.7.1 TQGDBN(DB_ARR, N, IWSG, IWSE)

Full name: Get DataBase Names and Number.

Purpose: Get the names and total number of thermodynamic and kinetic databases listed in the database initiation file of Thermo-Calc: TC_INITD.TDB.

Arguments:

Name	Type	Value set on call or returned
DB_ARR	Character*24 array	Return database names.
N	Interger	Return total number of databases available.
IWSG	Integer array	Workspace.
IWSE	Integer array	Workspace.

C-interface:

```
tq_gdbn(    tc_databases_strings* databases,
            TC_INT* n,
            TC_INT* iwsg,
            TC_INT* iwse);
```

Comments: IERR = 1001 Failled to find the initiation file.

4.7.2 TQOPDB(TDB, IWSG, IWSE)

Full name: Open DataBase.

Purpose: Open a thermodynamic or kinetic database.

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.

C-interface:

```
tq_opdb(    TC_STRING database,
            TC_INT* iwsg,
            TC_INT* iwse);
```

Comments: IERR = 1001 Failled to find the initiation file.
IERR = 1002 Database or its license not available.
.

4.7.3 TQLIDE(EL_ARR, N, IWSG, 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.

C-interface: tq_lide(tc_elements_strings* elements,
TC_INT* num,
TC_INT* iwsg,
TC_INT* iwse);

Comments:

4.7.4 TQAPDB(TDB, IWSG, IWSE)

Full name: Append DataBase.

Purpose: Append a thermodynamic or kinetic database.

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.

C-interface: tq_apdb(TC_STRING database,
TC_INT* iwsg,
TC_INT* iwse);

Comments: IERR = 1002 Database or its license not available.

4.7.5 TQDEFEL(ELNAM, IWSG, IWSE)

Full name: Define Element.

Purpose: Define a system element.

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.

C-interface: tq_defel(TC_STRING element,
TC_INT* iwsg,
TC_INT* iwse);

Comments: IERR = 1011 Element not included in the chosen database.
IERR = 1012 Element already defined.

4.7.6 TQREJEL(ELNAM, IWSG, IWSE)

Full name: Reject Element.

Purpose: Reject a defined system element.

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.

C-interface:
```
tq_rejel( TC_STRING element,
          TC_INT* iwsg,
          TC_INT* iwse);
```

Comments:
IERR = 1013 Element not included in the chosen database.
IERR = 1014 Element already rejected.

4.7.7 TQRESPH(PHNAM, IWSG, IWSE)

Full name: Restore Phase.

Purpose: Restore a rejected system phase.

Arguments:

Name	Type	Value set on call or returned
PHNAM	Character*24	Set to a phase name.
IWSG	Integer array	Workspace.
IWSE	Integer array	Workspace.

C-interface:
```
tq_resph(   TC_STRING phase,
            TC_INT* iwsg,
            TC_INT* iwse);
```

Comments:
IERR = 1015 Phase not included in the chosen database.
IERR = 1016 Phase already restored.

4.7.8 TQREJPH(PHNAM, IWSG, IWSE)

Full name: Reject Phase.

Purpose: Reject a system phase.

Arguments:

Name	Type	Value set on call or returned
PHNAM	Character*24	Set to a phase name.
IWSG	Integer array	Workspace.
IWSE	Integer array	Workspace.

C-interface:
```
tq_rejph(   TC_STRING phase,
            TC_INT* iwsg,
            TC_INT* iwse);
```

Comments:
IERR = 1017 Phase not included in the chosen database.
IERR = 1018 Phase already rejected.

4.7.9 TQLISPH(PH_ARR, N, IWSG, 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.

C-interface:

```
tq_lisph(tc_phases_strings* phases,
         TC_INT* num,
         TC_INT* iwsg,
         TC_INT* iwse);
```

Comments:

4.7.10 TQLISSF(PH_ARR, N, IWSG, 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.

C-interface:

```
tq_lissf( tc_phases_strings* phases,
          TC_INT* num,
          TC_INT* iwsg,
          TC_INT* iwse);
```

Comments:

4.7.11 TQGDAT(IWSG, 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.

C-interface:

```
tq_gdat( TC_INT* iwsg,
         TC_INT* iwse);
```

Comments:

4.7.12 TQREJS(IWSG, 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.

Arguments:

Name	Type	Value set on call or returned
IWSG	Integer array	Workspace.
IWSE	Integer array	Workspace.

C-interface:

```
tq_rejsy(TC_INT* iwsg,
         TC_INT* iwse);
```

Comments: In any application programs, either TQINI or TQINI3 (see 4.1.0 and 4.1.1) 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.

4.8 Interpolation scheme

4.8.1 TQIPS_INIT_TOP(NELC, NPHC, PHASESTR, IERR, IWSG, 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.

Arguments:

Name	Type	Value set on call or returned
IERR	Integer	Returns the error code.
IWSG	Integer array	Workspace.
IWSE	Integer array	Workspace.

C-interface:

```
tq_ips_init_top( TC_INT* err,
                 TC_INT* iwsg,
                 TC_INT* iwse)
```

Comments: IERR is returned with "0" if no error occurs.

4.8.2 TQIPS_INIT_BRANCH(TISCOND, TISCNST, PISCOND, PISCNST, IDEPEL, NSTEP, IPHSTA, T, TMIN, TMAX, P, PMIN, PMAX, RMEMFR, PHAMNT, XMIN, XMAX, IBRANCH, IERR IWSG, 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.

Arguments:

Name	Type	Value set on call or returned
TISCOND	Logical	Set to TRUE if temperature is a condition in this branch.
TISCNST	Logical	Set to TRUE if temperature is constant in this branch.
PISCOND	Logical	Set to TRUE if pressure is a condition in this branch.
PISCNST	Logical	Set to TRUE if pressure is constant in this branch.
IDEPEL	Logical array	Set to TRUE for each element for which conditions are present.
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 interpolation scheme for this 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	Returns the branch number
IERR	Integer	Returns error code.
IWSG	Integer array	Workspace.
IWSE	Integer array	Workspace.

C-interface:

```
tq_ips_init_branch(     TC_BOOL t_is_condition,
                        TC_BOOL t_is_constant,
                        TC_BOOL p_is_condition,
                        TC_BOOL p_is_constant,
                        TC_BOOL* independent_elements,
                        TC_INT dicretization_type,
                        TC_INT nr_of_steps,
                        TC_INT* state_of_phases,
                        TC_FLOAT t,
                        TC_FLOAT tmin,
                        TC_FLOAT tmax,
                        TC_FLOAT p,
                        TC_FLOAT pmin,
                        TC_FLOAT pmax,
                        TC_FLOAT memory_fraction,
                        TC_FLOAT* amount_of_phases,
                        TC_FLOAT* xmin,
                        TC_FLOAT* xmax,
                        TC_INT* branch_nr,
                        TC_INT* err,
                         TC_INT* iwsg,
                        TC_INT* iwse)
```

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.

4.8.3 TQIPS_INIT_FUNCTION(STRING, IBRANCH, IERR, IWSG, IWSE)

Full name: Initiates a function for a specific branch whose value(s) are to be retreived from the adaptive interpolation scheme.

Purpose: Initiates a function or state variable for a specific branch whose value(s) are to be retreived 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 statevariable 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.

C-interface:

```
tq_ips_init_function(   TC_STRING function_string,
                        TC_INT branch_nr,
                        TC_INT* err,
                        TC_INT* iwsg,
                        TC_INT* iwse);
```

Comments:

4.8.4 TQIPS_GET_VALUE(IBRANCH, NOSCHEME, ARR, RESULT, IERR, ISHORT, IWSG, 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 where 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.

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);

Comments:

4.9 Composition set reordering routines

4.9.1 TQROINIT(NWSR, IWSR, IWSG, 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.

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.

C-interface:

Comments: NWSR=1000 should be enough for several composition sets

4.9.2 TQSETRX(PHASE, X, IWSR, IWSG, IWSE)

Full name: Set ideal composition in this phase

Purpose: Store composition of phase in IWSR for future use.

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.

C-interface:

Comments: The order in the X array is the order of the components in the system

4.9.3 TQORDER(IWSR, IWSG, IWSE)

Full name: ReOrder CS in current EQ

Purpose: With this subroutine the ideal composition set by the user is used to reorder the CS in respective phase to minimize the distance compared to present eq.

Arguments:	Name	Type	Value set on call or returned
	IWSR	Integer array	Workspace.
	IWSG	Integer array	Workspace.
	IWSE	Integer array	Workspace.

C-interface:

Comments: Calling routine more than once in a row should affect nothing. Routine will minimize the distance between the set ideal compositnion and the composition found in the present equilibria, and reorder the CS in the equillibria to achive the minima. This do not affekt the properties of the equilibria.

4.9.4 TQLROX(IWSR, IWSG, IWSE)

Full name: List content of IWSR set by user.

Purpose: With this subroutine the ideal composition set by the user using TQSETRX in IWSR is listed in the output unit.

Arguments:

Name	Type	Value set on call or returned
IWSR	Integer array	Workspace.
IWSG	Integer array	Workspace.
IWSE	Integer array	Workspace.

C-interface:

Comments: For debuging informing the user..

6. Appendix 1 Compiler settings

6.1 Compiling fortran code

6.1.1 Windows

6.1.1.1 Visual Studio 2010, Intel Fortran Composer 12

6.1.1.1.1 32-bit configuration

Compiler flags:

/iface:default

Example:

ifort /iface:default /c tqex01.F
ifort/exe:tqex01.exe tqex01.obj libtq-win-Win32-8-beta1.lib

6.1.1.1.2 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-x64-8-beta1.lib

6.1.1.2 Visual C++ 6.0, Compaq Visual Fortran 6.6C (not supported)

6.1.1.1.2.1 32-bit configuration

Compiler flags:

/iface:nomixed_str_len_arg
/iface:cref

Example:

df /iface:nomixed_str_len_arg /iface:cref /compile_only tqex01.F
df /exe:tqex01.exe tqex01.obj libtq-win-Win32-8-beta1.lib

6.1.2 Linux

6.1.2.1 GNU compiler version 4.4

6.1.2.1.1 32-bit configuration

Compiler flags:
None

Example:
```
gfortran44 –c tqex01.F
gfortran44 –o tqex01 tqex01.o libtq-linux-ia32-gfortran44-8-beta1.so
```

6.1.2.1.2 64-bit configuration

Compiler flags:
-fdefault-real-8
-fdefault-double-8
-fdefault-integer-8

Example:
```
gfortran44 –c    -fdefault-real-8 -fdefault-double-8 \
                 -fdefault-integer-8  tqex01.F
gfortran44 –o tqex01 tqex01.o libtq-linux-x86_64-gfortran44-8-beta1.so
```

6.1.2.2 Intel fortran compiler

6.1.2.2.1 32-bit configuration

Compiler flags:
None

Example:
```
ifort –c tqex01.F
ifort –o tqex01 tqex01.o libtq-linux-ia32-ifort-8-beta1.so
```

6.1.2.2.2 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-8-beta1.so
```

6.2 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.

6.2.1 Windows

6.2.1.1 Visual Studio 2010

NOTE: 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.

6.2.1.1.1 32-bit configuration

Compiler flags:

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

Example:

```
cl /c /DWIN32 /I..\tq\C\include tqex01.c
link /OUT:tqex01.exe tqex01.obj libtq-win-Win32-8-beta1.lib
```

6.2.1.1.2 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-x64-8-beta1.lib
```

6.2.1.2 Visual Studio 6

6.2.1.1.2.1 32-bit configuration

Compiler flags:

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

Example:

```
cl /c /DWIN32 /I..\tq\C\include tqex01.c
link /OUT:tqex01.exe tqex01.obj libtq-win-Win32-8-beta1.lib
```

6.2.2 Linux

6.2.2.1 GNU compiler version 4.4

6.2.2.1.1 32-bit configuration

Compiler flags:

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

Example:

```
gcc44 –c -I../tq/C/include tqex01.c
gcc44 –o tqex01 tqex01.o libtq-linux-ia32-gfortran44-8-beta1.so
```

6.2.2.1.2 64-bit configuration

Compiler flags:

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

Example:

```
gcc44 –c -I../tq/C/include tqex01.c
gcc44 –o tqex01 tqex01.o libtq-linux-x86_64-gfortran44-8-beta1.so
```