Programming Languages

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

FORTRAN

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

C-Interface

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

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

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

The definition of some of these data types vary depending on what platform and compiler is used. It is important to define these and for the definitions to be correctly set (see "Compiler Settings" on page 145).

Common C-Procedure Definitions

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

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

Initialization Subroutines

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

TQINI

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

TQSIO

Fortran	**TQSIO(OPTION, IVAL)**	
C-interface	**tq_sio(TC_STRING option,TC_INT ival);**	
Full name:	Set Input/Output Option.	
Purpose:	With this subroutine the application program can re-direct input and output from the Thermo-Calc package.	
Comments:	OPTION is a character identifying the Input/Output option. The current internal value is set to the value in IVAL. If the value is illegal the error condition is set.	
Arguments		
Name	**Type**	**Value set on call or returned**
OPTION	Character*8	Set to a value given in "Legal Input/Output Options for TQSIO and TQGIO" on page 17
IVAL	Integer	Set to an internal value.

TQGIO

Fortran	**TQGIO(OPTION, IVAL)**	
C-interface	**tq_gio(TC_STRING option,TC_INT* ival);**	
Full name:	Get Input/output Unit.	
Purpose:	Obtain a value of Input/Output option.	
Comments:	OPTION is a character identifying the Input/Output option. IVAL is an integer where its current internal value is returned.	
Arguments		
Name	**Type**	**Value set on call or returned**
OPTION	Character*8	Set to a value given in "Legal Input/Output Options for TQSIO and TQGIO" on page 17.
IVAL	Integer	Return the current internal value.

TQRFIL

Fortran	**TQRFIL(FILE, IWSG, IWSE)**
C-interface	**tq_rfil(TC_STRING file,TC_INT* iwsg,TC_INT* iwse);**
Full name:	Read File.
Purpose:	Read a thermodynamic data file in the Thermo-Calc format.
Comments:	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 Thermo-Calc and then stored in a GES file by using the save command in the Gibbs-Energy-System module inside Thermo-Calc. When the GES file is read into the workspace by this subroutine it is possible to manipulate data by changing components and status for components or phases.

Arguments

Name	Type	Value set on call or returned
FILE	Character*60	Legal file name
IWSG	Integer array	Workspace
IWSE	Integer array	Workspace

TQSSU

Fortran	**TQSSU(QUANT, UNIT, IWSG, IWSE)**	
C-interface	**tq_ssu(TC_STRING quant,TC_STRING unit,TC_INT* iwsg,TC_INT* iwse);**	
Full name:	Set System Unit.	
Purpose:	Set the unit for a quantity (like mass, volume, etc.).	
Comments:	Default units are SI unless changes are made by this subroutine. The legal quantities and units are listed in "Units for TQSSU and TQGSU" on page 16.	
Arguments		
Name	**Type**	**Value set on call or returned**
QUANT	Character*60	Set to a legal quantity
UNIT	Character*60	Set to a legal unit
IWSG	Integer array	Workspace
IWSE	Integer array	Workspace

TQGSU

Fortran	**TQGSU(QUANT, UNIT, IWSG, IWSE)**	
C-interface	**tq_gsu(TC_STRING quant,TC_STRING unit,TC_STRING_LENGTH strlen_unit,TC_INT* iwsg,TC_INT* iwse);**	
Full name:	Get System Unit.	
Purpose:	To find what units the TQ-Interface is currently using for a system quantity.	
Comments:	The legal quantities and units are listed in "Units for TQSSU and TQGSU" on page 16.	
Arguments		
Name	**Type**	**Value set on call or returned**
QUANT	Character*60	Set to a legal quantit.
UNIT	Character*60	Return the current unit.
IWSG	Integer array	Workspace
IWSE	Integer array	Workspace

TQSAME

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

TQGVER

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

TQSCOM

Fortran	**TQSCOM(NCOM, NAMES, STOI, IWSG, IWSE)**
C-interface	**tq_scom(TC_INT num,tc_components_strings* components,TC_FLOAT* stoi,TC_INT* iwsg,TC_INT* iwse);**
Full name:	Set System Component.
Purpose:	A new set of system components can be defined. The new number of components must be the same as previously. The number of system components can be changed by suspending a component by "TQCSSC" on page 43.

Comments:	• The set of components must be linearly independent.
	• The names of the new system components are given in NAMES.
	• STOI is a matrix with dimension STOI (1:NCOM,1:NCOM) which gives the stoichiometry of the new components expressed in the old ones.
	• The default set for components is taken from in the input thermodynamic data file.
	• Legal values for the array elements in NAMES are constituent names.
	• The components are numbered as 1 NCOM in the order they are supplied in this call. The conversion from component name to index is also done by "TQGSCI" on page 34.
	
	Example
	
	For example, to transform from the components A, B, C to `A2B`, `B4CC2` use the following values of STOI:
	
	`A2B2.0 1.0 0.0`
	`B4C 0.0 4.0 1.0`
	`C2 0.0 0.0 2.0`

Arguments

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

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

TQGCOM

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

TQGSCI

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

TQGNP

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

TQGNPC

Fortran	**TQGNPC(INDEXP, NPCON, IWSG, IWSE)**	
C-interface	**tq_gnpc(TC_INT indexp,TC_INT* npcon,TC_INT* iwsg,TC_INT* iwse);**	
Full name:	Get Number of Phase Constituent.	
Purpose:	With this subroutine the number of constituents in a phase can be obtained.	
Comments:	To have also the names, fractions etc. of the constituents, use "TQGPD" on page 78.	
Arguments		
Name	**Type**	**Value set on call or returned**
INDEXP	Integer	Set to a phase index.
NPCON	Integer	Return the number of the phase constituents.
IWSG	Integer array	Workspace
IWSE	Integer array	Workspace

TQCSSC

Fortran	TQCSSC (INDEXC, STATUS, IWSG, IWSE)	
C-interface	tq_cssc(TC_INT index,TC_STRING status,TC_INT* iwsg,TC_INT* iwse);	
Full name:	Change Status of System Component	
Purpose:	With this subroutine the application program can change status for a system component.	
Comments:	The legal values for STATUS are ENTERED, SUSPENDED and SPECIAL "Legal Component Status" on page 28 By suspending a system component some phases may also become suspended if they contain this component. For example, in the system Fe-O-S if O is suspended all phases that must dissolve oxygen is automatically suspended. The fraction of oxygen is set to zero in phases that can dissolve oxygen but can also exist without oxygen.	
Arguments		
Name	**Type**	**Value set on call or returned**
INDEXC	Integer	Set to a component index.
STATUS	Character*12	Set to the new status
IWSG	Integer array	Workspace
IWSE	Integer array	Workspace

TQGSSC

This is a logical function.

Fortran	**STATUS=TQGSSC (INDEXC, IWSG, IWSE)**	
C-interface	**status=tq_gssc(TC_INT index,TC_INT* iwsg,TC_INT* iwse);**	
Full name:	Get Status of System Component.	
Purpose:	This function returns TRUE if the system component is ENTERED or FALSE if it is SUSPENDED.	
Comments:	The legal values for STATUS are given in "Legal Component Status" on page 28.If the C-interface is used the value returned is of type: TC_BOOL.	
Arguments		
Name	**Type**	**Value set on call or returned**
INDEXC	Integer	Set to a component index.
IWSG	Integer array	Workspace
IWSE	Integer array	Workspace

TQCSP

Fortran	**TQCSP (INDEXP, STATUS, VAL, IWSG, IWSE)**	
C-interface	**tq_csp(TC_INT index,TC_STRING status,TC_FLOAT amount,TC_INT* iwsg,TC_INT* iwse);**	
Full name:	Change Status of Phase.	
Purpose:	Change status for a phase.	
Comments:	The legal values for STATUS are given in "Legal Phase Status" on page 29. For ENTERED phase, VAL is provided as a start value. It is normally set to zero if the phase is not likely to be stable and one if expected to be stable. Setting a phase SUSPENDED or DORMANT is a way to calculate a metastable equilibrium if the phase would be stable. With the DORMANT status one can know if it would be stable or not. For these two statuses, VAL is irrelevant and may be simply put to zero. For FIXED phase the exact amount of the phase must be given. Note that the amount is in number of mole formula units. Setting a phase FIXED decreases the degrees of freedom in the system by 1. To restore the lost degree of freedom the phase should be reset ENTERED. Set a FIXED phase to zero amount is the best way to get the phase stability limits like liquidus or solidus.	
Arguments		
Name	**Type**	**Value set on call or returned**
INDEXP	Integer	Set to a phase index.
STATUS	Character*12	Set to the status code
VAL	Double precision	Set to phase amount in number of mole formula units.
IWSG	Integer array	Workspace
IWSE	Integer array	Workspace

Possible State Variables to Set Conditions in TQSETC

STAVAR	INDEXP	INDEXC	Meaning	Comments
T			Temperature	of the whole system
P			Pressure	of the whole system
MU	note [1.]	Yes	Chemical potential	of a system component
MUC	Yes	Yes	Chemical potential	of a phase constituent
AC	note[1]	Yes	Activity	of a system component
ACC	Yes	Yes	Activity	of a phase constituent
V			Volume	of the whole system
G			Gibbs energy	of the whole system
H			Enthalpy	of the whole system
S			Entropy	of the whole system
N			Moles	of all system components
N		Yes	Moles	of a system component
NP	note [2.]		Moles	of a phase
M			Total mass	of all system components
M		Yes	Mass	of a system component
BP	note[2]		Mass	of a phase
IN	Yes	Yes	Input amount	in moles of phase constituents
IM	Yes	Yes	Input amount	in mass units of phase constituents
X		Yes	Mole fraction	of a system component
W		Yes	Mass (Weight) fraction	of a system component
X%		Yes	Mole percent	of a system component
W%		Yes	Mass (Weight) fraction	of a system component

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

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

TQREMC

Fortran	**TQREMC(NUMCON, IWSG, IWSE)**	
C-interface	**tq_remc(TC_INT numcon,TC_INT* iwsg,TC_INT* iwse);**	
Full name:	Remove Condition.	
Purpose:	Remove the condition numbered NUMCON.	
Comments:	"TQSETC" on page 53 and "TQCSTM" on page 59 return an index for each condition set. This value must be supplied in this call. In order to change a condition to something else, not just a new value, one must first remove the condition. If one just wants to change the value of a condition one may call TQSETC again instead.	
Arguments		
Name	**Type**	**Value set on call or returned**
NUMCON	Integer	Set to a condition number.
IWSG	Integer array	Workspace
IWSE	Integer array	Workspace

TQCSTM

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

TQNSEG

Fortran	**TQNSEG(ID, IWSG, IWSE)**	
C-interface	**tq_nseg(TC_STRING id, TC_INT* iwsg,TC_INT* iwse);**	
Full name:	New Equilibrium Segment.	
Purpose:	With this subroutine the application program can create a new equilibrium description with the same thermodynamic data. This subroutine is useful when simulating several equilibria representing local conditions, for example, in the reactor simulator.	
Comments:	TQNSEG does not read any thermodynamic file. This must have already been done with "TQRFIL" on page 22. Note that when several segments are used, an equilibrium should be computed when a segment is selected before any data is retrieved.	
Arguments		
Name	**Type**	**Value set on call or returned**
ID	Character*24	Set as identifier of the equilibrium segment.
IWSG	Integer array	Workspace
IWSE	Integer array	Workspace

TQSSEG

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

TQGPD

Fortran	**TQGPD (INDEXP, NSUB, NSCON, SITES, YFRAC, EXTRA, IWSG, IWSE)**
C-interface	**tq_gpd(TC_INT indexp,TC_INT* nsub,TC_INT* nscon,TC_FLOAT* sites,TC_FLOAT* yfrac,TC_FLOAT* extra,TC_INT* iwsg,TC_INT* iwse);**
Full name:	Get Phase Data.
Purpose:	The application program can get data for the constituents of a phase.
Comments:	With this subroutine the application program can determine the structure of the phase and the fraction of the constituents and other things. Note that YFRAC is constituent fraction, not mole fractions.A substitutional phase has NSUB equal to 1, which is identical to no sublattice. That is true for the gas phase too. The maximum number of sublattices are 10. The constituents of a phase are numbered sequentially from 1 for the first constituent on the first sublattice, to NPCON (See "TQGNPC" on page 42) for the last constituent on the last sublattice. NSCON (L) is the number of constituents on sublattice L. The sum of NSCON over all sublattices is equal to NPCON. Note that constituents that are DORMANT and SUSPENDED still are counted in NPCON and NSCON. They also have a fraction in YFRAC (which must be zero of course). EXTRA may contain extra information about the phase, total mass for example. These are yet to be defined.

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)

Miscellaneous Subroutines

Purpose	Subroutine
List status	"TQLS" on the next page
List conditions	"TQLC" on page 86
List equilibrium	"TQLE" on page 87
Force automatic start values	"TQFASV" on page 88
Keep composition set numbers	"TQKEEP_CS_NUMBERS" on page 89
Set default major constituent	"TQSDMC" on page 90
Set start phase constitution	"TQSSPC" on page 91
Set start value of a state variable	"TQSSV" on page 92
Reinitiate the calculation workspace	"TQPINI" on page 93
Set numerical limits	"TQSNL" on page 94
Set maximum number of grid points	"TQSMNG" on page 95
Set equilibrium calculation options	"TQSECO" on page 96
Set error code and give message	"ST1ERR" on page 97
Set error code	"ST2ERR" on page 98
Get error code and give message	"SG1ERR or TQG1ERR" on page 99 *
Get error code	"SG2ERR or TQG2ERR" on page 100 *
Get error code and message	"SG3ERR or TQG3ERR" on page 101 *
Reset error code and message	"RESERR or TQRSERR" on page 102
Save a POLY-3 file	"TQSP3F" on page 103
	* Logical function

TQLS

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

TQSSV

Fortran	**TQSSV(STAVAR, IP, IC, VALUE, IWSG, IWSE)**	
C-interface	**tq_ssv(TC_STRING stavar,TC_INT ip,TC_INT ic,TC_FLOAT value,TC_INT* iwsg,TC_INT* iwse);**	
Full name:	Set Start Variable.	
Purpose:	To set start-value for a state variable.	
Comments:	It is not necessary unless the calculation fails.	
Arguments		
Name	**Type**	**Value set on call or returned**
STAVAR	Character*8	Set as a state variable listed in "Possible State Variables to Set Conditions in TQSETC" on page 52
IP	Integer	Set as a phase index (if needed).
IC	Integer	Set as a component or constituent index (if needed).
VALUE	Double precision	Set to the value.
IWSG	Integer array	Workspace
IWSE	Integer array	Workspace

TQPINI

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

TQSNL

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

TQSMNG

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

TQSECO

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

ST1ERR

Fortran	**ST1ERR(IERR, SUBR, MESS)**	
C-interface	**tq_st1err(TC_INT ierr,TC_STRING subr,TC_STRING mess);**	
Full name:	Set Error Code and Give Message.	
Purpose:	This is called when an error that cannot be handled by the current program unit occurs. The error message is printed on the error unit but also saved internally in the error handling package. The program unit should return to the calling program.	
Comments:	The error-handling routines are those defined by SGTE for use in the thermodynamic model package. Note that the error-handling is constructed in such a way that when a subroutine detects an error it cannot handle, it should first call an ST* subroutine to set an appropriate error code and then return to the calling subroutine. In that subroutine the error code should be tested, and possibly that subroutine can correct the error and proceed, otherwise it should return to its calling subroutine and so on, until either the error is corrected or the top level of the program is reached.In this way it is possible to design a program where minor problems at a low level do not cause program to terminate. Instead, the error is passed up to a higher level where it can be corrected or ignored. The normal subroutines to use are ST2ERR to set the error code and SG2ERR to check it. The other subroutines are less used. The TQ subroutines normally do not clear the error code when called. An error set in an earlier subroutine but not tested and detected after that call may cause strange error messages later on. This should be used only for fatal or almost fatal errors.	
Arguments		
Name	**Type**	**Value set on call or returned**
IERR	Integer	Set to an error code.
SUBR	Character*6	Set to the current subroutine name.
MESS	Character*72	Set to the error message to be printed

ST2ERR

Fortran	**ST2ERR(IERR, SUBR, MESS)**	
C-interface	**tq_st2err(TC_INT ierr,TC_STRING subr,TC_STRING mess);**	
Full name:	Set Error Code.	
Purpose:	Called when an error occurs that cannot be handled by the current program unit. The program unit should return to the calling program.	
Comments:	Identical to ST1ERR except that it is silent, i.e., no error message is printed. This should be the normal subroutine to call when detecting errors that should be handled by a higher level of the program.	
Arguments		
Name	**Type**	**Value set on call or returned**
IERR	Integer	Set to an error code.
SUBR	Character*6	Set to the current subroutine name.
MESS	Character*72	Set to the error message to be printed

SG1ERR or TQG1ERR

This is a logical function.

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

SG2ERR or TQG2ERR

This is a logical function.

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

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

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

SG3ERR or TQG3ERR

This is a logical function.

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

RESERR or TQRSERR

Fortran	**RESERR or TQRSERR**
C-interface	**tq_reserr();**
Full name:	Reset Error Code and Message.
Purpose:	This subroutine resets the error code. A subsequent call to the SG* functions gives no error.
Comments:	This should be used when the error has been cleared so that execution can continue. Unless the error code is cleared by this subroutine the SG* functions continue to report the same error.
Arguments	None

TQSP3F

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

Compiler Settings

"Programming Languages" on page 9

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

Compiling FORTRAN Code

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

Windows: Visual Studio 2010, Intel FORTRAN Composer 16

64-bit configuration

Compiler flags:

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

Example:

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

Linux: GNU compiler version 4.4

64-bit configuration

Compiler flags:

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

Example:

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

Linux: Intel FORTRAN Compiler

64-bit configuration

Compiler flags:

```
-real-size 64
```

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

Example:

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

Compiling C code

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

Windows: Visual Studio 2010

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

64-bit configuration

Compiler flags:

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

Example:

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

Linux: GNU compiler version 4.4

64-bit configuration

Compiler flags:

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

Example:

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