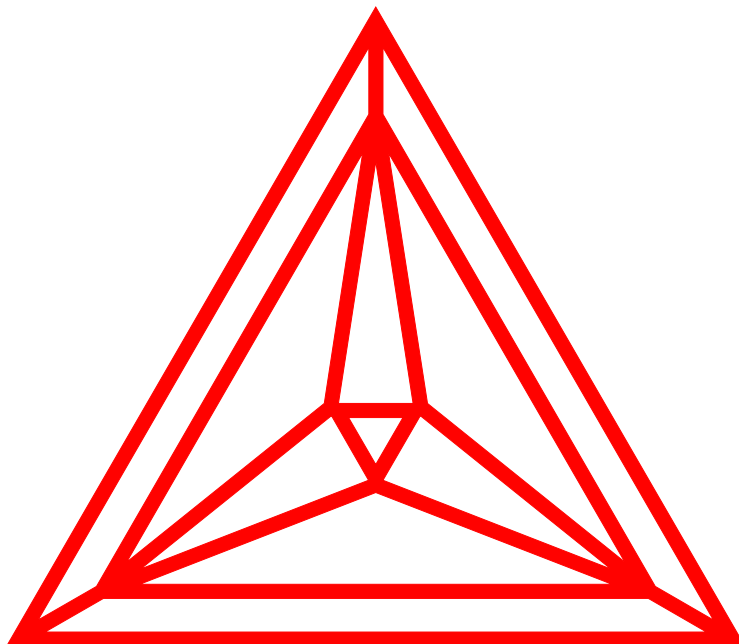


Thermodynamic Calculation Interface

TC-API

(Version 6)

Programmer's Guide



Thermo-Calc Software AB
Norra Stationsgatan 93 5 tr
SE-113 64 Stockholm, Sweden

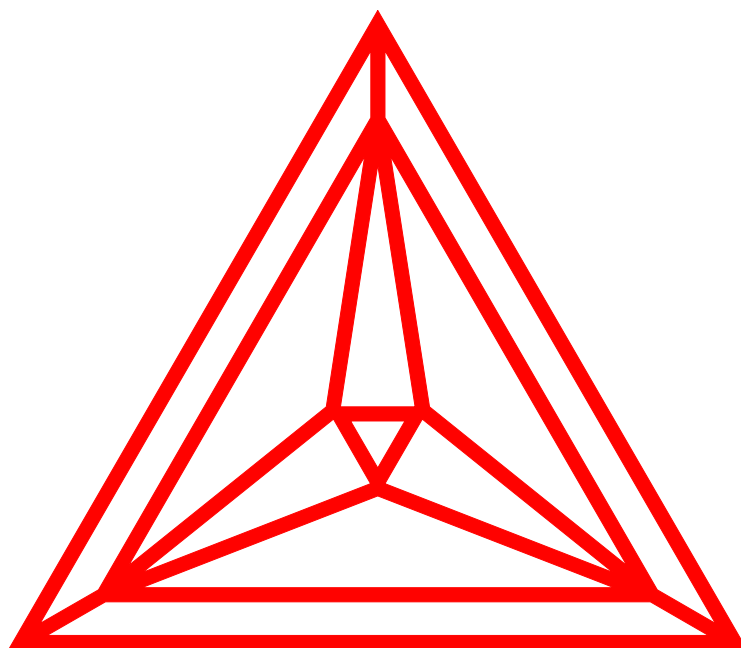
2013

Thermodynamic Calculation Interface

TC-API

(Version 6)

Programmer's Guide



Developer: Thermo-Calc Software AB
Norra Stationsgatan 93 5 tr
SE - 113 64 Stockholm, Sweden

Copyright © 1995-2013 Foundation of Computational Thermodynamics
Stockholm, Sweden

Copyright:

The Thermo-Calc and DICTRA software are the exclusive copyright properties of the STT Foundation (Foundation of Computational Thermodynamics, Stockholm, Sweden). All rights are reserved worldwide.

Thermo-Calc Software AB has the exclusive rights for further developing and marketing all kinds of versions of Thermo-Calc and DICTRA software/database/interface packages, worldwide.

This *TC-API Programmer's Guide*, as well as all other related documentation, is the copyright property of Thermo-Calc Software AB.

It is absolutely forbidden to make any illegal copies of the software, databases, interfaces, and their manuals (User's Guide and Examples Book) and other technical publications (Reference Book and Technical Information). Any unauthorized duplication of such copyrighted products, is a violation of international copyright law. Individuals or organizations (companies, research companies, governmental institutes, and universities) that make or permit to make unauthorized copies may be subject to prosecution.

The utilization of the Thermo-Calc and DICTRA software/database/interface packages and their manuals and other technical information are extensively and permanently governed by the *Thermo-Calc Software AB END USER LICENSE AGREEMENT (EULA)*, which is connected with the software.

Disclaimers:

Thermo-Calc Software AB and the STT Foundation reserve the rights to further developments of the Thermo-Calc and DICTRA software and related software/database/interface products, and to revisions of their manuals and other publications, with no obligation to notify any individual or organization of such developments and revisions. In no event shall Thermo-Calc Software AB and the STT Foundation be liable to any loss of profit or any other commercial damage, including but not limited to special, consequential or other damage.

Please visit the Thermo-Calc Software web site (www.thermocalc.com) for any modification and/or improvement that have been incorporated into the programs, or for any amendment that have made to the contents of the various User's Guides and to the FAQ lists and other technical information publications.

Acknowledgement of Copyright and Trademark Names:

Various third-party software names that are protected by copyright and/or trademarks are mentioned for descriptive purposes, within this User's Guide and other documents of the Thermo-Calc and DICTRA software/database/interface packages. Due acknowledgement is herein made of all such protections.

Contents

1	Thermo-Calc c-api	1
1.1	Installed files	1
1.1.1	TCAPI libraries.	1
1.1.2	Source folder.	1
1.1.3	Project folders for building the example code	1
1.2	Explicit loading or Dynamic linking	1
1.3	About the source code	2
1.3.1	For dynamic linking	2
1.3.2	For explicit loading	2
1.4	When starting developing proprietary projects the following code is needed	2
1.4.1	For dynamic linking	2
1.4.2	For explicit loading	2
2	Data Structure Index	3
2.1	Data Structures	3
3	File Index	5
3.1	File List	5
4	Data Structure Documentation	7
4.1	<code>_tc_function_library</code> Struct Reference	7
4.1.1	Detailed Description	7
5	File Documentation	9
5.1	<code>source/example1.c</code> File Reference	9
5.1.1	Detailed Description	9
5.2	<code>source/example2.c</code> File Reference	9
5.2.1	Detailed Description	9
5.3	<code>source/example3.c</code> File Reference	9
5.3.1	Detailed Description	9
5.4	<code>source/libtc.h</code> File Reference	10
5.4.1	Detailed Description	10
5.4.2	Typedef Documentation	10

5.4.2.1	tc_function_library	10
5.4.3	Function Documentation	10
5.4.3.1	importFunctions	10
5.5	source/tcapi.h File Reference	11
5.5.1	Detailed Description	12
5.5.2	Function Documentation	12
5.5.2.1	tc_append_database	12
5.5.2.2	tc_check_license	12
5.5.2.3	tc_component_status	12
5.5.2.4	tc_compute_equilibrium	12
5.5.2.5	tc_create_new_equilibrium	12
5.5.2.6	tc_database	12
5.5.2.7	tc_define_components	12
5.5.2.8	tc_degrees_of_freedom	12
5.5.2.9	tc_delete_condition	13
5.5.2.10	tc_delete_symbol	13
5.5.2.11	tc_element	13
5.5.2.12	tc_element_reject	13
5.5.2.13	tc_element_select	13
5.5.2.14	tc_enter_ges5_parameter	13
5.5.2.15	tc_enter_symbol	13
5.5.2.16	tc_error	13
5.5.2.17	tc_ges5	13
5.5.2.18	tc_get_data	13
5.5.2.19	tc_get_derivatives	13
5.5.2.20	tc_get_ges5_parameter	14
5.5.2.21	tc_get_value	14
5.5.2.22	tc_init_root	14
5.5.2.23	tc_init_root3	14
5.5.2.24	tc_list_component	14
5.5.2.25	tc_list_conditions	14
5.5.2.26	tc_list_phase	14
5.5.2.27	tc_list_species	14
5.5.2.28	tc_list_symbols	14
5.5.2.29	tc_open_database	14
5.5.2.30	tc_phase	14
5.5.2.31	tc_phase_all_constituents	15
5.5.2.32	tc_phase_constituents	15
5.5.2.33	tc_phase_reject	15
5.5.2.34	tc_phase_select	15

5.5.2.35	<code>tc_phase_status</code>	15
5.5.2.36	<code>tc_phase_structure</code>	15
5.5.2.37	<code>tc_poly3</code>	15
5.5.2.38	<code>tc_read_poly3_file</code>	15
5.5.2.39	<code>tc_reject_constituent</code>	15
5.5.2.40	<code>tc_reset_error</code>	15
5.5.2.41	<code>tc_restore_constituent</code>	16
5.5.2.42	<code>tc_save_poly3_file</code>	16
5.5.2.43	<code>tc_select_equilibrium</code>	16
5.5.2.44	<code>tc_set_component_status</code>	16
5.5.2.45	<code>tc_set_condition</code>	16
5.5.2.46	<code>tc_set_minimization_option</code>	16
5.5.2.47	<code>tc_set_phase_addition</code>	16
5.5.2.48	<code>tc_set_phase_status</code>	16
5.5.2.49	<code>tc_set_start_value</code>	16
5.5.2.50	<code>tc_species_status</code>	16
5.5.2.51	<code>tc_version</code>	16
5.6	source/tcExamples.h File Reference	16
5.6.1	Detailed Description	17
5.6.2	Function Documentation	17
5.6.2.1	example1	17
5.6.2.2	example2	17
5.6.2.3	example3	17
5.7	source/tcMain.c File Reference	17
5.7.1	Detailed Description	17
5.7.2	Function Documentation	17
5.7.2.1	importLibThermoCalc	17
5.7.2.2	loadTCLibraryInCurrentDir	18
5.7.2.3	main	18
5.8	source/tcutils.h File Reference	18
5.8.1	Detailed Description	18
5.8.2	Function Documentation	18
5.8.2.1	getTempEnvironmentPath	18
5.8.2.2	getThermoCalcEnvironmentPath	18

Chapter 1

Thermo-Calc c-api 6

The main part of this manual is a technical description of the TCAPI. To find details on the Thermodynamic applications of each library function see the section on [tcapi.h](#).

1.1 Installed files

In the distribution of Thermo-Calc c-api 6, the following folders and files can be found.

1.1.1 TCAPI libraries.

These could be .lib, .dll, .so -files depending on your installation.

E.g. on a 64 bit Windows system you will find:

tcapi-win-x64-6.dll and tcapi-win-x64-6.lib

1.1.2 Source folder.

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

1.1.3 Project folders for building the example code

Linux:

-> Linux/Linux-Dynamic-Linking

-> Linux/Linux-Explicit-Loading

Windows:

-> Windows-Mingw32-Explicit-Loading

-> Windows-Studio-Project-Dynamic-Linking

-> Windows-Studio-Project-Explicit-Loading

1.2 Explicit loading or Dynamic linking

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

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

1.3 About the source code

1.3.1 For dynamic linking

Simple thermodynamic calculations to demonstrate the use of this api.

[example1.c](#)

[example2.c](#)

[example3.c](#)

Utility functions for finding the correct environment variables

[tcutils.c](#)

[tcutils.h](#)

Declarations and documentation on all TC-API functions

[tcapi.h](#)

Thermo-Calc proprietary declarations and definitions. DO NOT EDIT. [tc_data_defs.h](#)

1.3.2 For explicit loading

Utilities to simplify working with explicitly loaded dll. [libtc.c](#)

[libtc.h](#)

Simple thermodynamic calculations to demonstrate the use of this api.

[tcExamples.c](#)

[tcExamples.h](#)

Main program to exemplify how this api can be used

[tcMain.c](#)

Utility functions for finding the correct environment variables

[tcutils.c](#)

[tcutils.h](#)

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

[tc_data_defs.h](#)

1.4 When starting developing proprietary projects the following code is needed

1.4.1 For dynamic linking

[tcapi.h](#)

[tc_data_defs.h](#)

1.4.2 For explicit loading

[libtc.c](#)

[libtc.h](#)

[tc_data_defs.h](#)

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

Chapter 2

Data Structure Index

2.1 Data Structures

Here are the data structures with brief descriptions:

[_tc_function_library](#) 7

Chapter 3

File Index

3.1 File List

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

source/example1.c	9
source/example2.c	9
source/example3.c	9
source/libtc.h	10
source/tcapi.h	11
source/tcExamples.h	16
source/tcMain.c	17
source/tcutils.h	18

Chapter 4

Data Structure Documentation

4.1 `_tc_function_library` Struct Reference

```
#include <libtc.h>
```

4.1.1 Detailed Description

Data structure that holds the published and supported functions of the Thermo-Calc api.

This datastructure and it's descriptions is mostly of technical interest. For documentation of each function see [tcapi.h](#)

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

- [source/libtc.h](#)

Chapter 5

File Documentation

5.1 source/example1.c File Reference

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

5.1.1 Detailed Description

Basic api example that calculates an equilibrium in the Fe-Cr-C system and retrieves certain quantities.

5.2 source/example2.c File Reference

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

5.2.1 Detailed Description

Basic api example that calculates an equilibrium in the Fe-Cr-C system and retrieves certain quantities.

5.3 source/example3.c File Reference

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

5.3.1 Detailed Description

Shows how to display certain values in arrays.

5.4 source/libtc.h File Reference

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

Data Structures

- struct [_tc_function_library](#)

Typedefs

- typedef struct [_tc_function_library](#) [tc_function_library](#)

Functions

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

5.4.1 Detailed Description

This file contains pointers and Definitions of the published and supported Thermo-Calc functions so that it is easier to use them when explicitly loading the dll. It is all contained in one data structure [_tc_function_library](#)

For documentation of each function see [tcapi.h](#)

5.4.2 Typedef Documentation

5.4.2.1 typedef struct [_tc_function_library](#) [tc_function_library](#)

Data structure that holds the published and supported functions of the Thermo-Calc api.

This datastructure and it's descriptions is mostly of technical interest. For documentation of each function see [tcapi.h](#)

5.4.3 Function Documentation

5.4.3.1 int [importFunctions](#) (TCHANDLE *tcHandle*, [tc_function_library](#) * *tc*, char * *message*)

TCHANDLE tcHandle Handle to shared library

[tc_function_library](#)* tc pointer to data structure that holds functions. Must be allocated by caller.

char* message Error message. Must be allocated by caller.

Call this function to import functions from library.

The shared library must be loaded prior to calling this function.

The [tc_function_library](#)* tc must be allocated by caller.

5.5 source/tcapi.h File Reference

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

Functions

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

- void `tc_save_poly3_file` (TC_STRING filename)
- void `tc_select_equilibrium` (TC_INT equilibrium)
- void `tc_set_component_status` (TC_STRING component_name, TC_STRING status)
- void `tc_set_condition` (TC_STRING condition, TC_FLOAT value)
- void `tc_set_minimization_option` (TC_INT *global_flag, TC_INT *max_gridpoints, TC_INT *frequency, TC_INT *mesh_flag)
- void `tc_set_phase_addition` (TC_STRING phase_name, TC_FLOAT addition)
- void `tc_set_phase_status` (TC_STRING phase_name, TC_STRING status, TC_FLOAT value)
- void `tc_set_start_value` (TC_STRING state_variable, TC_FLOAT starting_value)
- TC_STRING `tc_species_status` (TC_STRING species_name)
- void `tc_version` (TC_STRING str, TC_INT str_len)

5.5.1 Detailed Description

Complete description of the available functions in the TCAPI.

5.5.2 Function Documentation

5.5.2.1 void `tc_append_database` (TC_STRING *name*)

Appends the named database

5.5.2.2 TC_BOOL `tc_check_license` (TC_STRING *name*, TC_STRING *message*, TC_STRING_LENGTH *strlen_message*)

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

5.5.2.3 TC_STRING `tc_component_status` (TC_STRING *component_name*)

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

5.5.2.4 void `tc_compute_equilibrium` ()

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

5.5.2.5 void `tc_create_new_equilibrium` (TC_INT *equilibrium*)

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

5.5.2.6 TC_INT `tc_database` (TC_STRING *datan*, TC_INT *linelen*)

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

5.5.2.7 void `tc_define_components` (TC_STRING *component_name*, TC_INT *strlen*, TC_INT *number_of_components*)

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

5.5.2.8 TC_INT `tc_degrees_of_freedom` ()

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

5.5.2.9 void `tc_delete_condition` (TC_STRING *condition*)

Deletes the condition for the expression in "condition".

5.5.2.10 void `tc_delete_symbol` (TC_STRING *symbol*)

Deletes a symbol in the system.

5.5.2.11 TC_INT `tc_element` (TC_STRING *elements*, TC_INT *linelen*)

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

5.5.2.12 void `tc_element_reject` (TC_STRING *element_name*)

Rejects "element name" in the currently selected database.

5.5.2.13 void `tc_element_select` (TC_STRING *element_name*)

Selects "element name" in the currently selected database.

5.5.2.14 void `tc_enter_ges5_parameter` (TC_STRING *parameter*, TC_STRING *expression*)

Enters a parameter expression

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

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

5.5.2.16 TC_BOOL `tc_error` (TC_INT * *error_number*, TC_STRING *message*, TC_INT *strlen*)

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

5.5.2.17 void `tc_ges5` (TC_STRING *command*)

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

5.5.2.18 void `tc_get_data` ()

Executes the database command "GET_DATA"

5.5.2.19 void `tc_get_derivatives` (TC_STRING *phase_name*, TC_FLOAT * *arr1*, TC_FLOAT * *arr2*)

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

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

Retrieves the expression of a parameter name

5.5.2.21 TC_FLOAT tc_get_value (TC_STRING *symbol*)

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

5.5.2.22 TC_INT tc_init_root ()

Initializes the Thermo-Calc system, must be called prior to anything else.

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

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

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

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

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

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

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

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

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

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

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

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

5.5.2.29 void tc_open_database (TC_STRING *name*)

Opens the named database "*name_of_database*"

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

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

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

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

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

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

5.5.2.33 `void tc_phase_reject (TC_STRING phase_name)`

Rejects the phase in "phase_name"

5.5.2.34 `void tc_phase_select (TC_STRING phase_name)`

Selects the phase in "phase_name"

5.5.2.35 `TC_STRING tc_phase_status (TC_STRING phase_name)`

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

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

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

5.5.2.37 `void tc_poly3 (TC_STRING command)`

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

5.5.2.38 `void tc_read_poly3_file (TC_STRING filename)`

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

5.5.2.39 `void tc_reject_constituent (TC_STRING phase_name, TC_INT sublattice, TC_STRING constituent)`

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

5.5.2.40 `void tc_reset_error ()`

Resets the error if an error has been set

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

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

5.5.2.42 void tc_save_poly3_file (TC_STRING *filename*)

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

5.5.2.43 void tc_select_equilibrium (TC_INT *equilibrium*)

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

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

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

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

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

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

Sets parameters for global minimization

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

Sets the addition "addition" to the Gibbs

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

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

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

Sets a starting value for the "state variable" to "start value".

5.5.2.50 TC_STRING tc_species_status (TC_STRING *species_name*)

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

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

Returns the version of Thermo-Calc in "version_name"

5.6 source/tcExamples.h File Reference

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

Functions

- void [example1](#) (tc_function_library *tc)
- void [example2](#) (tc_function_library *tc)
- void [example3](#) (tc_function_library *tc)

5.6.1 Detailed Description

Runs the equivalent examples of [example1.c](#), [example2.c](#), [example3.c](#)

5.6.2 Function Documentation

5.6.2.1 void example1 (tc_function_library * tc)

Basic api example that calculates an equilibrium in the Fe-Cr-C system and retrieves certain quantities.

5.6.2.2 void example2 (tc_function_library * tc)

Basic api example that calculates an equilibrium in the Fe-Cr-C system and retrieves certain quantities.

5.6.2.3 void example3 (tc_function_library * tc)

Shows how to display certain values in arrays.

5.7 source/tcMain.c File Reference

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

Functions

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

5.7.1 Detailed Description

This is the main file to run the Thermo-Calc examples. It contains a short example how the dll can be loaded and How to fetch the Thermo-Calc functions.

To get started, build this project and run it. The executable will be placed next to a copy of the tcapi dll/library.

5.7.2 Function Documentation

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

Parameters

<i>tc</i>	pointer to data structure that contains Thermo-Calc functions.
<i>message</i>	error message. Should be allocated by caller

This function does two things.

- 1) Loads the library
- 2) Calls importFunctions that fetches the functions to the data structure.

5.7.2.2 TCHANDLE loadTCLibraryInCurrentDir ()

Loads the library (os specific) and returns a handle to the library.

This function expects that the executable is placed next to the library. However, the library can be placed anywhere as long as the correct path is given.

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

Main function

Start here to run the different examples Each example needs the path to a temporary directory to store log files, and the path to the thermo-calc installation where it looks for the databases.

This can be changed if for example another set of databases should be used.

5.8 source/tcutils.h File Reference

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

Functions

- void [getThermoCalcEnvironmentPath](#) (char *pathBuffer)
- void [getTempEnvironmentPath](#) (char *pathBuffer)

5.8.1 Detailed Description

Simple utility functions to work with environment variables.

5.8.2 Function Documentation

5.8.2.1 void getTempEnvironmentPath (char * pathBuffer)

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

5.8.2.2 void getThermoCalcEnvironmentPath (char * pathBuffer)

Get path from the values of Thermo-Calc environment variables TC3_HOME (Thermo-Calc 3.0) TCPATH Older versions and fallback

Index

`_tc_function_library`, 7

example1
 `tcExamples.h`, 17

example2
 `tcExamples.h`, 17

example3
 `tcExamples.h`, 17

`getTempEnvironmentPath`
 `tcutils.h`, 18

`getThermoCalcEnvironmentPath`
 `tcutils.h`, 18

`importFunctions`
 `libtc.h`, 10

`importLibThermoCalc`
 `tcMain.c`, 17

`libtc.h`
 `importFunctions`, 10
 `_tc_function_library`, 10

`loadTCLibraryInCurrentDir`
 `tcMain.c`, 18

main
 `tcMain.c`, 18

`source/example1.c`, 9

`source/example2.c`, 9

`source/example3.c`, 9

`source/libtc.h`, 10

`source/tcExamples.h`, 16

`source/tcMain.c`, 17

`source/tcapi.h`, 11

`source/tcutils.h`, 18

`tc_append_database`
 `tcapi.h`, 12

`tc_check_license`
 `tcapi.h`, 12

`tc_component_status`
 `tcapi.h`, 12

`tc_compute_equilibrium`
 `tcapi.h`, 12

`tc_create_new_equilibrium`
 `tcapi.h`, 12

`tc_database`
 `tcapi.h`, 12

`tc_define_components`
 `tcapi.h`, 12

`tc_degrees_of_freedom`
 `tcapi.h`, 12

`tc_delete_condition`
 `tcapi.h`, 12

`tc_delete_symbol`
 `tcapi.h`, 13

`tc_element`
 `tcapi.h`, 13

`tc_element_reject`
 `tcapi.h`, 13

`tc_element_select`
 `tcapi.h`, 13

`tc_enter_ges5_parameter`
 `tcapi.h`, 13

`tc_enter_symbol`
 `tcapi.h`, 13

`tc_error`
 `tcapi.h`, 13

`tc_function_library`
 `libtc.h`, 10

`tc_ges5`
 `tcapi.h`, 13

`tc_get_data`
 `tcapi.h`, 13

`tc_get_derivatives`
 `tcapi.h`, 13

`tc_get_ges5_parameter`
 `tcapi.h`, 13

`tc_get_value`
 `tcapi.h`, 14

`tc_init_root`
 `tcapi.h`, 14

`tc_init_root3`
 `tcapi.h`, 14

`tc_list_component`
 `tcapi.h`, 14

`tc_list_conditions`
 `tcapi.h`, 14

`tc_list_phase`
 `tcapi.h`, 14

`tc_list_species`
 `tcapi.h`, 14

`tc_list_symbols`
 `tcapi.h`, 14

`tc_open_database`
 `tcapi.h`, 14

`tc_phase`
 `tcapi.h`, 14

`tc_phase_all_constituents`

- tcapi.h, 14
- tc_phase_constituents
 - tcapi.h, 15
- tc_phase_reject
 - tcapi.h, 15
- tc_phase_select
 - tcapi.h, 15
- tc_phase_status
 - tcapi.h, 15
- tc_phase_structure
 - tcapi.h, 15
- tc_poly3
 - tcapi.h, 15
- tc_read_poly3_file
 - tcapi.h, 15
- tc_reject_constituent
 - tcapi.h, 15
- tc_reset_error
 - tcapi.h, 15
- tc_restore_constituent
 - tcapi.h, 15
- tc_save_poly3_file
 - tcapi.h, 16
- tc_select_equilibrium
 - tcapi.h, 16
- tc_set_component_status
 - tcapi.h, 16
- tc_set_condition
 - tcapi.h, 16
- tc_set_minimization_option
 - tcapi.h, 16
- tc_set_phase_addition
 - tcapi.h, 16
- tc_set_phase_status
 - tcapi.h, 16
- tc_set_start_value
 - tcapi.h, 16
- tc_species_status
 - tcapi.h, 16
- tc_version
 - tcapi.h, 16
- tcExamples.h
 - example1, 17
 - example2, 17
 - example3, 17
- tcMain.c
 - importLibThermoCalc, 17
 - loadTCLibraryInCurrentDir, 18
 - main, 18
- tcapi.h
 - tc_append_database, 12
 - tc_check_license, 12
 - tc_component_status, 12
 - tc_compute_equilibrium, 12
 - tc_create_new_equilibrium, 12
 - tc_database, 12
 - tc_define_components, 12
 - tc_degrees_of_freedom, 12
 - tc_delete_condition, 12
 - tc_delete_symbol, 13
 - tc_element, 13
 - tc_element_reject, 13
 - tc_element_select, 13
 - tc_enter_ges5_parameter, 13
 - tc_enter_symbol, 13
 - tc_error, 13
 - tc_ges5, 13
 - tc_get_data, 13
 - tc_get_derivatives, 13
 - tc_get_ges5_parameter, 13
 - tc_get_value, 14
 - tc_init_root, 14
 - tc_init_root3, 14
 - tc_list_component, 14
 - tc_list_conditions, 14
 - tc_list_phase, 14
 - tc_list_species, 14
 - tc_list_symbols, 14
 - tc_open_database, 14
 - tc_phase, 14
 - tc_phase_all_constituents, 14
 - tc_phase_constituents, 15
 - tc_phase_reject, 15
 - tc_phase_select, 15
 - tc_phase_status, 15
 - tc_phase_structure, 15
 - tc_poly3, 15
 - tc_read_poly3_file, 15
 - tc_reject_constituent, 15
 - tc_reset_error, 15
 - tc_restore_constituent, 15
 - tc_save_poly3_file, 16
 - tc_select_equilibrium, 16
 - tc_set_component_status, 16
 - tc_set_condition, 16
 - tc_set_minimization_option, 16
 - tc_set_phase_addition, 16
 - tc_set_phase_status, 16
 - tc_set_start_value, 16
 - tc_species_status, 16
 - tc_version, 16
- tcutils.h
 - getTempEnvironmentPath, 18
 - getThermoCalcEnvironmentPath, 18