

hitran2refr

Richard J. Mathar, J. Opt. A: Pure Appl. Opt. 9 (2007) 470

Generated by Doxygen 1.8.2

Mon Oct 15 2012 11:46:06

Contents

1	Main Page	1
2	Module Index	1
2.1	Modules	1
3	Data Structure Index	1
3.1	Data Structures	1
4	File Index	2
4.1	File List	2
5	Module Documentation	2
5.1	refractive index calculation	2
5.1.1	Compilation	3
5.1.2	Databases	3
6	Data Structure Documentation	3
6.1	Air Class Reference	4
6.1.1	Constructor & Destructor Documentation	5
6.1.2	Member Function Documentation	6
6.1.3	Field Documentation	7
6.2	Air::compon Struct Reference	9
6.2.1	Field Documentation	10
6.3	molecule Class Reference	11
6.3.1	Constructor & Destructor Documentation	12
6.3.2	Member Function Documentation	12
6.3.3	Field Documentation	12
6.4	WagnerPruss::tab62aS Struct Reference	13
6.4.1	Field Documentation	13
6.5	WagnerPruss::tab62bS Struct Reference	14
6.5.1	Field Documentation	14
6.6	WagnerPruss::tab62cS Struct Reference	15
6.6.1	Field Documentation	15
6.7	transition Class Reference	16
6.7.1	Detailed Description	17
6.7.2	Constructor & Destructor Documentation	17
6.7.3	Member Function Documentation	18
6.7.4	Field Documentation	18
6.8	virial Class Reference	19
6.8.1	Constructor & Destructor Documentation	20

6.8.2	Member Function Documentation	20
6.8.3	Field Documentation	22
6.9	WagnerPruss Class Reference	22
6.9.1	Detailed Description	23
6.9.2	Constructor & Destructor Documentation	23
6.9.3	Member Function Documentation	23
6.9.4	Field Documentation	25
6.10	water Class Reference	26
6.10.1	Constructor & Destructor Documentation	27
6.10.2	Member Function Documentation	27
6.10.3	Field Documentation	28
7	File Documentation	28
7.1	Air.cxx File Reference	28
7.2	Air.h File Reference	29
7.2.1	Macro Definition Documentation	30
7.3	hitran2refr.cxx File Reference	30
7.3.1	Macro Definition Documentation	30
7.3.2	Function Documentation	31
7.3.3	Usage	31
7.4	hitran2refr.tcl File Reference	32
7.4.1	Function Documentation	32
7.5	molecule.cxx File Reference	32
7.5.1	Macro Definition Documentation	33
7.6	molecule.h File Reference	34
7.7	prconst.h File Reference	35
7.7.1	Macro Definition Documentation	35
7.8	tips_2003.cxx File Reference	36
7.8.1	Macro Definition Documentation	37
7.8.2	Function Documentation	37
7.9	tips_2003.h File Reference	47
7.9.1	Function Documentation	48
7.10	transition.cxx File Reference	58
7.10.1	Macro Definition Documentation	59
7.10.2	Function Documentation	59
7.11	transition.h File Reference	60
7.11.1	Macro Definition Documentation	61
7.11.2	Enumeration Type Documentation	61
7.12	units.h File Reference	62
7.12.1	Macro Definition Documentation	63

7.13 virial.cxx File Reference	64
7.13.1 Macro Definition Documentation	64
7.14 virial.h File Reference	65
7.14.1 Enumeration Type Documentation	65
7.15 WagnerPruss.cxx File Reference	66
7.16 WagnerPruss.h File Reference	66
7.17 water.cxx File Reference	67
7.17.1 Macro Definition Documentation	67
7.18 water.h File Reference	68
7.18.1 Enumeration Type Documentation	68

Index	68
--------------	-----------

1 Main Page

2 Module Index

2.1 Modules

Here is a list of all modules:

refractive index calculation	2
-------------------------------------	----------

3 Data Structure Index

3.1 Data Structures

Here are the data structures with brief descriptions:

Air	4
Air::compon	9
molecule	11
WagnerPruss::tab62aS	13
WagnerPruss::tab62bS	14
WagnerPruss::tab62cS	15
transition	16
virial	19
WagnerPruss	22
water	26

4 File Index

4.1 File List

Here is a list of all files with brief descriptions:

Air.cxx	28
Air.h	29
hitran2refr.cxx	30
hitran2refr.tcl	32
molecule.cxx	32
molecule.h	34
prconst.h	35
tips_2003.cxx	36
tips_2003.h	47
transition.cxx	58
transition.h	60
units.h	62
virial.cxx	64
virial.h	65
WagnerPruss.cxx	66
WagnerPruss.h	66
water.cxx	67
water.h	68

5 Module Documentation

5.1 refractive index calculation

Since

2001-07-18
2009-01-10 option -D to adapt to database file relocations
2010-01-20 added tcl GUI

Author

Richard J. Mathar

This small collection of C++ classes calculates refractive indices of air in the VIS to IR wavelength ranges, as published in [? ? ?].

5.1.1 Compilation

Compilation in the style of

```
make
```

is supported with a corresponding makefile. The default assumes the availability of the `GSL` libraries in the search paths for headers and libraries; the corresponding compiler options in `Makefile` may need adjustment. It is possible to compile the sources without availability of these two libraries, but I do not have the time to adjust the code or configuration to such needs.

5.1.2 Databases

The databases with line strengths come in various forms, not included in `hitran2refr.tgz`, [HITRAN 2004](#), [JPL](#) and [the CFA](#). The location of these files is specified in `hitran2refr.cxx` in the CPP definitions `HITRAN2000-DIR`, `SPECJPLDIR`, `CFAWWWDIR` and `JCP111DIR`. The directory with these directories is

```
..
```

by default and can be specified with the

```
-D
```

option of the main executable.

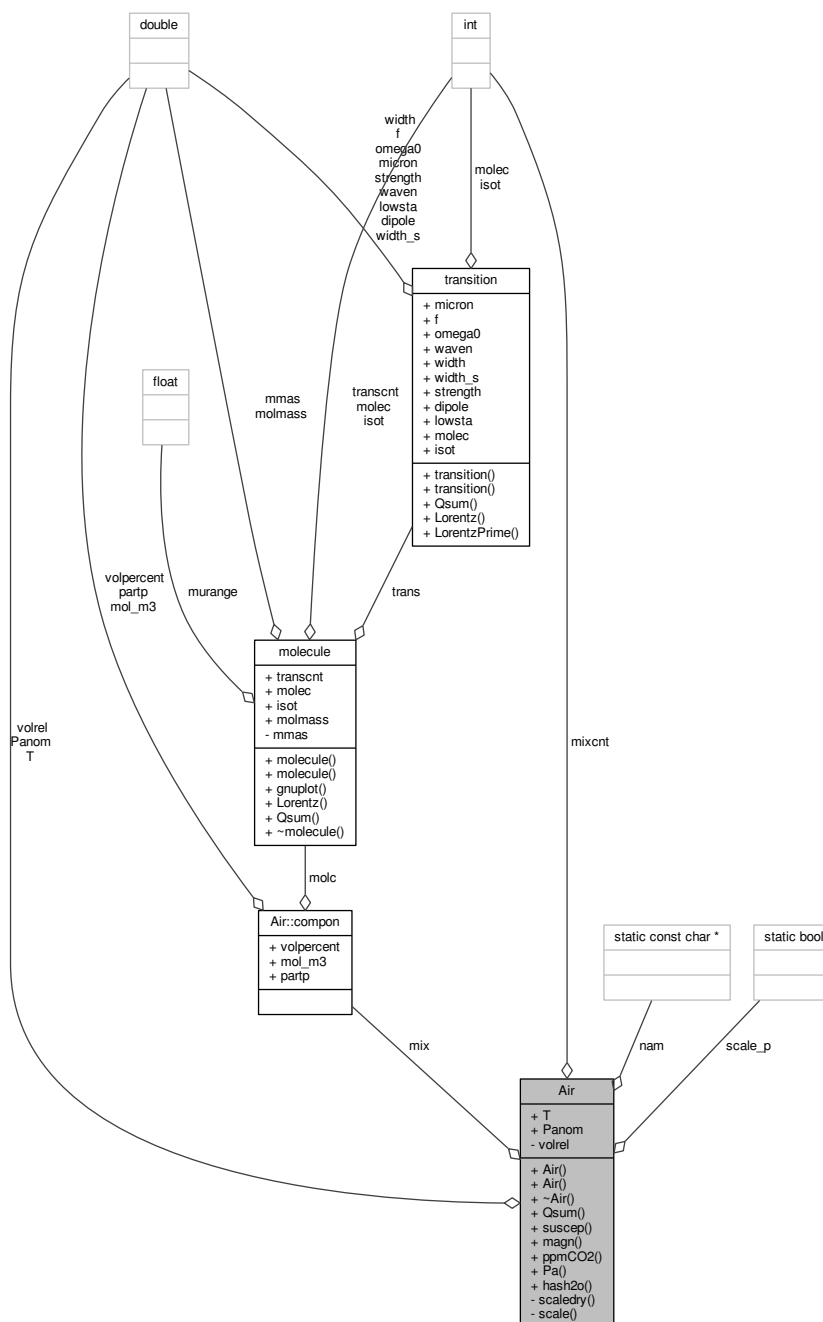
Implementation and documentation come without any warranty and are subject to quick and major changes at any point in the future.

[Richard J. Mathar home page www.strw.leidenuniv.nl/~mathar](http://www.strw.leidenuniv.nl/~mathar).

6 Data Structure Documentation

6.1 Air Class Reference

Collaboration diagram for Air:



Data Structures

- struct [compon](#)

Public Member Functions

- [Air](#) (const double temp, const double press)

- [Air](#) (const double temp, const double press, const double relhum, const double co2, const double o2, const double h2o, const double o3, const bool useV, const string dir, int fcount, char *argv[])
- [~Air](#) ()
- void [Qsum](#) ()
- complex< double > [suscep](#) (const double lambda)
- double [magn](#) (const double useV)
- double [ppmCO2](#) () const
- double [Pa](#) (double &mols, const bool excludh2o) const
- int [hash2o](#) (const int molecNo) const

Data Fields

- double [T](#)
- double [Panom](#)
- struct [compon](#) * [mix](#)
- int [mixcnt](#)

Static Public Attributes

- static bool [scale_p](#)

Private Member Functions

- void [scaledry](#) (const bool useV)
- void [scale](#) (const bool useV, const set< int > excl)

Static Private Attributes

- static const double [volrel](#) []
- static const char * [nam](#) []

6.1.1 Constructor & Destructor Documentation

6.1.1.1 [Air::Air](#) (const double *temp*, const double *press*)

Default Ctor.

Parameters

in	<i>temp</i>	
in	<i>press</i>	

6.1.1.2 [Air::Air](#) (const double *temp*, const double *press*, const double *relhum*, const double *co2*, const double *o2*, const double *h2o*, const double *o3*, const bool *useV*, const string *dir*, int *fcount*, char * *argv*[])

Constructor with files defining constituents.

Parameters

in	<i>temp</i>	
in	<i>press</i>	
in	<i>relhum</i>	
in	<i>co2</i>	
in	<i>h2o</i>	
in	<i>useV</i>	

in	<i>dir</i>	
in	<i>fcoun</i>	
in		

6.1.1.3 Air::~Air ()

6.1.2 Member Function Documentation

6.1.2.1 int Air::hash2o (const int *molecNo*) const

Check if water is in the molecular list.

Parameters

in	<i>molecNo</i>	the HITRAN molecular number to search for. Chose 1 for water, for example.
----	----------------	----------------------------------------------------------------------------

Returns

the index (position) of water in the line list. -1 if absent.

6.1.2.2 double Air::magn (const double *useV*)

Compute the magnetic susceptibility.

Parameters

in	<i>useV</i>	if true, use virials in the equation of state
----	-------------	-----------------------------------------------

See Also

[?]

6.1.2.3 double Air::Pa (double & *mols*, const bool *excludh2o*) const

Compute the total (external) pressure.

Parameters

in	<i>excludh2o</i>	if true, any contribution of H2O is ignored (skipped, implicitly zero)
----	------------------	------------------------------------------------------------------------

Returns

pressure [Pa]

6.1.2.4 double Air::ppmCO2 () const

Return the volume fraction (roughly 370) of carbon dioxide if it was in the list of molecules included. Otherwise return zero.

Returns

CO2 volume fraction in ppmv

6.1.2.5 void Air::Qsum ()

6.1.2.6 void Air::scale (const bool *useV*, const set< int > *excl*) [private]

Parameters

in	<i>useV</i>	if true, use virials in the equation of state, else ideal gas law
in	<i>excl</i>	list of HITRAN molecular numbers excluded from re-scaling

Since

2008-03-19

6.1.2.7 void Air::scaledry (const bool *useV*) [private]

Parameters

in	<i>useV</i>	if true, use virials in the equation of state, else ideal gas law
----	-------------	-------------------------------------------------------------------

6.1.2.8 complex< double > Air::suscep (const double *lambda*)

6.1.3 Field Documentation

6.1.3.1 struct compon* Air::mix

The array of the molecular constituents

6.1.3.2 int Air::mixcnt

Number (count) of constituents in the `mix` array

6.1.3.3 const char * Air::nam [static],[private]

Initial value:

```
= {
    "H2O", "CO2", "O3", "N2O", "CO", "CH4", "O2", "NO", "SO2", "NO2",
    "NH3", "HNO3", "OH", "HF", "HCl", "HBr", "HI", "ClO", "OCS", "H2CO",
    "HOC1", "N2", "HCN", "CH3C1", "H2O2", "C2H2", "C2H6", "PH3", "COF2", "SF6",
    "H2S", "HCOOH", "HO2", "O", "ClONO2", "NO+", "HOBr", "C2H4", "CH3OH", "",
    "", "", "", "", "", "", "", "", "", "", "", "", "", "", "",
    "", "", "", "", "", "", "", "", "", "", "", "", "", "", "",
    "", "", "", "", "", "", "", "", "", "", "", "", "", "", "",
    "", "", "", "", "", "", "", "", "", "", "", "", "", "", "",
    "", "", "", "", "", "", "", "", "", "", "", "", "", "", "",
    "", "", "", "", "", "", "", "", "", "", "", "", "", "", "",
    "Kr", "Ar", "H2", "He"}

```

Standard molecule denominations sorted in HITRAN order

6.1.3.4 double Air::Panom

pressure [Pa]. Total nominal pressure

6.1.3.5 bool Air::scale_p [static]

flag: if true, adjust the densities of the molecules in the dry portion to get the full `Air::Panom` if all partial pressures are added

6.1.3.6 double Air::T

temperature [K]

6.1.3.7 const double Air::volrel [static],[private]

Initial value:

= {

```

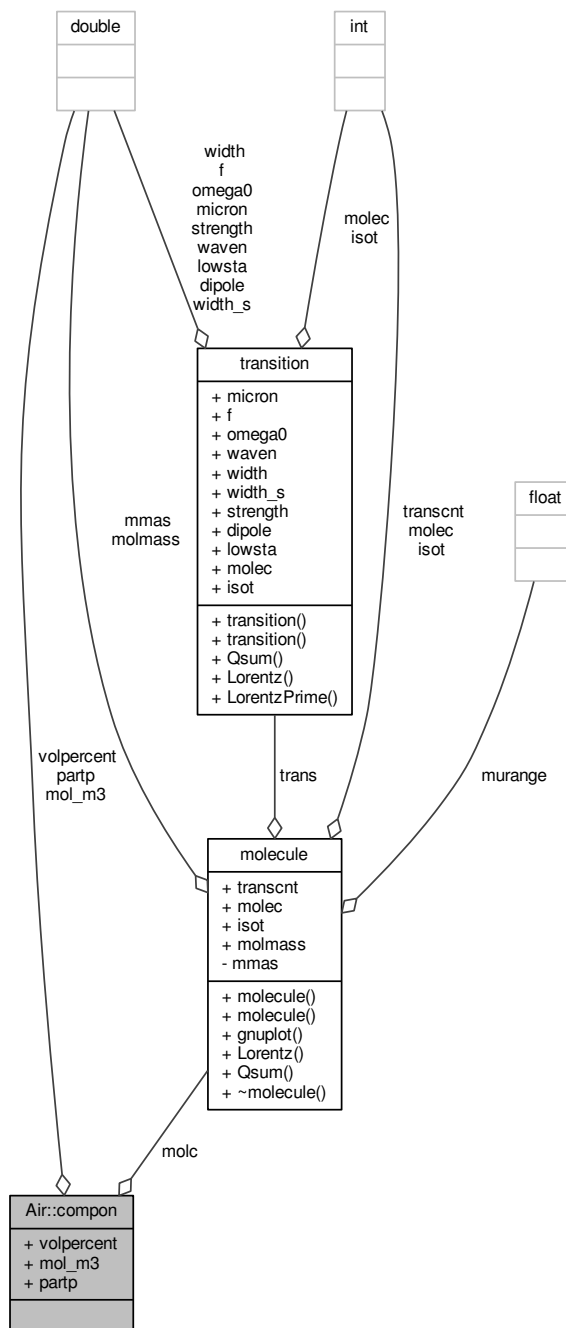
0.,3.7e-4,1.e-7,3.0e-7,1.0e-7,1.6e-6,0.20983,5.e-10,5.e-10,1.e-9,
4.e-7, 0., 3.9e-13, 0.,0.,0.,0.,0.,0.,0.,0.,
0.,0.781, 0.,0.,0.,0.,0.,0.,0.,0.,
0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,
0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,
0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,
0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,
0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,
0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,
0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,
0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,
0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,
1.e-6,9.34e-3,5.e-7,5.2e-6
}

```

Mixing ratios of the standard air, all between 0 and 1, ordered along the HITRAN molecule numbers but shifted by one index to start with the C/C++ 0-based notation: index 0 = H2O, 1=CO2, 2=O3, 3=N2O, 4=CO, 5=CH4, 6=O2, 7=NO, 8=SO2, 9=NO2, 10=NH3, 11=HNO3 (already in NO2), 12=OH, 21=N2, ATOMS_OFFSET-1=Ne, Kr, Ar, H2, He

6.2 Air::compon Struct Reference

Collaboration diagram for Air::compon:



Data Fields

- `molecule` * `molc`
- `double` `volpercent`
- `double` `mol_m3`
- `double` `partp`

6.2.1 Field Documentation

6.2.1.1 double Air::compon::mol_m3

count of molecules per cubic meter, [1/m³]

6.2.1.2 molecule* Air::compon::molc

The molecule species itself with its line list

6.2.1.3 double Air::compon::partp

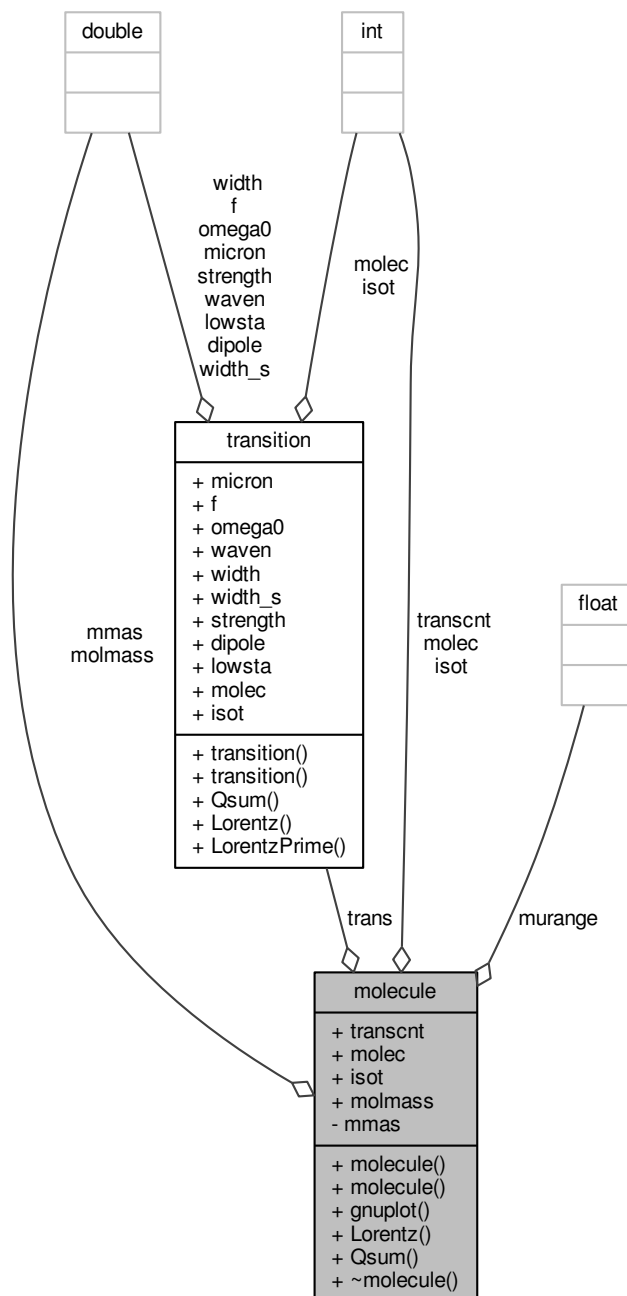
partial pressure [mm Hg]

6.2.1.4 double Air::compon::volpercent

The mixing ratio. This is the roughly percentage of the volume divided through 100, but this latter interpretation is only correct if all components are ideal gases.

6.3 molecule Class Reference

Collaboration diagram for molecule:



Public Member Functions

- [molecule](#) (int molno=0)
- [molecule](#) (const char *fname, const string dir, int molno=0, int iso=1)
- void [gnuplot](#) (const double lowlim, const double hilim, ostream &out=cout)
- complex< double > [Lorentz](#) (const double lambda) const

- void `Qsum` (double *T*)
- `~molecule` ()

Data Fields

- `transition` * `trans`
- int `transcnt`
- int `molec`
- int `isot`
- double `molmass`
- float `murange` [2]

Static Private Attributes

- static const double `mmas` []

6.3.1 Constructor & Destructor Documentation

6.3.1.1 `molecule::molecule (int molno = 0)`

6.3.1.2 `molecule::molecule (const char * fname, const string dir, int molno = 0, int iso = 1)`

6.3.1.3 `molecule::~molecule ()`

6.3.2 Member Function Documentation

6.3.2.1 `void molecule::gnuplot (const double lowlim, const double hilim, ostream & out = cout)`

6.3.2.2 `complex< double > molecule::Lorentz (const double lambda) const`

6.3.2.3 `void molecule::Qsum (double T)`

6.3.3 Field Documentation

6.3.3.1 `int molecule::isot`

6.3.3.2 `const double molecule::mmas` [static],[private]

Initial value:

```
= {
 18.010565, 43.989830, 47.984745, 44.001062, 27.994915, 16.031300, 31.989830
, 29.997989, 63.961901, 45.992904,
17.026549, 62.995644, 17.002740, 20.006229, 35.976678, 79.926160, 127.91229
7, 50.963768, 59.966986, 30.010565,
51.971593, 28.006147, 27.010899, 49.992328, 0.,0.,0.,0.,0.,0.,
0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,
0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,
0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,
0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,
0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,
0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,
0.,0.,0.,0.,0.,0.,0.,0.,0.,0.,20.183 ,
83.80 , 39.948 , 2.01565 , 4.006
}
```

6.3.3.3 `int molecule::molec`

6.3.3.4 `double molecule::molmass`

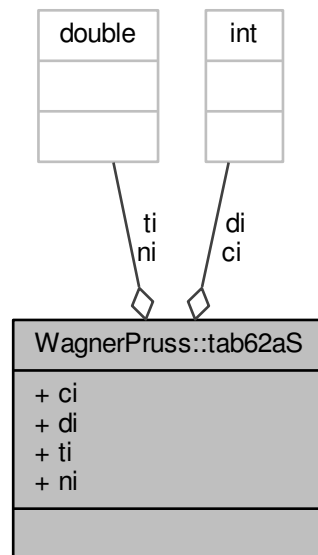
6.3.3.5 `float molecule::murange[2]`

6.3.3.6 transition* molecule::trans

6.3.3.7 int molecule::transcnt

6.4 WagnerPruss::tab62aS Struct Reference

Collaboration diagram for WagnerPruss::tab62aS:



Data Fields

- int [ci](#)
- int [di](#)
- double [ti](#)
- double [ni](#)

6.4.1 Field Documentation

6.4.1.1 int WagnerPruss::tab62aS::ci

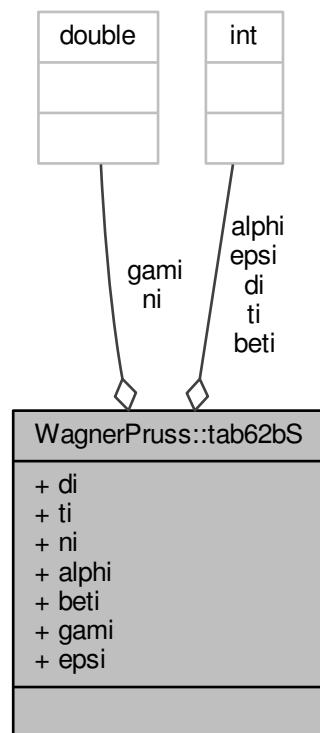
6.4.1.2 int WagnerPruss::tab62aS::di

6.4.1.3 double WagnerPruss::tab62aS::ni

6.4.1.4 double WagnerPruss::tab62aS::ti

6.5 WagnerPruss::tab62bS Struct Reference

Collaboration diagram for WagnerPruss::tab62bS:



Data Fields

- int [di](#)
- int [ti](#)
- double [ni](#)
- int [alphi](#)
- int [beti](#)
- double [gami](#)
- int [epsi](#)

6.5.1 Field Documentation

6.5.1.1 int WagnerPruss::tab62bS::alphi

6.5.1.2 int WagnerPruss::tab62bS::beti

6.5.1.3 int WagnerPruss::tab62bS::di

6.5.1.4 int WagnerPruss::tab62bS::epsi

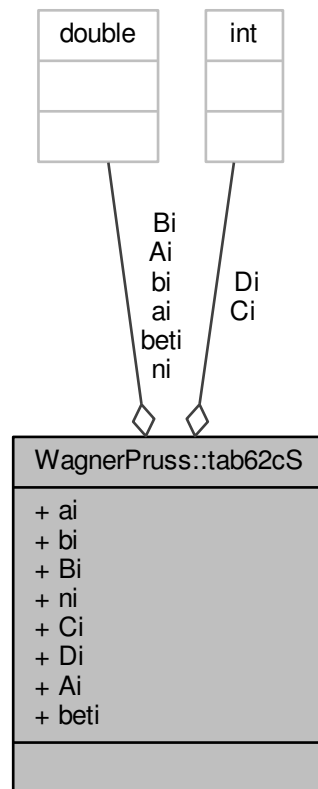
6.5.1.5 double WagnerPruss::tab62bS::gami

6.5.1.6 double WagnerPruss::tab62bS::ni

6.5.1.7 int WagnerPruss::tab62bS::ti

6.6 WagnerPruss::tab62cS Struct Reference

Collaboration diagram for WagnerPruss::tab62cS:



Data Fields

- double `ai`
- double `bi`
- double `Bi`
- double `ni`
- int `Ci`
- int `Di`
- double `Ai`
- double `beti`

6.6.1 Field Documentation

6.6.1.1 double WagnerPruss::tab62cS::ai

6.6.1.2 double WagnerPruss::tab62cS::Ai

6.6.1.3 double WagnerPruss::tab62cS::beti

6.6.1.4 double WagnerPruss::tab62cS::bi

6.6.1.5 double WagnerPruss::tab62cS::Bi

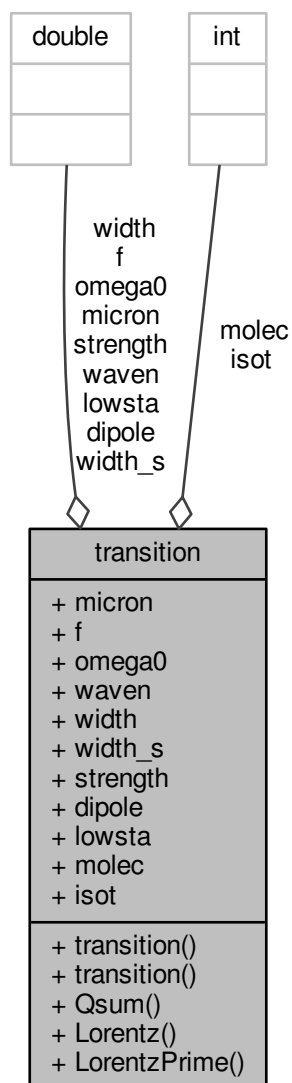
6.6.1.6 int WagnerPruss::tab62cS::Ci

6.6.1.7 int WagnerPruss::tab62cS::Di

6.6.1.8 double WagnerPruss::tab62cS::ni

6.7 transition Class Reference

Collaboration diagram for transition:



Public Member Functions

- [transition](#) (int molno=0)
- [transition](#) (istream &in, [catalog](#) &cat)
- void [Qsum](#) (const double T)
- complex< double > [Lorentz](#) (const double omega) const
- complex< double > [LorentzPrime](#) (const double omega) const

Data Fields

- double [micron](#)
- double [f](#)
- double [omega0](#)
- double [waven](#)
- double [width](#)
- double [width_s](#)
- double [strength](#)
- double [dipole](#)
- double [lowsta](#)
- int [molec](#)
- int [isot](#)

6.7.1 Detailed Description

A line transition with line strength and vacuum wavelength.

Author

Richard J. Mathar

6.7.2 Constructor & Destructor Documentation

6.7.2.1 transition::transition (int *molno* = 0)

default ctor.

Parameters

<code>in</code>	<code>molno</code>	molecular number
-----------------	--------------------	------------------

6.7.2.2 transition::transition (istream & *in*, [catalog](#) & *cat*)

Ctor with a string of a HITRAN line

Parameters

<code>in</code>	<code>in</code>	the ASCII line of one of the databases (catalogs)
<code>in</code>	<code>cat</code>	the type of catalog. That is, the format and units in which the columns of the input are organized.

For HITRAN input lines, recognize the I2, I1, F12.6 etc fortran specs with molecule/isotope, wavenumber, strength, etc.

For JPL input lines, we consider frequencies, intensities etc, and set the unspecified widths to zero.

6.7.3 Member Function Documentation

6.7.3.1 `complex< double > transition::Lorentz (const double omega) const`

Lorentz line profile. Note that this is $f/(\omega_0^2 - \omega^2 - i\gamma\omega)$, where γ is the FWHM, and replaced here by a line form that has the same integral (integrated line strength) but retains the symmetry of turning to the complex conjugated if the ω switches to its negative value.

Parameters

<code>in</code>	<code><i>omega</i></code>
-----------------	---------------------------

Returns

the complex value at the position of `omega` .

6.7.3.2 `complex< double > transition::LorentzPrime (const double omega) const`

First derivative of Lorentz line profile with respect to `omega`.

Parameters

<code>in</code>	<code><i>omega</i></code>
-----------------	---------------------------

Returns

the complex value at the position of `omega` .

6.7.3.3 `void transition::Qsum (const double T)`

If enable by the preprocessor macro, we adapt the line strength to the temperature.

[`?` , (9)], [`?` , (5)].

6.7.4 Field Documentation

6.7.4.1 `double transition::dipole`

squared dipole moment in HITRAN2000 [Debye²]

Since

HITRAN2004 Einstein A coefficient

6.7.4.2 `double transition::f`

unitless oscillator strength

6.7.4.3 `int transition::isot`

HITRAN isotope number

6.7.4.4 `double transition::lowsta`

position/energy of the lower state [1/cm]

6.7.4.5 `double transition::micron`

vacuum wavelength [μ]

6.7.4.6 int transition::molec

HITRAN molecule number. A value of zero is used to indicate non-initialized (quasi invalid) molecular species.

6.7.4.7 double transition::omega0

circular frequency [Hz]

6.7.4.8 double transition::strength

line strength [1/cm / mole/cm²]

6.7.4.9 double transition::waven

wavenumber [1/cm]

6.7.4.10 double transition::width

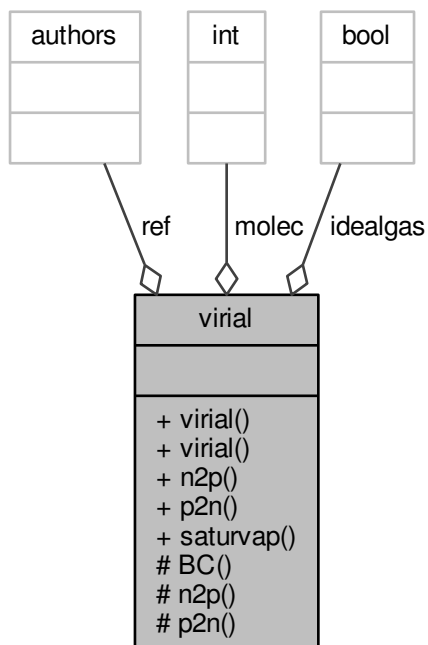
line widths [wavenumber per atm]. Gamma broadened air.

6.7.4.11 double transition::width_s

line widths [wavenumber per atm]. self broadening.

6.8 virial Class Reference

Collaboration diagram for virial:



Public Member Functions

- [virial](#) ()
- [virial](#) (const int species, const [authors](#) src=[None](#))
- double [n2p](#) (double mol_m3, const double T) const
- double [p2n](#) (const double p, const double T) const
- double [saturvap](#) (const double T) const

Protected Member Functions

- void [BC](#) (const double T, double &B, double &C) const
- double [n2p](#) (double mol_m3, const double T, const double viris[2]) const
- double [p2n](#) (const double p, const double T, const double viris[2]) const

Private Attributes

- bool [idealgas](#)
- int [molec](#)
- [authors](#) ref

6.8.1 Constructor & Destructor Documentation

6.8.1.1 [virial::virial](#) ()

default ctor. The default creator implies virials are essentially absent, which just gives the ideal gas behavior.

6.8.1.2 [virial::virial](#) (const int *species*, const [authors](#) *src* = [None](#))

Ctor. This constructor encodes a HITRAN molecule

Parameters

in	<i>species</i>	the HITRAN integer encoding a molecular species
----	----------------	-------------------------------------------------

6.8.2 Member Function Documentation

6.8.2.1 void [virial::BC](#) (const double *T*, double & *B*, double & *C*) const [protected]

Return virial coefficients.

Parameters

in	<i>T</i>	temperature [K]
----	----------	-----------------

Returns

the value of B [m^3/mol] and the value of C [m^6/mol^2].

6.8.2.2 double [virial::n2p](#) (double *mol_m3*, const double *T*) const

Convert number density into partial pressure. That is, convert moles per cubic meter into the partial pressure according to ideal or non-ideal gas equation.

Parameters

in	<i>mol_m3</i>	
in	<i>T</i>	

Returns

partial pressure [Pa]

6.8.2.3 `double virial::n2p (double mol_m3, const double T, const double virls[2]) const` [protected]

Convert number density to partial pressure.

Parameters

in	<i>mol_m3</i>	number density [mol/m ³]
in	<i>T</i>	temperature [K]
in	<i>virls</i>	virial coefficients

Returns

pressure [Pa]

6.8.2.4 `double virial::p2n (const double p, const double T) const`

Convert partial pressure to number density. Inverse functionality of `virial::n2p()`.

Parameters

in	<i>p</i>	partial pressure [Pa]
in	<i>T</i>	temperature [K]

Returns

molecular density [moles/m³]

6.8.2.5 `double virial::p2n (const double p, const double T, const double virls[2]) const` [protected]

Convert partial pressure to number density. Inverse functionality of `virial::n2p()`.

Parameters

in	<i>p</i>	partial pressure [Pa]
in	<i>T</i>	temperature [K]

Returns

density [mol/m³]

6.8.2.6 `double virial::saturvap (const double T) const`

saturation density.

Parameters

in	<i>T</i>	
----	----------	--

Returns

saturation density [kg/m³]

Warning

this is only implemented for the Wagner-Pruss model

6.8.3 Field Documentation

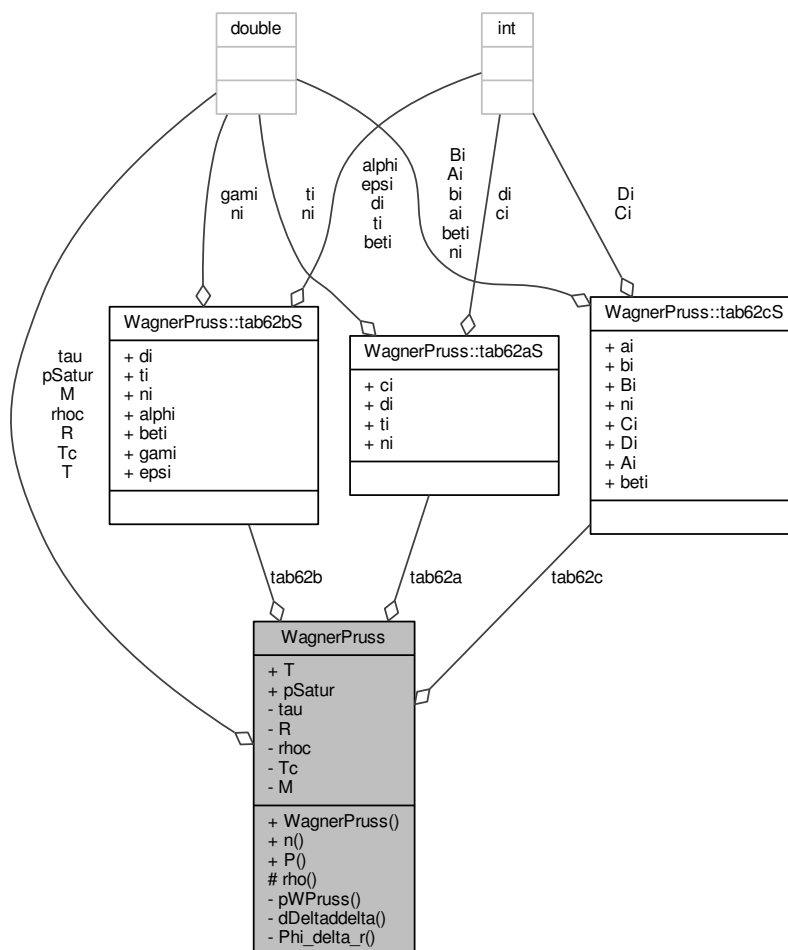
6.8.3.1 `bool virial::idealgas` [`private`]

6.8.3.2 `int virial::molec` [`private`]

6.8.3.3 `authors virial::ref` [`private`]

6.9 WagnerPruss Class Reference

Collaboration diagram for WagnerPruss:



Data Structures

- struct [tab62aS](#)
- struct [tab62bS](#)
- struct [tab62cS](#)

Public Member Functions

- [WagnerPruss](#) (double kelvin)
- double [n](#) (const double humidity) const
- double [P](#) (const double humidity) const

Data Fields

- double [T](#)
- double [pSatur](#)

Protected Member Functions

- double [rho](#) (const double humidity) const

Private Member Functions

- double [pWPruss](#) () const
- double [dDeltadelta](#) (const int i, const double dmin1, const double theta) const
- double [Phi_delta_r](#) (const double delta) const

Private Attributes

- double [tau](#)

Static Private Attributes

- static const [tab62aS](#) [tab62a](#) []
- static const [tab62bS](#) [tab62b](#) []
- static const [tab62cS](#) [tab62c](#) []
- static const double [R](#) =0.46151805
- static const double [rhoc](#) = 322e3
- static const double [Tc](#) =647.096
- static const double [M](#) = 18.015268

6.9.1 Detailed Description

Wagner and Pruss equation of state of water.

[?]

6.9.2 Constructor & Destructor Documentation**6.9.2.1 WagnerPruss::WagnerPruss (double *kelvin*)**

Ctor.

Parameters

<code>in</code>	<code>kelvin</code>	temperature of water (Kelvin)
-----------------	---------------------	-------------------------------

6.9.3 Member Function Documentation

6.9.3.1 `double WagnerPruss::dDeltadelta (const int i, const double dmin1, const double theta) const` [private]

Derivative of a thermodynamic potential.

Returns

$\partial\Delta/\partial\delta$, table 6.5, page 434 [?, p 398]

6.9.3.2 `double WagnerPruss::n (const double humidity) const`

Calculate water density.

Parameters

<code>in</code>	<code><i>humidity</i></code>	relative humidity in percent
-----------------	------------------------------	------------------------------

Returns

water density in moles per cubic meter

6.9.3.3 `double WagnerPruss::P (const double humidity) const`

Calculate partial pressure.

Parameters

<code>in</code>	<code><i>humidity</i></code>	humidity between 0 and 100 (percent)
-----------------	------------------------------	--------------------------------------

Returns

partial pressure [Pa]

6.9.3.4 `double WagnerPruss::Phi_delta_r (const double delta) const` [private]

Derivative of a thermodynamic potential.

Returns

$\partial\Phi'/\partial\delta$, [?, p 398]

6.9.3.5 `double WagnerPruss::pWPruss () const` [private]

Saturation pressure.

Returns

saturation pressure [Pa]

See Also

eq 2.5 [?, p 398]

6.9.3.6 `double WagnerPruss::rho (const double humidity) const` [protected]

Calculate density.

Parameters

in	<i>humidity</i>
----	-----------------

Returns

density in gram per cubic meter

6.9.4 Field Documentation

6.9.4.1 `const double WagnerPruss::M = 18.015268` [static],[private]

6.9.4.2 `double WagnerPruss::pSatur`

saturation pressure (Pa)

6.9.4.3 `const double WagnerPruss::R = 0.46151805` [static],[private]

6.9.4.4 `const double WagnerPruss::rhoc = 322e3` [static],[private]

6.9.4.5 `double WagnerPruss::T`

Temperature (Kelvin)

6.9.4.6 `const WagnerPruss::tab62aS WagnerPruss::tab62a` [static],[private]

6.9.4.7 `const WagnerPruss::tab62bS WagnerPruss::tab62b` [static],[private]

Initial value:

```
= {
  { 3, 0, -0.31306260323435e2, 20, 150, 1.21 ,1 },
  { 3, 1, 0.31546140237781e2, 20, 150, 1.21 ,1 },
  { 3, 4, -0.25213154341695e4, 20, 250, 1.25 ,1 } }
```

6.9.4.8 `const WagnerPruss::tab62cS WagnerPruss::tab62c` [static],[private]

Initial value:

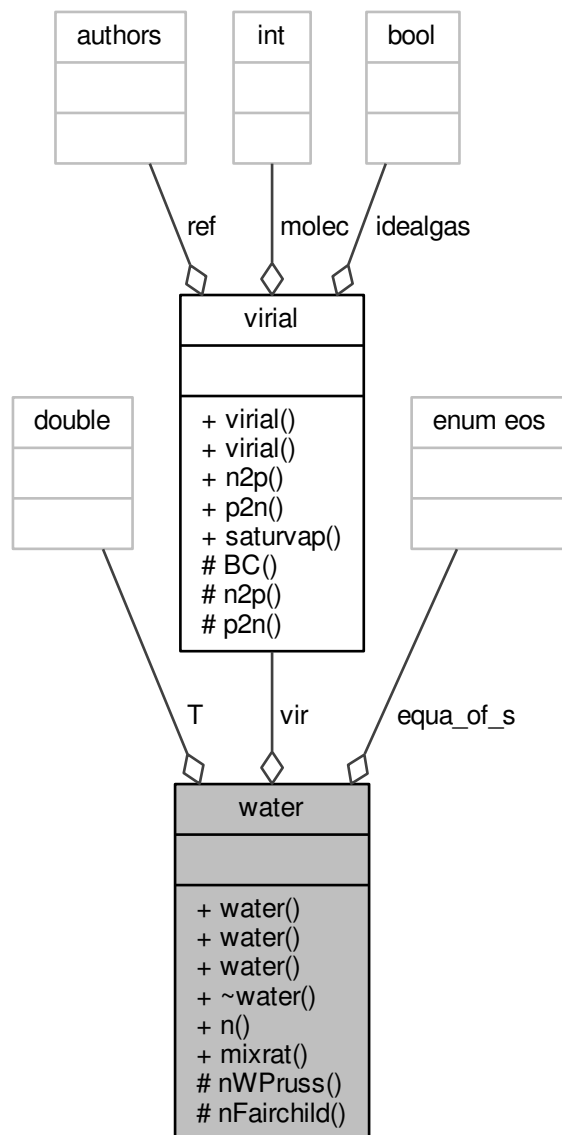
```
= {
  { 3.5, 0.85, 0.2, -0.14874640856724, 28, 700, 0.32 , 0.3 },
  { 3.5, 0.95, 0.2, 0.31806110878444, 32, 800, 0.32 , 0.3 } }
```

6.9.4.9 `double WagnerPruss::tau` [private]

6.9.4.10 `const double WagnerPruss::Tc = 647.096` [static],[private]

6.10 water Class Reference

Collaboration diagram for water:



Public Member Functions

- [water](#) (double kelvin, enum [eos](#) enhanc=[Fairchild](#))
- [water](#) (double kelvin, [virial](#) &v)
- [water](#) (const [water](#) &orig)
- [~water](#) ()
- double [n](#) (const double humidity) const
- double [mixrat](#) (const double humidity, const double pres) const

Protected Member Functions

- double [nWPruss](#) (const double humidity) const
- double [nFairchild](#) (const double humidity) const

Private Attributes

- double [T](#)
- enum [eos](#) [equa_of_s](#)
- [virial](#) * [vir](#)

6.10.1 Constructor & Destructor Documentation

6.10.1.1 `water::water (double kelvin, enum eos enhanc = Fairchild)`

6.10.1.2 `water::water (double kelvin, virial & v)`

Ctor

Parameters

in	<i>kelvin</i>	temperature [K]
in	<i>v</i>	a virial for the deviations from the ideal gas equation

6.10.1.3 `water::water (const water & orig)`

Copy ctor.

6.10.1.4 `water::~~water ()`

Dtor.

6.10.2 Member Function Documentation

6.10.2.1 `double water::mixrat (const double humidity, const double pres) const`

Compute water mixing ratio.

Parameters

in	<i>humidity</i>	relative humidity in percent
in	<i>pres</i>	total pressure [Pa]

Returns

mixing ration between 0 (dry) and 1

6.10.2.2 `double water::n (const double humidity) const`

Compute water density.

Parameters

in	<i>humidity</i>	relative humidity in percent, from 0 to 100
----	-----------------	---------------------------------------------

Returns

moles per cubic meter

6.10.2.3 `double water::nFairchild (const double humidity) const` [protected]

Water densities at the saturation point according to Fairchild

6.10.2.4 `double water::nWPruss (const double humidity) const` [protected]

Wagner-Pruss molecular density.

Parameters

<code>in</code>	<code><i>humidity</i></code>
-----------------	------------------------------

Returns

moles per cubic meter

6.10.3 Field Documentation

6.10.3.1 `enum eos water::equa_of_s` [private]

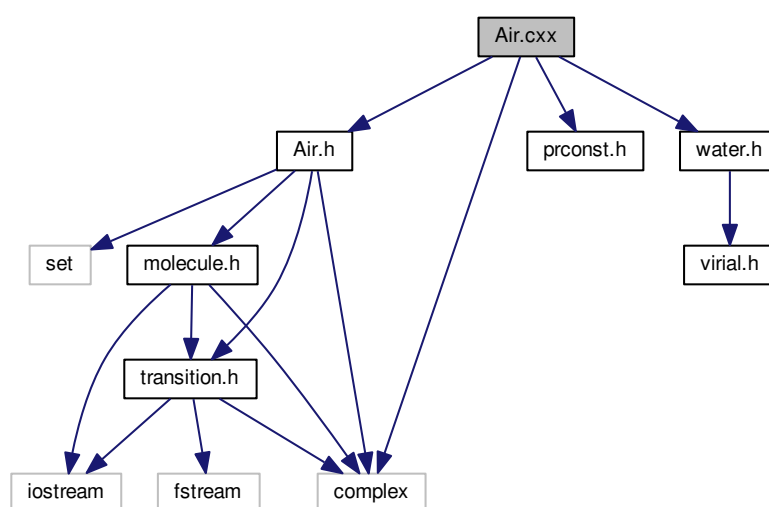
6.10.3.2 `double water::T` [private]

6.10.3.3 `virial* water::vir` [private]

7 File Documentation

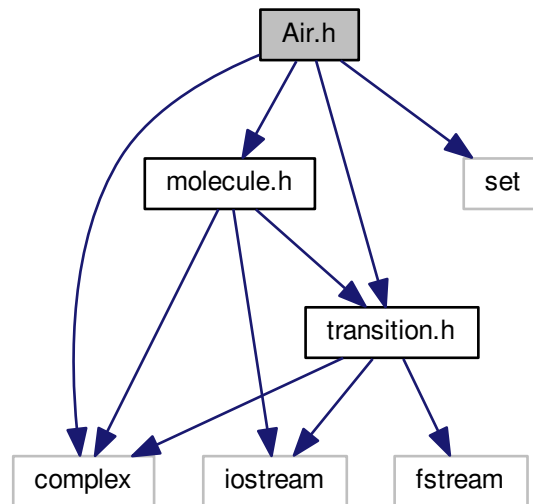
7.1 Air.cxx File Reference

Include dependency graph for Air.cxx:

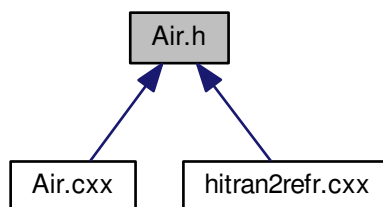


7.2 Air.h File Reference

Include dependency graph for Air.h:



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



Data Structures

- class [Air](#)
- struct [Air::compon](#)

Macros

- #define [KELVIN2CELS](#)(k) ((k)-273.15)
- #define [MMHG2PA](#)(q) (133.322*(q))
- #define [PA2MMHG](#)(p) ((p)/133.322)
- #define [ATOMS_OFFSET](#) 100

7.2.1 Macro Definition Documentation

7.2.1.1 #define ATOMS_OFFSET 100

7.2.1.2 #define KELVIN2CELS(*k*)((*k*)-273.15)

convert Kelvin to Celsius

7.2.1.3 #define MMHG2PA(*q*)(133.322*(*q*))

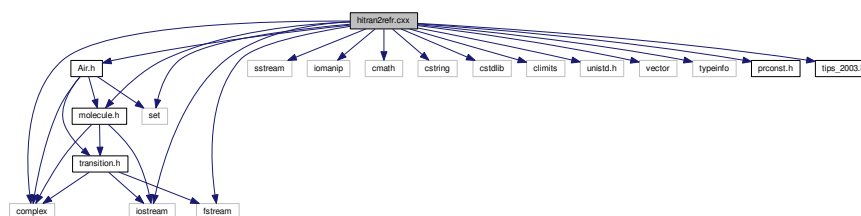
convert mm Hg to Pascal

7.2.1.4 #define PA2MMHG(*p*)((*p*)/133.322)

convert Pascal to mm Hg

7.3 hitran2refr.cxx File Reference

Include dependency graph for hitran2refr.cxx:



Macros

- #define [USE_MAGNETISM](#)
- #define [USE_QOFT](#) 296
- #define [MICRON2THZ](#)(*m*) ([VACUUMC](#)*1.e-6/(*m*))
- #define [WN2MICRON](#)(*w*) (1.e4/(*w*))
- #define [MICRON2WN](#)(*m*) (1.e4/(*m*))
- #define [CELS2KELVIN](#)(*c*) ((*c*)+273.15)

Functions

- void [usage](#) (char *argv0)
- int [main](#) (int argc, char *argv[])

7.3.1 Macro Definition Documentation

7.3.1.1 #define CELS2KELVIN(*c*)((*c*)+273.15)

convert Celsius to Kelvin

7.3.1.2 #define MICRON2THZ(*m*)([VACUUMC](#)*1.e-6/(*m*))

convert micron to Terahertz

7.3.1.3 #define MICRON2WN(*m*)(1.e4/(*m*))

convert micron to wavenumber

7.3.1.4 #define USE_MAGNETISM

7.3.1.5 #define USE_QOFT 296

7.3.1.6 #define WN2MICRON(*w*)(1.e4/(*w*))

convert wavenumber to micron

7.3.2 Function Documentation

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

Invocation from the UNIX shell.

7.3.3 Usage

Parameters

in	<i>argc</i>	argument count, including command line options.
in	<i>argv</i>	the vector of all command line arguments.

The general call consists of a number of options followed by two floating point arguments which are a spectral region in either wavelength or wavenumber units, and a list of file names in any of the few databases which correspond to molecular species.

```
hitran2refr [options] range_low range_hi molec1 [molec2 ...]
```

7.3.3.1 line options

-w If the option '-w' is used, the command line numbers of the spectral interval mean wavenumbers (1/cm), not wavelengths (micron). Also, the output is in wavenumbers, not microns.

-h <relhum> If the option '-h' followed by a number between 0 and 99 is provided, this means a relative humidity in percent. The default is dry air (zero percent)

-s <sampl> If the option '-s' followed by a positive number is provided, this number is the number of sampling points on the wavelength (or with the -w option wavenumber) axis. The default is 2000.

-p If this option '-p' is added, the mixing ratios of all gas components will be adjusted to ensure that the gas pressures adds to the full pressure, even though the mixing ratios in the internal table do not add exactly to 100 percent.

-P If this option '-P' followed by a number is added, the number indicates the full gas pressure in units of Pa. The default is 75058.

-C The option '-C' followed by a floating point number specifies a volume density of the carbon dioxide in units of ppmv. The default is 370.

-T The option '-T' followed by a floating point number specifies the air temperature in units of Celsius.

-V disable the use of virial expansions. Use only the ideal gas law. The default is to use virial expansions.

-D The option '-D' followed by a directory name is the name of the directory which contains the directories HITRAN, spec and cfa with the databases of line transitions. This prime use is if the current working directory is not the directory of the original executable, where the databases are at the original place. The default is "." (which indicates the directories ../HITRAN, ../spec and so on contain this information).

Returns

0 on success and 1 on failure. Failure indicates either use of invalid command line options or command line parameters that are out of reasonable bounds.

7.3.3.2 Examples

Compute real and imaginary part of $n-1$ for a relative humidity of 20%, a temperature of 23 C, a total pressure of 750 hPa, and 1000 points for wavelengths from 3 to 6 micron, using the hitran water line list in `011_hit06.par`, adding the transitions of the additional line list in `h2ofvals.txt`, the HITRAN nitrogen line list in `22_hit04.par`, the nitrogen line list in `n2fvals.txt` etc. Filter lines with an IEEE NaN output and create the ASCII file `tst.dat` with the table.

```
hitran2refr -s 1000 -h 20. -T 23 -p -P 75000 3.0 6.0 011_hit06.par h2ofvals.txt
           22_hit04.par
n2fvals.txt 031_hit04.par 02_hit04.par co2fvals.txt 06_hit04.par 05_hit04.par
           07_hit04.par
o2fvals.txt arfvals.txt nefvals.txt |sed '/NaN/d' | sed '/nan/d' > tst.dat
```

default is that h2o is set by relative humidity, not mixing ratio.

If the number of sampling points on the wavelength or wavenumber axis had not been provided by a command-line option, we chose some default.

The two command line arguments following the options are interpreted as the limits of the wavenumber or wavelength range of interest.

All command line arguments that follow the two number that specify the spectral region are interpreted as individual base file names of one of the databases, and are used to initialize the [Air](#) model with temperature, pressure, relative humidity, CO2 content.

7.3.3.3 void usage (char * argv0)

Print a short usage hint of the main program on standard output.

7.4 hitran2refr.tcl File Reference

Functions

- [hasdry](#) msel
- [go](#)

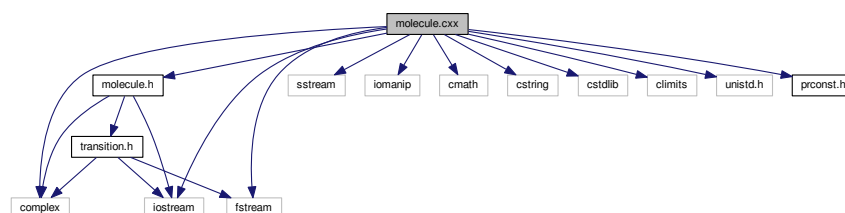
7.4.1 Function Documentation

7.4.1.1 go

7.4.1.2 hasdry msel

7.5 molecule.cxx File Reference

Include dependency graph for `molecule.cxx`:



Macros

- #define `WN2MICRON(w)` (1.e4/(w))
- #define `MICRON2OMEGA(m)` (2.*M_PI*1.e6*VACUUMC/(m))
- #define `HITRAN2000DIR` "HITRAN/HITRAN2004/By-Molecule/"
- #define `SPECJPLDIR` "spec/catalog/"
- #define `CFAWWWDIR` "cfa/"
- #define `JCP111DIR` "hitran_rjm/"
- #define `GEISADIR` "GEISA/"
- #define `GEISALL` 255

7.5.1 Macro Definition Documentation

7.5.1.1 #define CFAWWWDIR "cfa/"

Directory of the Cfa data base. This contains files named arvals.txt, c2ofvals.txt and so on.

See Also

www.cfa.harvard.edu.

7.5.1.2 #define GEISADIR "GEISA/"

Directory of the GEISA data base. This contains files named geisa2003_01.gs and so on. The newer GEISA-09 format shows files with names like line_GEISA2009_asc_gs08...

See Also

ara.lmd.polytechnique.fr.

7.5.1.3 #define GEISALL 255

Maximum number of bytes per GEISA 2009 input line (Fortran total 252), which is followed by
or

, then \0 In case of providing space for input data, this is to be considered larger than HITRANLL

7.5.1.4 #define HITRAN2000DIR "HITRAN/HITRAN2004/By-Molecule/"

Directory of the HITRAN2004 data base files. If not an absolute path name, it is interpreted relativ to the executable. This contains files named 02_hit04.par, 04_hit06.par and so on.

See Also

www.hitran.com.

7.5.1.5 #define JCP111DIR "hitran_rjm/"

Directory of the other data bases retrieved from other pieces of literature

7.5.1.6 #define MICRON2OMEGA(m) (2.*M_PI*1.e6*VACUUMC/(m))

convert micron to angular frequency

7.5.1.7 #define SPECJPLDIR "spec/catalog/"

Directory of the JPL data base. This contains files named c018003.cat and so on.

See Also

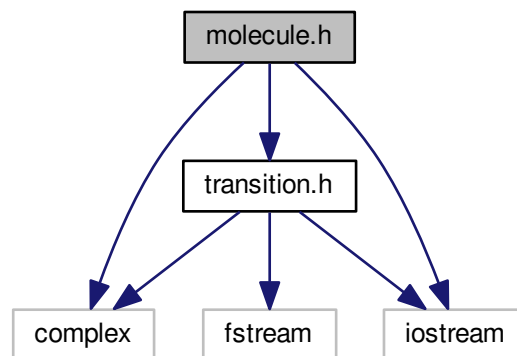
spec.jpl.nasa.gov.

7.5.1.8 `#define WN2MICRON(w) (1.e4/(w))`

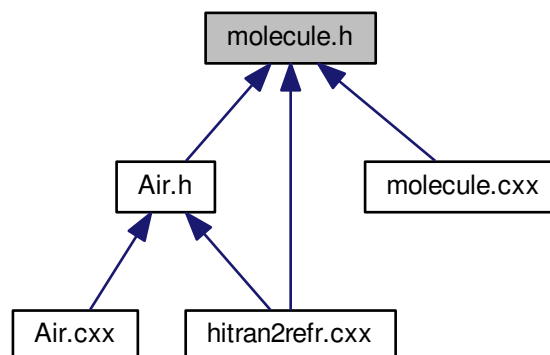
convert wavenumber to micron

7.6 molecule.h File Reference

Include dependency graph for molecule.h:



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

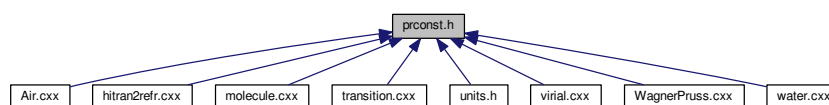


Data Structures

- class [molecule](#)

7.7 prconst.h File Reference

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



Macros

- #define [AVOGADRO](#) 6.02214199e23
- #define [RMOLAR](#) 8.314472
- #define [VACUUMC](#) 2.99792458e+8
- #define [HARTREE](#) 27.2113961
- #define [ECHARGE](#) 1.602176462e-19
- #define [EPSILONZERO](#) 8.854187817e-12
- #define [HBAR](#) 1.05457168e-34
- #define [EMASS](#) 9.10938188e-31
- #define [HC](#) 1.23984244e-4
- #define [PLANCK](#) 6.6260693e-34
- #define [BOLTZMANNK](#) 1.3806505e-23
- #define [WATERMOLWEIGHT](#) 18.015268
- #define [SECPERDAY](#) 86400.

7.7.1 Macro Definition Documentation

7.7.1.1 #define [AVOGADRO](#) 6.02214199e23

7.7.1.2 #define [BOLTZMANNK](#) 1.3806505e-23

7.7.1.3 #define [ECHARGE](#) 1.602176462e-19

7.7.1.4 #define [EMASS](#) 9.10938188e-31

7.7.1.5 #define [EPSILONZERO](#) 8.854187817e-12

7.7.1.6 #define [HARTREE](#) 27.2113961

7.7.1.7 #define [HBAR](#) 1.05457168e-34

7.7.1.8 #define [HC](#) 1.23984244e-4

7.7.1.9 #define [PLANCK](#) 6.6260693e-34

7.7.1.10 #define [RMOLAR](#) 8.314472

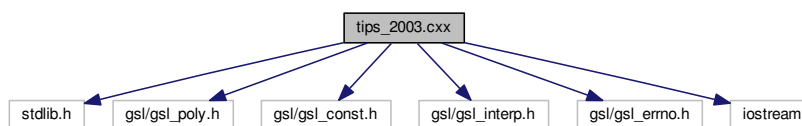
7.7.1.11 #define [SECPERDAY](#) 86400.

7.7.1.12 #define [VACUUMC](#) 2.99792458e+8

7.7.1.13 #define [WATERMOLWEIGHT](#) 18.015268

7.8 tips_2003.cxx File Reference

Include dependency graph for tips_2003.cxx:



Macros

- `#define NT 119`

Functions

- double [AtoB](#) (const double aa, const double *A, const double *B, const size_t npt)
- double [qt_h2O](#) (const double T, const int iso, const double *Tdat, int *gsi)
- double [qt_co2](#) (const double T, const int iso, const double *Tdat, int *gsi)
- double [qt_o3](#) (const double T, const int iso, const double *Tdat, int *gsi)
- double [qt_n2o](#) (const double T, const int iso, const double *Tdat, int *gsi)
- double [qt_co](#) (const double T, const int iso, const double *Tdat, int *gsi)
- double [qt_ch4](#) (const double T, const int iso, const double *Tdat, int *gsi)
- double [qt_o2](#) (const double T, const int iso, const double *Tdat, int *gsi)
- double [qt_no](#) (const double T, const int iso, const double *Tdat, int *gsi)
- double [qt_so2](#) (const double T, const int iso, const double *Tdat, int *gsi)
- double [qt_no2](#) (const double T, const int iso, const double *Tdat, int *gsi)
- double [qt_nh3](#) (const double T, const int iso, const double *Tdat, int *gsi)
- double [qt_hno3](#) (const double T, const int iso, const double *Tdat, int *gsi)
- double [qt_oh](#) (const double T, const int iso, const double *Tdat, int *gsi)
- double [qt_hf](#) (const double T, const int iso, const double *Tdat, int *gsi)
- double [qt_hcl](#) (const double T, const int iso, const double *Tdat, int *gsi)
- double [qt_hbr](#) (const double T, const int iso, const double *Tdat, int *gsi)
- double [qt_hi](#) (const double T, const int iso, const double *Tdat, int *gsi)
- double [qt_clo](#) (const double T, const int iso, const double *Tdat, int *gsi)
- double [qt_ocs](#) (const double T, const int iso, const double *Tdat, int *gsi)
- double [qt_h2co](#) (const double T, const int iso, const double *Tdat, int *gsi)
- double [qt_hocl](#) (const double T, const int iso, const double *Tdat, int *gsi)
- double [qt_n2](#) (const double T, const int iso, const double *Tdat, int *gsi)
- double [qt_hcn](#) (const double T, const int iso, const double *Tdat, int *gsi)
- double [qt_ch3cl](#) (const double T, const int iso, const double *Tdat, int *gsi)
- double [qt_h2o2](#) (const double T, const int iso, const double *Tdat, int *gsi)
- double [qt_c2h2](#) (const double T, const int iso, const double *Tdat, int *gsi)
- double [qt_c2h6](#) (const double T, const int iso, const double *Tdat, int *gsi)
- double [qt_ph3](#) (const double T, const int iso, const double *Tdat, int *gsi)
- double [qt_cof2](#) (const double T, const int iso, const double *Tdat, int *gsi)
- double [qt_sf6](#) (const double T, const int iso, const double *Tdat, int *gsi)
- double [qt_h2s](#) (const double T, const int iso, const double *Tdat, int *gsi)
- double [qt_hcooh](#) (const double T, const int iso, const double *Tdat, int *gsi)
- double [qt_ho2](#) (const double T, const int iso, const double *Tdat, int *gsi)
- double [qt_o](#) (const double T, const int iso, const double *Tdat, int *gsi)

- double [qt_clono2](#) (const double *T*, const int *iso*, const double **Tdat*, int **gsi*)
- double [qt_nop](#) (const double *T*, const int *iso*, const double **Tdat*, int **gsi*)
- double [qt_hobr](#) (const double *T*, const int *iso*, const double **Tdat*, int **gsi*)
- double [qt_c2h4](#) (const double *T*, const int *iso*, const double **Tdat*, int **gsi*)
- double [bd_tips_2003](#) (const int *mol*, const double *temp*, const int *iso*, int **gi*)

7.8.1 Macro Definition Documentation

7.8.1.1 #define NT 119

7.8.2 Function Documentation

7.8.2.1 double AtoB (const double *aa*, const double * *A*, const double * *B*, const size_t *npt*)

7.8.2.2 double bd.tips_2003 (const int *mol*, const double *temp*, const int *iso*, int * *gi*)

TIPS (total internal partition sum.

See Also

[?]

Parameters

in	<i>mol</i>	HITRAN molecule number
in	<i>temp</i>	temperature [K]
in	<i>iso</i>	isotopomer index, starting with 1 as the most abundant isotope as in HITRAN
out	<i>gi</i>	state independent degeneracy factor

Returns

total internal partition sum

Author

R. R. Gamache

Since

2003-12-18 better vibrational fundamentals for PH3

See Also

[TIPSGlh.f90](#)

7.8.2.3 double qt.c2h2 (const double *T*, const int *iso*, const double * *Tdat*, int * *gsi*)

total internal partition function.

Parameters

in	<i>T</i>	temperature [K]
in	<i>iso</i>	isotope code (HITRAN index)
out	<i>gsi</i>	state independent nuclear degeneracyfactor

Returns

Total Internal Partition Function (returned)

7.8.2.4 `double qt_c2h4 (const double T, const int iso, const double * Tdat, int * gsi)`

total internal partition function.

Parameters

in	<i>T</i>	temperature [K]
in	<i>iso</i>	isotope code (HITRAN index)
out	<i>gsi</i>	state independent nuclear degeneracyfactor

Returns

Total Internal Partition Function (returned)

7.8.2.5 `double qt_c2h6 (const double T, const int iso, const double * Tdat, int * gsi)`

total internal partition function.

Parameters

in	<i>T</i>	temperature [K]
in	<i>iso</i>	isotope code (HITRAN index)
out	<i>gsi</i>	state independent nuclear degeneracyfactor

Returns

Total Internal Partition Function (returned)

7.8.2.6 `double qt_ch3cl (const double T, const int iso, const double * Tdat, int * gsi)`

total internal partition function.

Parameters

in	<i>T</i>	temperature [K]
in	<i>iso</i>	isotope code (HITRAN index)
out	<i>gsi</i>	state independent nuclear degeneracyfactor

Returns

Total Internal Partition Function (returned)

7.8.2.7 `double qt_ch4 (const double T, const int iso, const double * Tdat, int * gsi)`

total internal partition function.

Parameters

in	<i>T</i>	temperature [K]
in	<i>iso</i>	isotope code (HITRAN index)
out	<i>gsi</i>	state independent nuclear degeneracyfactor

Returns

Total Internal Partition Function (returned)

7.8.2.8 `double qt_clo (const double T, const int iso, const double * Tdat, int * gsi)`

total internal partition function.

Parameters

in	<i>T</i>	temperature [K]
in	<i>iso</i>	isotope code (HITRAN index)
out	<i>gsi</i>	state independent nuclear degeneracyfactor

Returns

Total Internal Partition Function (returned)

7.8.2.9 `double qt_clono2 (const double T, const int iso, const double * Tdat, int * gsi)`

total internal partition function.

Parameters

in	<i>T</i>	temperature [K]
in	<i>iso</i>	isotope code (HITRAN index)
out	<i>gsi</i>	state independent nuclear degeneracyfactor

Returns

Total Internal Partition Function (returned)

7.8.2.10 `double qt_co (const double T, const int iso, const double * Tdat, int * gsi)`

total internal partition function.

Parameters

in	<i>T</i>	temperature [K]
in	<i>iso</i>	isotope code (HITRAN index)
out	<i>gsi</i>	state independent nuclear degeneracyfactor

Returns

Total Internal Partition Function (returned)

7.8.2.11 `double qt_co2 (const double T, const int iso, const double * Tdat, int * gsi)`

total internal partition function.

Parameters

in	<i>T</i>	temperature [K]
in	<i>iso</i>	isotope code (HITRAN index)
out	<i>gsi</i>	state independent nuclear degeneracyfactor

Returns

Total Internal Partition Function (returned)

7.8.2.12 `double qt_cof2 (const double T, const int iso, const double * Tdat, int * gsi)`

total internal partition function.

Parameters

in	<i>T</i>	temperature [K]
in	<i>iso</i>	isotope code (HITRAN index)
out	<i>gsi</i>	state independent nuclear degeneracyfactor

Returns

Total Internal Partition Function (returned)

7.8.2.13 `double qt_h2co (const double T, const int iso, const double * Tdat, int * gsi)`

total internal partition function.

Parameters

in	<i>T</i>	temperature [K]
in	<i>iso</i>	isotope code (HITRAN index)
out	<i>gsi</i>	state independent nuclear degeneracyfactor

Returns

Total Internal Partition Function (returned)

7.8.2.14 `double qt_h2o (const double T, const int iso, const double * Tdat, int * gsi)`

total internal partition function.

Parameters

in	<i>T</i>	temperature [K]
in	<i>iso</i>	isotope code (HITRAN index)
out	<i>gsi</i>	state independent nuclear degeneracyfactor

Returns

Total Internal Partition Function (returned)

7.8.2.15 `double qt_h2o2 (const double T, const int iso, const double * Tdat, int * gsi)`

total internal partition function.

Parameters

in	<i>T</i>	temperature [K]
in	<i>iso</i>	isotope code (HITRAN index)
out	<i>gsi</i>	state independent nuclear degeneracyfactor

Returns

Total Internal Partition Function (returned)

7.8.2.16 `double qt_h2s (const double T, const int iso, const double * Tdat, int * gsi)`

total internal partition function.

Parameters

in	<i>T</i>	temperature [K]
in	<i>iso</i>	isotope code (HITRAN index)
out	<i>gsi</i>	state independent nuclear degeneracyfactor

Returns

Total Internal Partition Function (returned)

7.8.2.17 `double qt_hbr (const double T, const int iso, const double * Tdat, int * gsi)`

total internal partition function.

Parameters

in	<i>T</i>	temperature [K]
in	<i>iso</i>	isotope code (HITRAN index)
out	<i>gsi</i>	state independent nuclear degeneracyfactor

Returns

Total Internal Partition Function (returned)

7.8.2.18 `double qt_hcl (const double T, const int iso, const double * Tdat, int * gsi)`

total internal partition function.

Parameters

in	<i>T</i>	temperature [K]
in	<i>iso</i>	isotope code (HITRAN index)
out	<i>gsi</i>	state independent nuclear degeneracyfactor

Returns

Total Internal Partition Function (returned)

7.8.2.19 `double qt_hcn (const double T, const int iso, const double * Tdat, int * gsi)`

total internal partition function.

Parameters

in	<i>T</i>	temperature [K]
in	<i>iso</i>	isotope code (HITRAN index)
out	<i>gsi</i>	state independent nuclear degeneracyfactor

Returns

Total Internal Partition Function (returned)

7.8.2.20 `double qt_hcooh (const double T, const int iso, const double * Tdat, int * gsi)`

total internal partition function.

Parameters

in	<i>T</i>	temperature [K]
in	<i>iso</i>	isotope code (HITRAN index)
out	<i>gsi</i>	state independent nuclear degeneracyfactor

Returns

Total Internal Partition Function (returned)

7.8.2.21 `double qt_hf (const double T, const int iso, const double * Tdat, int * gsi)`

total internal partition function.

Parameters

in	<i>T</i>	temperature [K]
in	<i>iso</i>	isotope code (HITRAN index)
out	<i>gsi</i>	state independent nuclear degeneracyfactor

Returns

Total Internal Partition Function (returned)

7.8.2.22 `double qt_hi (const double T, const int iso, const double * Tdat, int * gsi)`

total internal partition function.

Parameters

in	<i>T</i>	temperature [K]
in	<i>iso</i>	isotope code (HITRAN index)
out	<i>gsi</i>	state independent nuclear degeneracyfactor

Returns

Total Internal Partition Function (returned)

7.8.2.23 `double qt_hno3 (const double T, const int iso, const double * Tdat, int * gsi)`

total internal partition function.

Parameters

in	<i>T</i>	temperature [K]
in	<i>iso</i>	isotope code (HITRAN index)
out	<i>gsi</i>	state independent nuclear degeneracyfactor

Returns

Total Internal Partition Function (returned)

7.8.2.24 `double qt_ho2 (const double T, const int iso, const double * Tdat, int * gsi)`

total internal partition function.

Parameters

in	<i>T</i>	temperature [K]
in	<i>iso</i>	isotope code (HITRAN index)
out	<i>gsi</i>	state independent nuclear degeneracyfactor

Returns

Total Internal Partition Function (returned)

7.8.2.25 `double qt_hobr (const double T, const int iso, const double * Tdat, int * gsi)`

total internal partition function.

Parameters

in	<i>T</i>	temperature [K]
in	<i>iso</i>	isotope code (HITRAN index)
out	<i>gsi</i>	state independent nuclear degeneracyfactor

Returns

Total Internal Partition Function (returned)

7.8.2.26 `double qt_hocl (const double T, const int iso, const double * Tdat, int * gsi)`

total internal partition function.

Parameters

in	<i>T</i>	temperature [K]
in	<i>iso</i>	isotope code (HITRAN index)
out	<i>gsi</i>	state independent nuclear degeneracyfactor

Returns

Total Internal Partition Function (returned)

7.8.2.27 `double qt_n2 (const double T, const int iso, const double * Tdat, int * gsi)`

total internal partition function.

Parameters

in	<i>T</i>	temperature [K]
in	<i>iso</i>	isotope code (HITRAN index)
out	<i>gsi</i>	state independent nuclear degeneracyfactor

Returns

Total Internal Partition Function (returned)

7.8.2.28 `double qt_n2o (const double T, const int iso, const double * Tdat, int * gsi)`

total internal partition function.

Parameters

in	<i>T</i>	temperature [K]
in	<i>iso</i>	isotope code (HITRAN index)
out	<i>gsi</i>	state independent nuclear degeneracyfactor

Returns

Total Internal Partition Function (returned)

7.8.2.29 `double qt_nh3 (const double T, const int iso, const double * Tdat, int * gsi)`

total internal partition function.

Parameters

in	<i>T</i>	temperature [K]
in	<i>iso</i>	isotope code (HITRAN index)
out	<i>gsi</i>	state independent nuclear degeneracyfactor

Returns

Total Internal Partition Function (returned)

7.8.2.30 `double qt_no (const double T, const int iso, const double * Tdat, int * gsi)`

total internal partition function.

Parameters

in	<i>T</i>	temperature [K]
in	<i>iso</i>	isotope code (HITRAN index)
out	<i>gsi</i>	state independent nuclear degeneracyfactor

Returns

Total Internal Partition Function (returned)

7.8.2.31 `double qt_no2 (const double T, const int iso, const double * Tdat, int * gsi)`

total internal partition function.

Parameters

in	<i>T</i>	temperature [K]
in	<i>iso</i>	isotope code (HITRAN index)
out	<i>gsi</i>	state independent nuclear degeneracyfactor

Returns

Total Internal Partition Function (returned)

7.8.2.32 `double qt_nop (const double T, const int iso, const double * Tdat, int * gsi)`

total internal partition function.

Parameters

in	<i>T</i>	temperature [K]
in	<i>iso</i>	isotope code (HITRAN index)
out	<i>gsi</i>	state independent nuclear degeneracyfactor

Returns

Total Internal Partition Function (returned)

7.8.2.33 `double qt_o (const double T, const int iso, const double * Tdat, int * gsi)`

total internal partition function.

Parameters

in	<i>T</i>	temperature [K]
in	<i>iso</i>	isotope code (HITRAN index)
out	<i>gsi</i>	state independent nuclear degeneracyfactor

Returns

Total Internal Partition Function (returned)

7.8.2.34 `double qt_o2 (const double T, const int iso, const double * Tdat, int * gsi)`

total internal partition function.

Parameters

in	<i>T</i>	temperature [K]
in	<i>iso</i>	isotope code (HITRAN index)
out	<i>gsi</i>	state independent nuclear degeneracyfactor

Returns

Total Internal Partition Function (returned)

7.8.2.35 `double qt_o3 (const double T, const int iso, const double * Tdat, int * gsi)`

total internal partition function.

Parameters

in	<i>T</i>	temperature [K]
in	<i>iso</i>	isotope code (HITRAN index)
out	<i>gsi</i>	state independent nuclear degeneracyfactor

Returns

Total Internal Partition Function (returned)

7.8.2.36 `double qt_ocs (const double T, const int iso, const double * Tdat, int * gsi)`

total internal partition function.

Parameters

in	<i>T</i>	temperature [K]
in	<i>iso</i>	isotope code (HITRAN index)
out	<i>gsi</i>	state independent nuclear degeneracyfactor

Returns

Total Internal Partition Function (returned)

7.8.2.37 `double qt_oh (const double T, const int iso, const double * Tdat, int * gsi)`

total internal partition function.

Parameters

in	<i>T</i>	temperature [K]
in	<i>iso</i>	isotope code (HITRAN index)
out	<i>gsi</i>	state independent nuclear degeneracyfactor

Returns

Total Internal Partition Function (returned)

7.8.2.38 `double qt_ph3 (const double T, const int iso, const double * Tdat, int * gsi)`

total internal partition function.

Parameters

in	<i>T</i>	temperature [K]
in	<i>iso</i>	isotope code (HITRAN index)
out	<i>gsi</i>	state independent nuclear degeneracyfactor

Returns

Total Internal Partition Function (returned)

7.8.2.39 `double qt_sf6 (const double T, const int iso, const double * Tdat, int * gsi)`

total internal partition function.

Parameters

in	<i>T</i>	temperature [K]
in	<i>iso</i>	isotope code (HITRAN index)
out	<i>gsi</i>	state independent nuclear degeneracyfactor

Returns

Total Internal Partition Function (returned)

7.8.2.40 `double qt_so2 (const double T, const int iso, const double * Tdat, int * gsi)`

total internal partition function.

Parameters

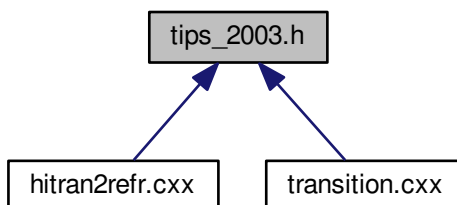
in	<i>T</i>	temperature [K]
in	<i>iso</i>	isotope code (HITRAN index)
out	<i>gsi</i>	state independent nuclear degeneracyfactor

Returns

Total Internal Partition Function (returned)

7.9 tips_2003.h File Reference

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

**Functions**

- double [AtoB](#) (const double aa, const double *A, const double *B, const size_t npt)
- double [qt_h2O](#) (const double T, const int iso, const double *Tdat, int *gsi)
- double [qt_co2](#) (const double T, const int iso, const double *Tdat, int *gsi)
- double [qt_o3](#) (const double T, const int iso, const double *Tdat, int *gsi)
- double [qt_n2o](#) (const double T, const int iso, const double *Tdat, int *gsi)
- double [qt_co](#) (const double T, const int iso, const double *Tdat, int *gsi)
- double [qt_ch4](#) (const double T, const int iso, const double *Tdat, int *gsi)
- double [qt_o2](#) (const double T, const int iso, const double *Tdat, int *gsi)
- double [qt_no](#) (const double T, const int iso, const double *Tdat, int *gsi)
- double [qt_so2](#) (const double T, const int iso, const double *Tdat, int *gsi)
- double [qt_no2](#) (const double T, const int iso, const double *Tdat, int *gsi)
- double [qt_nh3](#) (const double T, const int iso, const double *Tdat, int *gsi)
- double [qt_hno3](#) (const double T, const int iso, const double *Tdat, int *gsi)
- double [qt_oh](#) (const double T, const int iso, const double *Tdat, int *gsi)
- double [qt_hf](#) (const double T, const int iso, const double *Tdat, int *gsi)
- double [qt_hcl](#) (const double T, const int iso, const double *Tdat, int *gsi)
- double [qt_hbr](#) (const double T, const int iso, const double *Tdat, int *gsi)

- double `qt_hi` (const double T, const int iso, const double *Tdat, int *gsi)
- double `qt_clo` (const double T, const int iso, const double *Tdat, int *gsi)
- double `qt_ocs` (const double T, const int iso, const double *Tdat, int *gsi)
- double `qt_h2co` (const double T, const int iso, const double *Tdat, int *gsi)
- double `qt_hocl` (const double T, const int iso, const double *Tdat, int *gsi)
- double `qt_n2` (const double T, const int iso, const double *Tdat, int *gsi)
- double `qt_hcn` (const double T, const int iso, const double *Tdat, int *gsi)
- double `qt_ch3cl` (const double T, const int iso, const double *Tdat, int *gsi)
- double `qt_h2o2` (const double T, const int iso, const double *Tdat, int *gsi)
- double `qt_c2h2` (const double T, const int iso, const double *Tdat, int *gsi)
- double `qt_c2h6` (const double T, const int iso, const double *Tdat, int *gsi)
- double `qt_ph3` (const double T, const int iso, const double *Tdat, int *gsi)
- double `qt_cof2` (const double T, const int iso, const double *Tdat, int *gsi)
- double `qt_sf6` (const double T, const int iso, const double *Tdat, int *gsi)
- double `qt_h2s` (const double T, const int iso, const double *Tdat, int *gsi)
- double `qt_hcooh` (const double T, const int iso, const double *Tdat, int *gsi)
- double `qt_ho2` (const double T, const int iso, const double *Tdat, int *gsi)
- double `qt_o` (const double T, const int iso, const double *Tdat, int *gsi)
- double `qt_clono2` (const double T, const int iso, const double *Tdat, int *gsi)
- double `qt_nop` (const double T, const int iso, const double *Tdat, int *gsi)
- double `qt_hobr` (const double T, const int iso, const double *Tdat, int *gsi)
- double `qt_c2h4` (const double T, const int iso, const double *Tdat, int *gsi)
- double `bd_tips_2003` (const int mol, const double temp, const int iso, int *gi)

7.9.1 Function Documentation

7.9.1.1 double `AtoB` (const double *aa*, const double * *A*, const double * *B*, const size_t *npt*)

7.9.1.2 double `bd.tips.2003` (const int *mol*, const double *temp*, const int *iso*, int * *gi*)

TIPS (total internal partition sum.

See Also

[?]

Parameters

in	<i>mol</i>	HITRAN molecule number
in	<i>temp</i>	temperature [K]
in	<i>iso</i>	isotopomer index, starting with 1 as the most abundant isotope as in HITRAN
out	<i>gi</i>	state independent degeneracy factor

Returns

total internal partition sum

Author

R. R. Gamache

Since

2003-12-18 better vibrational fundamentals for PH3

See Also

[TIPsglh.f90](#)

7.9.1.3 `double qt_c2h2 (const double T, const int iso, const double * Tdat, int * gsi)`

total internal partition function.

Parameters

in	<i>T</i>	temperature [K]
in	<i>iso</i>	isotope code (HITRAN index)
out	<i>gsi</i>	state independent nuclear degeneracyfactor

Returns

Total Internal Partition Function (returned)

7.9.1.4 `double qt_c2h4 (const double T, const int iso, const double * Tdat, int * gsi)`

total internal partition function.

Parameters

in	<i>T</i>	temperature [K]
in	<i>iso</i>	isotope code (HITRAN index)
out	<i>gsi</i>	state independent nuclear degeneracyfactor

Returns

Total Internal Partition Function (returned)

7.9.1.5 `double qt_c2h6 (const double T, const int iso, const double * Tdat, int * gsi)`

total internal partition function.

Parameters

in	<i>T</i>	temperature [K]
in	<i>iso</i>	isotope code (HITRAN index)
out	<i>gsi</i>	state independent nuclear degeneracyfactor

Returns

Total Internal Partition Function (returned)

7.9.1.6 `double qt_ch3cl (const double T, const int iso, const double * Tdat, int * gsi)`

total internal partition function.

Parameters

in	<i>T</i>	temperature [K]
in	<i>iso</i>	isotope code (HITRAN index)
out	<i>gsi</i>	state independent nuclear degeneracyfactor

Returns

Total Internal Partition Function (returned)

7.9.1.7 `double qt_ch4 (const double T, const int iso, const double * Tdat, int * gsi)`

total internal partition function.

Parameters

in	<i>T</i>	temperature [K]
in	<i>iso</i>	isotope code (HITRAN index)
out	<i>gsi</i>	state independent nuclear degeneracyfactor

Returns

Total Internal Partition Function (returned)

7.9.1.8 `double qt_clo (const double T, const int iso, const double * Tdat, int * gsi)`

total internal partition function.

Parameters

in	<i>T</i>	temperature [K]
in	<i>iso</i>	isotope code (HITRAN index)
out	<i>gsi</i>	state independent nuclear degeneracyfactor

Returns

Total Internal Partition Function (returned)

7.9.1.9 `double qt_clono2 (const double T, const int iso, const double * Tdat, int * gsi)`

total internal partition function.

Parameters

in	<i>T</i>	temperature [K]
in	<i>iso</i>	isotope code (HITRAN index)
out	<i>gsi</i>	state independent nuclear degeneracyfactor

Returns

Total Internal Partition Function (returned)

7.9.1.10 `double qt_co (const double T, const int iso, const double * Tdat, int * gsi)`

total internal partition function.

Parameters

in	<i>T</i>	temperature [K]
in	<i>iso</i>	isotope code (HITRAN index)
out	<i>gsi</i>	state independent nuclear degeneracyfactor

Returns

Total Internal Partition Function (returned)

7.9.1.11 `double qt_co2 (const double T, const int iso, const double * Tdat, int * gsi)`

total internal partition function.

Parameters

in	<i>T</i>	temperature [K]
in	<i>iso</i>	isotope code (HITRAN index)
out	<i>gsi</i>	state independent nuclear degeneracyfactor

Returns

Total Internal Partition Function (returned)

7.9.1.12 `double qt_cof2 (const double T, const int iso, const double * Tdat, int * gsi)`

total internal partition function.

Parameters

in	<i>T</i>	temperature [K]
in	<i>iso</i>	isotope code (HITRAN index)
out	<i>gsi</i>	state independent nuclear degeneracyfactor

Returns

Total Internal Partition Function (returned)

7.9.1.13 `double qt_h2co (const double T, const int iso, const double * Tdat, int * gsi)`

total internal partition function.

Parameters

in	<i>T</i>	temperature [K]
in	<i>iso</i>	isotope code (HITRAN index)
out	<i>gsi</i>	state independent nuclear degeneracyfactor

Returns

Total Internal Partition Function (returned)

7.9.1.14 `double qt_h2O (const double T, const int iso, const double * Tdat, int * gsi)`

total internal partition function.

Parameters

in	<i>T</i>	temperature [K]
in	<i>iso</i>	isotope code (HITRAN index)
out	<i>gsi</i>	state independent nuclear degeneracyfactor

Returns

Total Internal Partition Function (returned)

7.9.1.15 `double qt_h2o2 (const double T, const int iso, const double * Tdat, int * gsi)`

total internal partition function.

Parameters

in	<i>T</i>	temperature [K]
in	<i>iso</i>	isotope code (HITRAN index)
out	<i>gsi</i>	state independent nuclear degeneracyfactor

Returns

Total Internal Partition Function (returned)

7.9.1.16 `double qt_h2s (const double T, const int iso, const double * Tdat, int * gsi)`

total internal partition function.

Parameters

in	<i>T</i>	temperature [K]
in	<i>iso</i>	isotope code (HITRAN index)
out	<i>gsi</i>	state independent nuclear degeneracyfactor

Returns

Total Internal Partition Function (returned)

7.9.1.17 `double qt_hbr (const double T, const int iso, const double * Tdat, int * gsi)`

total internal partition function.

Parameters

in	<i>T</i>	temperature [K]
in	<i>iso</i>	isotope code (HITRAN index)
out	<i>gsi</i>	state independent nuclear degeneracyfactor

Returns

Total Internal Partition Function (returned)

7.9.1.18 `double qt_hcl (const double T, const int iso, const double * Tdat, int * gsi)`

total internal partition function.

Parameters

in	<i>T</i>	temperature [K]
in	<i>iso</i>	isotope code (HITRAN index)
out	<i>gsi</i>	state independent nuclear degeneracyfactor

Returns

Total Internal Partition Function (returned)

7.9.1.19 `double qt_hcn (const double T, const int iso, const double * Tdat, int * gsi)`

total internal partition function.

Parameters

in	<i>T</i>	temperature [K]
in	<i>iso</i>	isotope code (HITRAN index)
out	<i>gsi</i>	state independent nuclear degeneracyfactor

Returns

Total Internal Partition Function (returned)

7.9.1.20 `double qt_hcooh (const double T, const int iso, const double * Tdat, int * gsi)`

total internal partition function.

Parameters

in	<i>T</i>	temperature [K]
in	<i>iso</i>	isotope code (HITRAN index)
out	<i>gsi</i>	state independent nuclear degeneracyfactor

Returns

Total Internal Partition Function (returned)

7.9.1.21 `double qt_hf (const double T, const int iso, const double * Tdat, int * gsi)`

total internal partition function.

Parameters

in	<i>T</i>	temperature [K]
in	<i>iso</i>	isotope code (HITRAN index)
out	<i>gsi</i>	state independent nuclear degeneracyfactor

Returns

Total Internal Partition Function (returned)

7.9.1.22 `double qt_hi (const double T, const int iso, const double * Tdat, int * gsi)`

total internal partition function.

Parameters

in	<i>T</i>	temperature [K]
in	<i>iso</i>	isotope code (HITRAN index)
out	<i>gsi</i>	state independent nuclear degeneracyfactor

Returns

Total Internal Partition Function (returned)

7.9.1.23 `double qt_hno3 (const double T, const int iso, const double * Tdat, int * gsi)`

total internal partition function.

Parameters

in	<i>T</i>	temperature [K]
in	<i>iso</i>	isotope code (HITRAN index)
out	<i>gsi</i>	state independent nuclear degeneracyfactor

Returns

Total Internal Partition Function (returned)

7.9.1.24 `double qt_ho2 (const double T, const int iso, const double * Tdat, int * gsi)`

total internal partition function.

Parameters

in	<i>T</i>	temperature [K]
in	<i>iso</i>	isotope code (HITRAN index)
out	<i>gsi</i>	state independent nuclear degeneracyfactor

Returns

Total Internal Partition Function (returned)

7.9.1.25 `double qt_hobr (const double T, const int iso, const double * Tdat, int * gsi)`

total internal partition function.

Parameters

in	<i>T</i>	temperature [K]
in	<i>iso</i>	isotope code (HITRAN index)
out	<i>gsi</i>	state independent nuclear degeneracyfactor

Returns

Total Internal Partition Function (returned)

7.9.1.26 `double qt_hocl (const double T, const int iso, const double * Tdat, int * gsi)`

total internal partition function.

Parameters

in	<i>T</i>	temperature [K]
in	<i>iso</i>	isotope code (HITRAN index)
out	<i>gsi</i>	state independent nuclear degeneracyfactor

Returns

Total Internal Partition Function (returned)

7.9.1.27 `double qt_n2 (const double T, const int iso, const double * Tdat, int * gsi)`

total internal partition function.

Parameters

in	<i>T</i>	temperature [K]
in	<i>iso</i>	isotope code (HITRAN index)
out	<i>gsi</i>	state independent nuclear degeneracyfactor

Returns

Total Internal Partition Function (returned)

7.9.1.28 `double qt_n2o (const double T, const int iso, const double * Tdat, int * gsi)`

total internal partition function.

Parameters

in	<i>T</i>	temperature [K]
in	<i>iso</i>	isotope code (HITRAN index)
out	<i>gsi</i>	state independent nuclear degeneracyfactor

Returns

Total Internal Partition Function (returned)

7.9.1.29 `double qt_nh3 (const double T, const int iso, const double * Tdat, int * gsi)`

total internal partition function.

Parameters

in	<i>T</i>	temperature [K]
in	<i>iso</i>	isotope code (HITRAN index)
out	<i>gsi</i>	state independent nuclear degeneracyfactor

Returns

Total Internal Partition Function (returned)

7.9.1.30 `double qt_no (const double T, const int iso, const double * Tdat, int * gsi)`

total internal partition function.

Parameters

in	<i>T</i>	temperature [K]
in	<i>iso</i>	isotope code (HITRAN index)
out	<i>gsi</i>	state independent nuclear degeneracyfactor

Returns

Total Internal Partition Function (returned)

7.9.1.31 `double qt_no2 (const double T, const int iso, const double * Tdat, int * gsi)`

total internal partition function.

Parameters

in	<i>T</i>	temperature [K]
in	<i>iso</i>	isotope code (HITRAN index)
out	<i>gsi</i>	state independent nuclear degeneracyfactor

Returns

Total Internal Partition Function (returned)

7.9.1.32 `double qt_nop (const double T, const int iso, const double * Tdat, int * gsi)`

total internal partition function.

Parameters

in	<i>T</i>	temperature [K]
in	<i>iso</i>	isotope code (HITRAN index)
out	<i>gsi</i>	state independent nuclear degeneracyfactor

Returns

Total Internal Partition Function (returned)

7.9.1.33 `double qt_o (const double T, const int iso, const double * Tdat, int * gsi)`

total internal partition function.

Parameters

in	<i>T</i>	temperature [K]
in	<i>iso</i>	isotope code (HITRAN index)
out	<i>gsi</i>	state independent nuclear degeneracyfactor

Returns

Total Internal Partition Function (returned)

7.9.1.34 `double qt_o2 (const double T, const int iso, const double * Tdat, int * gsi)`

total internal partition function.

Parameters

in	<i>T</i>	temperature [K]
in	<i>iso</i>	isotope code (HITRAN index)
out	<i>gsi</i>	state independent nuclear degeneracyfactor

Returns

Total Internal Partition Function (returned)

7.9.1.35 `double qt_o3 (const double T, const int iso, const double * Tdat, int * gsi)`

total internal partition function.

Parameters

in	<i>T</i>	temperature [K]
in	<i>iso</i>	isotope code (HITRAN index)
out	<i>gsi</i>	state independent nuclear degeneracyfactor

Returns

Total Internal Partition Function (returned)

7.9.1.36 `double qt_ocs (const double T, const int iso, const double * Tdat, int * gsi)`

total internal partition function.

Parameters

in	<i>T</i>	temperature [K]
in	<i>iso</i>	isotope code (HITRAN index)
out	<i>gsi</i>	state independent nuclear degeneracyfactor

Returns

Total Internal Partition Function (returned)

7.9.1.37 `double qt_oh (const double T, const int iso, const double * Tdat, int * gsi)`

total internal partition function.

Parameters

in	<i>T</i>	temperature [K]
in	<i>iso</i>	isotope code (HITRAN index)
out	<i>gsi</i>	state independent nuclear degeneracyfactor

Returns

Total Internal Partition Function (returned)

7.9.1.38 `double qt_ph3 (const double T, const int iso, const double * Tdat, int * gsi)`

total internal partition function.

Parameters

in	<i>T</i>	temperature [K]
in	<i>iso</i>	isotope code (HITRAN index)
out	<i>gsi</i>	state independent nuclear degeneracyfactor

Returns

Total Internal Partition Function (returned)

7.9.1.39 `double qt_sf (const double T, const int iso, const double * Tdat, int * gsi)`

total internal partition function.

Parameters

in	<i>T</i>	temperature [K]
in	<i>iso</i>	isotope code (HITRAN index)
out	<i>gsi</i>	state independent nuclear degeneracyfactor

Returns

Total Internal Partition Function (returned)

7.9.1.40 `double qt_so2 (const double T, const int iso, const double * Tdat, int * gsi)`

total internal partition function.

Parameters

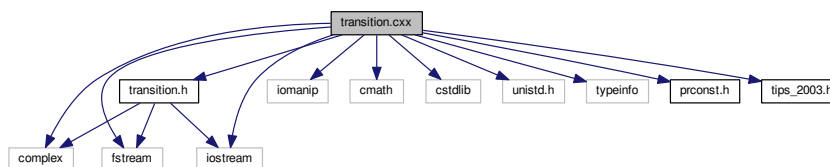
in	<i>T</i>	temperature [K]
in	<i>iso</i>	isotope code (HITRAN index)
out	<i>gsi</i>	state independent nuclear degeneracyfactor

Returns

Total Internal Partition Function (returned)

7.10 transition.cxx File Reference

Include dependency graph for transition.cxx:

**Macros**

- `#define ATOMS_OFFSET 100`
- `#define MHZ2WN(h) (1.e4*(h)/VACUUMC)`
- `#define EV2WN(e) ((e)/HC)`
- `#define WN2HARTREE(w) (HC*(w)/HARTREE)`
- `#define WN2MICRON(w) (1.e4/(w))`
- `#define MICRON2WN(m) (1.e4/(m))`
- `#define WN2OMEGA(w) (2.*M_PI*100.*VACUUMC*(w))`
- `#define WN2HZ(w) (100.*VACUUMC*(w))`

- #define MICRON2OMEGA(m) (2.*M_PI*1.e6*VACUUMC/(m))
- #define DMICRON2DOMECA(m, o) ((m)*(o)*(o)/(2.*M_PI*VACUUMC))

Functions

- template<class TYPE >
istream & getval (istream &is, TYPE &buf, int width)
- void tag2molec (const int jpltag, int &molec, int &isot)

7.10.1 Macro Definition Documentation

7.10.1.1 #define ATOMS_OFFSET 100

7.10.1.2 #define DMICRON2DOMECA(m, o)((m)*(o)*(o)/(2.*M_PI*VACUUMC))

convert differential in micron to differential of angular frequency. $d\omega/d\lambda = 2\pi c/\lambda^2$, $d\omega = d\lambda\omega/\lambda = d\lambda\omega^2/(2\pi c)$

7.10.1.3 #define EV2WN(e)((e)/HC)

convert eV to wavenumber

7.10.1.4 #define MHZ2WN(h) (1.e4*(h)/VACUUMC)

convert Megahertz to wavenumber

7.10.1.5 #define MICRON2OMEGA(m) (2.*M_PI*1.e6*VACUUMC/(m))

convert micron to angular frequency

7.10.1.6 #define MICRON2WN(m) (1.e4/(m))

convert micron to wavenumber

7.10.1.7 #define WN2HARTREE(w) (HC*(w)/HARTREE)

convert wavenumber to hartree

7.10.1.8 #define WN2HZ(w) (100.*VACUUMC*(w))

Convert spectroscopic wavenumber [1/cm] to frequency [Hz]

7.10.1.9 #define WN2MICRON(w) (1.e4/(w))

convert wavenumber to micron

7.10.1.10 #define WN2OMEGA(w) (2.*M_PI*100.*VACUUMC*(w))

convert wavenumber to angular frequency

7.10.2 Function Documentation

7.10.2.1 template<class TYPE > istream& getval (istream & is, TYPE & buf, int width)

Obtain a formatted value from a HITRAN line (input stream). Read a formatted value of the indicated type from the input stream, but only to a maximum number of bytes

Parameters

in, out	<i>is</i>	the input file stream. On exit this is re-positioned to a point <code>width</code> further than the position it had before calling the function.
out	<i>buf</i>	the position to write the values read into
in	<i>width</i>	maximum number of bytes to consider for input. This is the decisive difference to the standard way of reading streams which would swallow everything up to the next blank. This here is a reader made to deal with Fortran-formatted, not necessarily blank-delimited lines of input.

7.10.2.2 void tag2molec (const int *jpltag*, int & *molec*, int & *isot*)

Convert JPL molecular tags to HITRAN molecular numbers.

Parameters

in	<i>jpltag</i>	one of the 5-digit JPL molecular ID's
out	<i>molec</i>	the HITRAN molecular number
out	<i>isot</i>	the HITRAN isotope number

Author

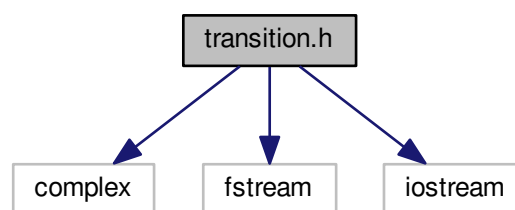
Richard J. Mathar

Warning

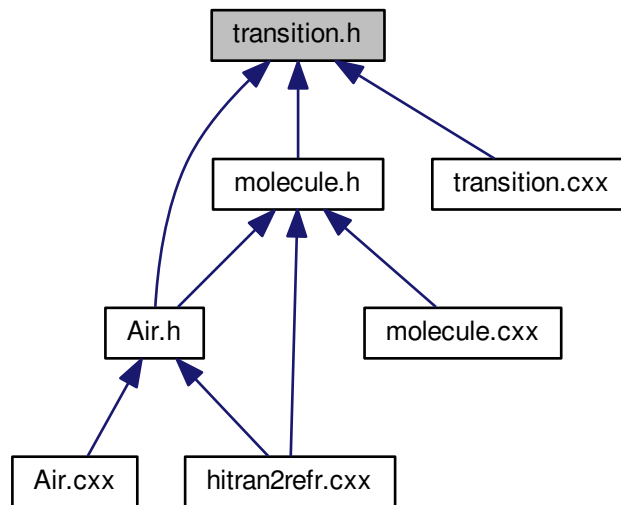
Only a limited list of molecules is handled.

7.11 transition.h File Reference

Include dependency graph for transition.h:



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



Data Structures

- class [transition](#)

Macros

- #define [USE_QOFT](#) 296
- #define [GEISALL](#) 255

Enumerations

- enum [catalog](#) {
[HITRAN](#), [SPECJPL](#), [CFAWWW](#), [JCP111](#),
[GEISA](#), [GEISA_09](#) }

7.11.1 Macro Definition Documentation

7.11.1.1 #define GEISALL 255

Maximum number of bytes per GEISA 2009 input line (Fortran total 252), which is followed by

or

, then \0 In case of providing space for input data, this is to be considered larger than HITRANLL

7.11.1.2 #define USE_QOFT 296

7.11.2 Enumeration Type Documentation

7.11.2.1 enum catalog

flag which input format is used in the ASCII files, one of <http://www.hitran.com>, <http://spec.jpl.nasa.gov/> line or <http://cfa-www.harvard.edu/firs/sao92.html>

Since

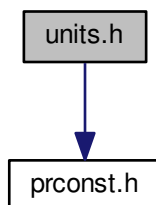
2010-06-10 GEISA-09

Enumerator:

HITRAN
SPECJPL
CFWWW
JCP111
GEISA
GEISA_09

7.12 units.h File Reference

Include dependency graph for units.h:



Macros

- #define [ARCSEC2DEG](#)(a) ((a)/3600.)
- #define [MAS2DEG](#)(m) ((m)/1.e6/3600.)
- #define [DEG2ARCSEC](#)(d) (3600.*(d))
- #define [ARCMIN2DEG](#)(a) ((a)/60.)
- #define [DEG2RAD](#)(d) (M_PI*(d)/180.)
- #define [RAD2DEG](#)(r) (180.*(r)/M_PI)
- #define [MICRON2WN](#)(m) (1.e4/(m))
- #define [WN2MICRON](#)(w) (1.e4/(w))
- #define [K2KAYSER](#)(k) ((k)/(2.*M_PI*100.))
- #define [KAYSER2K](#)(k) (2.*M_PI*100.*(k))
- #define [K2MICRON](#)(k) (2.*M_PI*1.e6/(k))
- #define [MICRON2K](#)(m) (2.*M_PI*1.e6/(m))
- #define [WN2HZ](#)(w) (100.*[VACUUMC](#)*(w))
- #define [HZ2WN](#)(h) ((h)/(100.*[VACUUMC](#)))
- #define [KELVIN2CELS](#)(k) ((k)-273.15)
- #define [CELS2KELVIN](#)(c) ((c)+273.15)

- #define RAD2ARCSEC(*r*) (DEG2ARCSEC(RAD2DEG(*r*)))
- #define ARCSEC2RAD(*a*) (DEG2RAD(ARCSEC2DEG(*a*)))
- #define ARCMIN2RAD(*a*) (DEG2RAD(ARCMIN2DEG(*a*)))
- #define D2AREA(*d*) (M_PI_4*(*d*)*(*d*))

7.12.1 Macro Definition Documentation

7.12.1.1 #define ARCMIN2DEG(*a*) ((*a*)/60.)

Convert arc-minutes to degrees

7.12.1.2 #define ARCMIN2RAD(*a*) (DEG2RAD(ARCMIN2DEG(*a*)))

convert arc-minutes to radians

7.12.1.3 #define ARCSEC2DEG(*a*) ((*a*)/3600.)

Convert arcseconds to degrees

7.12.1.4 #define ARCSEC2RAD(*a*) (DEG2RAD(ARCSEC2DEG(*a*)))

convert arcseconds to radians

7.12.1.5 #define CELS2KELVIN(*c*) ((*c*)+273.15)

convert Celsius to Kelvin

7.12.1.6 #define D2AREA(*d*) (M_PI_4*(*d*)*(*d*))

convert diameter to circle surface area

7.12.1.7 #define DEG2ARCSEC(*d*) (3600.*(d))

Convert degrees to arcseconds

7.12.1.8 #define DEG2RAD(*d*) (M_PI*(d)/180.)

Convert degrees to radians

7.12.1.9 #define HZ2WN(*h*) ((*h*)/(100.*VACUUMC))

Convert frequency [Hz] to spectroscopic wavenumber [1/cm]

7.12.1.10 #define K2KAYSER(*k*) ((*k*)/(2.*M_PI*100.))

Convert momentum measured in [1/m] to spectroscopic wavenumber [1/cm=kayser] The parameter *k* multiplied with a distance [m] would give a phase [radians], so the difference between momentum and spectroscopic wavenumber used elsewhere is the factor 2 pi and the factor 100 for conversion of 1/cm to 1/m.

7.12.1.11 #define K2MICRON(*k*) (2.*M_PI*1.e6/(*k*))

Convert momentum measured in [1/m] to wavelength [micron]

7.12.1.12 #define KAYSER2K(*k*) (2.*M_PI*100.*(k))

7.12.1.13 #define KELVIN2CELS(*k*) ((*k*)-273.15)

convert Kelvin to Celsius

7.12.1.14 `#define MAS2DEG(m) ((m)/1.e6/3600.)`

Convert milli-arcseconds to degrees

7.12.1.15 `#define MICRON2K(m) (2.*M.PI*1.e6/(m))`

Convert wavelength [micron] to momentum measured in [1/m]

7.12.1.16 `#define MICRON2WN(m) (1.e4/(m))`

Convert wavelength [microns] to spectroscopic wavenumbers [1/cm=kayser].

7.12.1.17 `#define RAD2ARCSEC(r) (DEG2ARCSEC(RAD2DEG(r)))`

convert radians to arcseconds

7.12.1.18 `#define RAD2DEG(r) (180.*(r)/M.PI)`

Convert radians to degrees

7.12.1.19 `#define WN2HZ(w) (100.*VACUUMC*(w))`

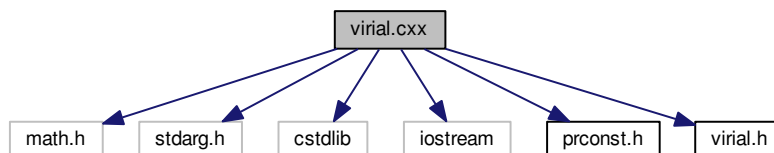
Convert spectroscopic wavenumber [1/cm] to frequency [Hz]

7.12.1.20 `#define WN2MICRON(w) (1.e4/(w))`

Convert spectroscopic wavenumber [1/cm=kayser] into wavelength [microns]

7.13 virial.cxx File Reference

Include dependency graph for virial.cxx:



Macros

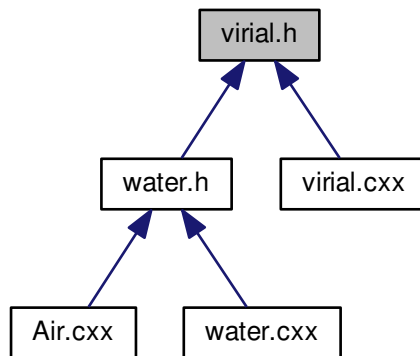
- `#define VIRIAL_C`

7.13.1 Macro Definition Documentation

7.13.1.1 `#define VIRIAL_C`

7.14 virial.h File Reference

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



Data Structures

- class [virial](#)

Enumerations

- enum [authors](#) { [WagPruss](#), [HarveyLemmon](#), [HillMcMillan](#), [None](#) }

7.14.1 Enumeration Type Documentation

7.14.1.1 enum authors

Enumerator:

WagPruss

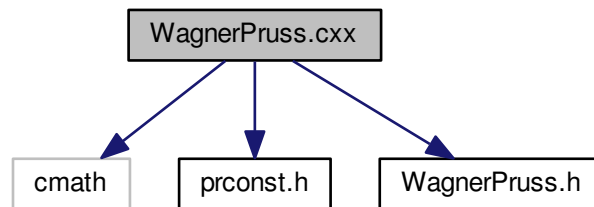
HarveyLemmon

HillMcMillan

None

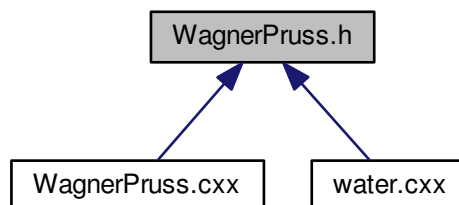
7.15 WagnerPruss.cxx File Reference

Include dependency graph for WagnerPruss.cxx:



7.16 WagnerPruss.h File Reference

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

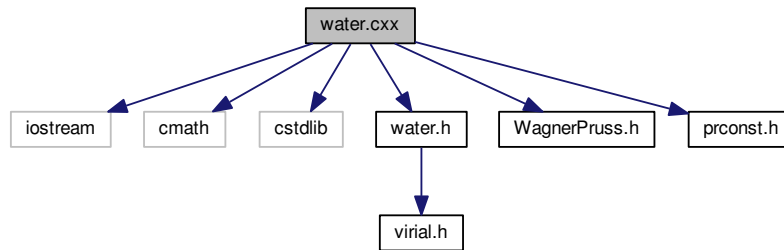


Data Structures

- class [WagnerPruss](#)
- struct [WagnerPruss::tab62aS](#)
- struct [WagnerPruss::tab62bS](#)
- struct [WagnerPruss::tab62cS](#)

7.17 water.cxx File Reference

Include dependency graph for water.cxx:



Macros

- #define [GRAINSPERGRAMM](#) 0.06479891
- #define [FEETPERMETER](#) 0.3048
- #define [FAREN2KELVIN](#)(f) (((f)-32.)*5./9.+273.15)
- #define [KELVIN2FAREN](#)(k) (9.*((k)-273.15)/5.+32.)

7.17.1 Macro Definition Documentation

7.17.1.1 #define [FAREN2KELVIN](#)(f) (((f)-32.)*5./9.+273.15)

convert Farenheit to Kelvin

7.17.1.2 #define [FEETPERMETER](#) 0.3048

conversion factor feet per meter

7.17.1.3 #define [GRAINSPERGRAMM](#) 0.06479891

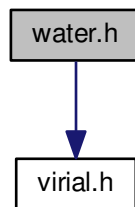
conversion factor grains/g

7.17.1.4 #define [KELVIN2FAREN](#)(k) (9.*((k)-273.15)/5.+32.)

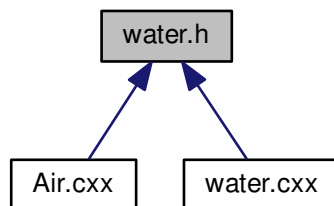
convert Kelvin to Farenheit

7.18 water.h File Reference

Include dependency graph for water.h:



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



Data Structures

- class [water](#)

Enumerations

- enum [eos](#) { [Fairchild](#), [Virial](#), [WPruss](#) }

7.18.1 Enumeration Type Documentation

7.18.1.1 enum eos

Enumerator:

Fairchild

Virial

WPruss

Index

- ~Air
 - Air, [6](#)
- ~molecule
 - molecule, [12](#)
- ~water
 - water, [27](#)
- ARCMIN2DEG
 - units.h, [63](#)
- ARCMIN2RAD
 - units.h, [63](#)
- ARCSEC2DEG
 - units.h, [63](#)
- ARCSEC2RAD
 - units.h, [63](#)
- ATOMS_OFFSET
 - Air.h, [30](#)
 - transition.cxx, [59](#)
- AVOGADRO
 - prconst.h, [35](#)
- Ai
 - WagnerPruss::tab62cS, [15](#)
- ai
 - WagnerPruss::tab62cS, [15](#)
- Air, [4](#)
 - ~Air, [6](#)
 - Air, [5](#)
 - hash2o, [6](#)
 - magn, [6](#)
 - mix, [7](#)
 - mixcnt, [7](#)
 - nam, [7](#)
 - Pa, [6](#)
 - Panom, [7](#)
 - ppmCO2, [6](#)
 - Qsum, [6](#)
 - scale, [6](#)
 - scale_p, [7](#)
 - scaledry, [7](#)
 - suscep, [7](#)
 - T, [7](#)
 - volrel, [7](#)
- Air.cxx, [28](#)
- Air.h, [29](#)
 - ATOMS_OFFSET, [30](#)
 - KELVIN2CELS, [30](#)
 - MMHG2PA, [30](#)
 - PA2MMHG, [30](#)
- Air::compon, [9](#)
 - mol_m3, [10](#)
 - molc, [10](#)
 - partp, [10](#)
 - volpercent, [10](#)
- alphi
 - WagnerPruss::tab62bS, [14](#)
- AtoB
 - tips_2003.cxx, [37](#)
 - tips_2003.h, [48](#)
- authors
 - virial.h, [65](#)
- BC
 - virial, [20](#)
- BOLTZMANNK
 - prconst.h, [35](#)
- bd_tips_2003
 - tips_2003.cxx, [37](#)
 - tips_2003.h, [48](#)
- beti
 - WagnerPruss::tab62bS, [14](#)
 - WagnerPruss::tab62cS, [16](#)
- Bi
 - WagnerPruss::tab62cS, [16](#)
- bi
 - WagnerPruss::tab62cS, [16](#)
- CFAWWW
 - transition.h, [62](#)
- CELS2KELVIN
 - hitran2refr.cxx, [30](#)
 - units.h, [63](#)
- CFAWWWDIR
 - molecule.cxx, [33](#)
- catalog
 - transition.h, [61](#)
- Ci
 - WagnerPruss::tab62cS, [16](#)
- ci
 - WagnerPruss::tab62aS, [13](#)
- D2AREA
 - units.h, [63](#)
- dDeltadelta
 - WagnerPruss, [23](#)
- DEG2ARCSEC
 - units.h, [63](#)
- DEG2RAD
 - units.h, [63](#)
- DMICRON2DOMEGA
 - transition.cxx, [59](#)
- Di
 - WagnerPruss::tab62cS, [16](#)
- di
 - WagnerPruss::tab62aS, [13](#)
 - WagnerPruss::tab62bS, [14](#)
- dipole
 - transition, [18](#)
- ECHARGE
 - prconst.h, [35](#)
- EMASS

- prconst.h, 35
- EPSILONZERO
 - prconst.h, 35
- EV2WN
 - transition.cxx, 59
- eos
 - water.h, 68
- epsi
 - WagnerPruss::tab62bS, 14
- equa_of_s
 - water, 28
- f
 - transition, 18
- FAREN2KELVIN
 - water.cxx, 67
- FEETPERMETER
 - water.cxx, 67
- Fairchild
 - water.h, 68
- GEISA
 - transition.h, 62
- GEISA_09
 - transition.h, 62
- GEISADIR
 - molecule.cxx, 33
- GEISALL
 - molecule.cxx, 33
 - transition.h, 61
- GRAINSPERGRAMM
 - water.cxx, 67
- gami
 - WagnerPruss::tab62bS, 14
- getval
 - transition.cxx, 59
- gnuplot
 - molecule, 12
- go
 - hitran2refr.tcl, 32
- HITRAN
 - transition.h, 62
- HARTREE
 - prconst.h, 35
- HBAR
 - prconst.h, 35
- HC
 - prconst.h, 35
- HITRAN2000DIR
 - molecule.cxx, 33
- HZ2WN
 - units.h, 63
- HarveyLemmon
 - virial.h, 65
- hasdry
 - hitran2refr.tcl, 32
- hash2o
 - Air, 6
- HillMcMillan
 - virial.h, 65
- hitran2refr.cxx, 30
 - CELS2KELVIN, 30
 - MICRON2THZ, 30
 - MICRON2WN, 30
 - main, 31
 - USE_MAGNETISM, 31
 - USE_QOFT, 31
 - usage, 32
 - WN2MICRON, 31
- hitran2refr.tcl, 32
 - go, 32
 - hasdry, 32
- idealgas
 - virial, 22
- isot
 - molecule, 12
 - transition, 18
- JCP111
 - transition.h, 62
- JCP111DIR
 - molecule.cxx, 33
- K2KAYSER
 - units.h, 63
- K2MICRON
 - units.h, 63
- KAYSER2K
 - units.h, 63
- KELVIN2CELS
 - Air.h, 30
 - units.h, 63
- KELVIN2FAREN
 - water.cxx, 67
- Lorentz
 - molecule, 12
 - transition, 18
- LorentzPrime
 - transition, 18
- lowsta
 - transition, 18
- M
 - WagnerPruss, 25
- MAS2DEG
 - units.h, 63
- MHZ2WN
 - transition.cxx, 59
- MICRON2K
 - units.h, 64
- MICRON2OMEGA
 - molecule.cxx, 33
 - transition.cxx, 59
- MICRON2THZ
 - hitran2refr.cxx, 30

- MICRON2WN
 - hitran2refr.cxx, [30](#)
 - transition.cxx, [59](#)
 - units.h, [64](#)
- MMHG2PA
 - Air.h, [30](#)
- magn
 - Air, [6](#)
- main
 - hitran2refr.cxx, [31](#)
- micron
 - transition, [18](#)
- mix
 - Air, [7](#)
- mixcnt
 - Air, [7](#)
- mixrat
 - water, [27](#)
- mmas
 - molecule, [12](#)
- mol_m3
 - Air::compon, [10](#)
- molc
 - Air::compon, [10](#)
- molec
 - molecule, [12](#)
 - transition, [18](#)
 - virial, [22](#)
- molecule, [11](#)
 - ~molecule, [12](#)
 - gnuplot, [12](#)
 - isot, [12](#)
 - Lorentz, [12](#)
 - mmas, [12](#)
 - molec, [12](#)
 - molecule, [12](#)
 - molmass, [12](#)
 - murange, [12](#)
 - Qsum, [12](#)
 - trans, [12](#)
 - transcnt, [13](#)
- molecule.cxx, [32](#)
 - CFAWWDIR, [33](#)
 - GEISADIR, [33](#)
 - GEISALL, [33](#)
 - HITRAN2000DIR, [33](#)
 - JCP111DIR, [33](#)
 - MICRON2OMEGA, [33](#)
 - SPECJPLDIR, [33](#)
 - WN2MICRON, [34](#)
- molecule.h, [34](#)
- molmass
 - molecule, [12](#)
- murange
 - molecule, [12](#)
- n
 - WagnerPruss, [24](#)
 - water, [27](#)
- n2p
 - virial, [20, 21](#)
- nFairchild
 - water, [28](#)
- NT
 - tips_2003.cxx, [37](#)
- nWPruss
 - water, [28](#)
- nam
 - Air, [7](#)
- ni
 - WagnerPruss::tab62aS, [13](#)
 - WagnerPruss::tab62bS, [14](#)
 - WagnerPruss::tab62cS, [16](#)
- None
 - virial.h, [65](#)
- omega0
 - transition, [19](#)
- P
 - WagnerPruss, [24](#)
- p2n
 - virial, [21](#)
- PA2MMHG
 - Air.h, [30](#)
- PLANCK
 - prconst.h, [35](#)
- pSatur
 - WagnerPruss, [25](#)
- pWPruss
 - WagnerPruss, [24](#)
- Pa
 - Air, [6](#)
- Panom
 - Air, [7](#)
- partp
 - Air::compon, [10](#)
- Phi_delta_r
 - WagnerPruss, [24](#)
- ppmCO2
 - Air, [6](#)
- prconst.h, [35](#)
 - AVOGADRO, [35](#)
 - BOLTZMANNK, [35](#)
 - ECHARGE, [35](#)
 - EMASS, [35](#)
 - EPSILONZERO, [35](#)
 - HARTREE, [35](#)
 - HBAR, [35](#)
 - HC, [35](#)
 - PLANCK, [35](#)
 - RMOLAR, [35](#)
 - SECPERDAY, [35](#)
 - VACUUMC, [35](#)
 - WATERMOLWEIGHT, [35](#)
- Qsum
 - Air, [6](#)

- molecule, [12](#)
- transition, [18](#)
- qt_c2h2
 - [tips_2003.cxx](#), [37](#)
 - [tips_2003.h](#), [49](#)
- qt_c2h4
 - [tips_2003.cxx](#), [38](#)
 - [tips_2003.h](#), [49](#)
- qt_c2h6
 - [tips_2003.cxx](#), [38](#)
 - [tips_2003.h](#), [49](#)
- qt_ch3cl
 - [tips_2003.cxx](#), [38](#)
 - [tips_2003.h](#), [49](#)
- qt_ch4
 - [tips_2003.cxx](#), [38](#)
 - [tips_2003.h](#), [50](#)
- qt_clo
 - [tips_2003.cxx](#), [39](#)
 - [tips_2003.h](#), [50](#)
- qt_clono2
 - [tips_2003.cxx](#), [39](#)
 - [tips_2003.h](#), [50](#)
- qt_co
 - [tips_2003.cxx](#), [39](#)
 - [tips_2003.h](#), [50](#)
- qt_co2
 - [tips_2003.cxx](#), [39](#)
 - [tips_2003.h](#), [51](#)
- qt_cof2
 - [tips_2003.cxx](#), [40](#)
 - [tips_2003.h](#), [51](#)
- qt_h2O
 - [tips_2003.cxx](#), [40](#)
 - [tips_2003.h](#), [51](#)
- qt_h2co
 - [tips_2003.cxx](#), [40](#)
 - [tips_2003.h](#), [51](#)
- qt_h2o2
 - [tips_2003.cxx](#), [40](#)
 - [tips_2003.h](#), [52](#)
- qt_h2s
 - [tips_2003.cxx](#), [41](#)
 - [tips_2003.h](#), [52](#)
- qt_hbr
 - [tips_2003.cxx](#), [41](#)
 - [tips_2003.h](#), [52](#)
- qt_hcl
 - [tips_2003.cxx](#), [41](#)
 - [tips_2003.h](#), [52](#)
- qt_hcn
 - [tips_2003.cxx](#), [41](#)
 - [tips_2003.h](#), [53](#)
- qt_hcooh
 - [tips_2003.cxx](#), [42](#)
 - [tips_2003.h](#), [53](#)
- qt_hf
 - [tips_2003.cxx](#), [42](#)
 - [tips_2003.h](#), [53](#)
- qt_hi
 - [tips_2003.cxx](#), [42](#)
 - [tips_2003.h](#), [53](#)
- qt_hno3
 - [tips_2003.cxx](#), [42](#)
 - [tips_2003.h](#), [54](#)
- qt_ho2
 - [tips_2003.cxx](#), [43](#)
 - [tips_2003.h](#), [54](#)
- qt_hobr
 - [tips_2003.cxx](#), [43](#)
 - [tips_2003.h](#), [54](#)
- qt_hocl
 - [tips_2003.cxx](#), [43](#)
 - [tips_2003.h](#), [54](#)
- qt_n2
 - [tips_2003.cxx](#), [43](#)
 - [tips_2003.h](#), [55](#)
- qt_n2o
 - [tips_2003.cxx](#), [44](#)
 - [tips_2003.h](#), [55](#)
- qt_nh3
 - [tips_2003.cxx](#), [44](#)
 - [tips_2003.h](#), [55](#)
- qt_no
 - [tips_2003.cxx](#), [44](#)
 - [tips_2003.h](#), [55](#)
- qt_no2
 - [tips_2003.cxx](#), [44](#)
 - [tips_2003.h](#), [56](#)
- qt_nop
 - [tips_2003.cxx](#), [45](#)
 - [tips_2003.h](#), [56](#)
- qt_o
 - [tips_2003.cxx](#), [45](#)
 - [tips_2003.h](#), [56](#)
- qt_o2
 - [tips_2003.cxx](#), [45](#)
 - [tips_2003.h](#), [56](#)
- qt_o3
 - [tips_2003.cxx](#), [45](#)
 - [tips_2003.h](#), [57](#)
- qt_ocs
 - [tips_2003.cxx](#), [46](#)
 - [tips_2003.h](#), [57](#)
- qt_oh
 - [tips_2003.cxx](#), [46](#)
 - [tips_2003.h](#), [57](#)
- qt_ph3
 - [tips_2003.cxx](#), [46](#)
 - [tips_2003.h](#), [57](#)
- qt_sf6
 - [tips_2003.cxx](#), [46](#)
 - [tips_2003.h](#), [58](#)
- qt_so2
 - [tips_2003.cxx](#), [47](#)
 - [tips_2003.h](#), [58](#)

- R
 - WagnerPruss, 25
- RAD2ARCSEC
 - units.h, 64
- RAD2DEG
 - units.h, 64
- RMOLAR
 - prconst.h, 35
- ref
 - virial, 22
- refractive index calculation, 2
- rho
 - WagnerPruss, 24
- rhoc
 - WagnerPruss, 25
- SPECJPL
 - transition.h, 62
- SECPERDAY
 - prconst.h, 35
- SPECJPLDIR
 - molecule.cxx, 33
- saturvap
 - virial, 21
- scale
 - Air, 6
- scale_p
 - Air, 7
- scaledry
 - Air, 7
- strength
 - transition, 19
- suscep
 - Air, 7
- T
 - Air, 7
 - WagnerPruss, 25
 - water, 28
- tab62a
 - WagnerPruss, 25
- tab62b
 - WagnerPruss, 25
- tab62c
 - WagnerPruss, 25
- tag2molec
 - transition.cxx, 60
- tau
 - WagnerPruss, 25
- Tc
 - WagnerPruss, 25
- ti
 - WagnerPruss::tab62aS, 13
 - WagnerPruss::tab62bS, 15
- tips_2003.cxx, 36
 - AtoB, 37
 - bd_tips_2003, 37
 - NT, 37
 - qt_c2h2, 37
 - qt_c2h4, 38
 - qt_c2h6, 38
 - qt_ch3cl, 38
 - qt_ch4, 38
 - qt_clo, 39
 - qt_clono2, 39
 - qt_co, 39
 - qt_co2, 39
 - qt_cof2, 40
 - qt_h2O, 40
 - qt_h2co, 40
 - qt_h2o2, 40
 - qt_h2s, 41
 - qt_hbr, 41
 - qt_hcl, 41
 - qt_hcn, 41
 - qt_hcooh, 42
 - qt_hf, 42
 - qt_hi, 42
 - qt_hno3, 42
 - qt_ho2, 43
 - qt_hobr, 43
 - qt_hocl, 43
 - qt_n2, 43
 - qt_n2o, 44
 - qt_nh3, 44
 - qt_no, 44
 - qt_no2, 44
 - qt_nop, 45
 - qt_o, 45
 - qt_o2, 45
 - qt_o3, 45
 - qt_ocs, 46
 - qt_oh, 46
 - qt_ph3, 46
 - qt_sf6, 46
 - qt_so2, 47
- tips_2003.h, 47
 - AtoB, 48
 - bd_tips_2003, 48
 - qt_c2h2, 49
 - qt_c2h4, 49
 - qt_c2h6, 49
 - qt_ch3cl, 49
 - qt_ch4, 50
 - qt_clo, 50
 - qt_clono2, 50
 - qt_co, 50
 - qt_co2, 51
 - qt_cof2, 51
 - qt_h2O, 51
 - qt_h2co, 51
 - qt_h2o2, 52
 - qt_h2s, 52
 - qt_hbr, 52
 - qt_hcl, 52
 - qt_hcn, 53
 - qt_hcooh, 53

- qt_hf, [53](#)
- qt_hi, [53](#)
- qt_hno3, [54](#)
- qt_ho2, [54](#)
- qt_hobr, [54](#)
- qt_hocl, [54](#)
- qt_n2, [55](#)
- qt_n2o, [55](#)
- qt_nh3, [55](#)
- qt_no, [55](#)
- qt_no2, [56](#)
- qt_nop, [56](#)
- qt_o, [56](#)
- qt_o2, [56](#)
- qt_o3, [57](#)
- qt_ocs, [57](#)
- qt_oh, [57](#)
- qt_ph3, [57](#)
- qt_sf6, [58](#)
- qt_so2, [58](#)
- trans
 - molecule, [12](#)
- transcnt
 - molecule, [13](#)
- transition, [16](#)
 - dipole, [18](#)
 - f, [18](#)
 - isot, [18](#)
 - Lorentz, [18](#)
 - LorentzPrime, [18](#)
 - lowsta, [18](#)
 - micron, [18](#)
 - molec, [18](#)
 - omega0, [19](#)
 - Qsum, [18](#)
 - strength, [19](#)
 - transition, [17](#)
 - waven, [19](#)
 - width, [19](#)
 - width_s, [19](#)
- transition.h
 - CFAWWW, [62](#)
 - GEISA, [62](#)
 - GEISA_09, [62](#)
 - HITRAN, [62](#)
 - JCP111, [62](#)
 - SPECJPL, [62](#)
- transition.cxx, [58](#)
 - ATOMS_OFFSET, [59](#)
 - DMICRON2DOMEGA, [59](#)
 - EV2WN, [59](#)
 - getval, [59](#)
 - MHZ2WN, [59](#)
 - MICRON2OMEGA, [59](#)
 - MICRON2WN, [59](#)
 - tag2molec, [60](#)
 - WN2HARTREE, [59](#)
 - WN2HZ, [59](#)
 - WN2MICRON, [59](#)
 - WN2OMEGA, [59](#)
- transition.h, [60](#)
 - catalog, [61](#)
 - GEISALL, [61](#)
 - USE_QOFT, [61](#)
- USE_MAGNETISM
 - hitran2refr.cxx, [31](#)
- USE_QOFT
 - hitran2refr.cxx, [31](#)
 - transition.h, [61](#)
- units.h, [62](#)
 - ARCMIN2DEG, [63](#)
 - ARCMIN2RAD, [63](#)
 - ARCSEC2DEG, [63](#)
 - ARCSEC2RAD, [63](#)
 - CELS2KELVIN, [63](#)
 - D2AREA, [63](#)
 - DEG2ARCSEC, [63](#)
 - DEG2RAD, [63](#)
 - HZ2WN, [63](#)
 - K2KAYSER, [63](#)
 - K2MICRON, [63](#)
 - KAYSER2K, [63](#)
 - KELVIN2CELS, [63](#)
 - MAS2DEG, [63](#)
 - MICRON2K, [64](#)
 - MICRON2WN, [64](#)
 - RAD2ARCSEC, [64](#)
 - RAD2DEG, [64](#)
 - WN2HZ, [64](#)
 - WN2MICRON, [64](#)
- usage
 - hitran2refr.cxx, [32](#)
- VACUUMC
 - prconst.h, [35](#)
- VIRIAL_C
 - virial.cxx, [64](#)
- vir
 - water, [28](#)
- Virial
 - water.h, [68](#)
- virial, [19](#)
 - BC, [20](#)
 - idealgas, [22](#)
 - molec, [22](#)
 - n2p, [20](#), [21](#)
 - p2n, [21](#)
 - ref, [22](#)
 - saturvap, [21](#)
 - virial, [20](#)
- virial.h
 - HarveyLemmon, [65](#)
 - HillMcMillan, [65](#)
 - None, [65](#)
 - WagPruss, [65](#)
- virial.cxx, [64](#)

- VIRIAL_C, 64
- virial.h, 65
 - authors, 65
- volpercent
 - Air::compon, 10
- volrel
 - Air, 7
- WPruss
 - water.h, 68
- WATERMOLWEIGHT
 - prconst.h, 35
- WN2HARTREE
 - transition.cxx, 59
- WN2HZ
 - transition.cxx, 59
 - units.h, 64
- WN2MICRON
 - hitran2refr.cxx, 31
 - molecule.cxx, 34
 - transition.cxx, 59
 - units.h, 64
- WN2OMEGA
 - transition.cxx, 59
- WagPruss
 - virial.h, 65
- WagnerPruss, 22
 - dDeltadelta, 23
 - M, 25
 - n, 24
 - P, 24
 - pSatur, 25
 - pWPruss, 24
 - Phi_delta_r, 24
 - R, 25
 - rho, 24
 - rhoc, 25
 - T, 25
 - tab62a, 25
 - tab62b, 25
 - tab62c, 25
 - tau, 25
 - Tc, 25
 - WagnerPruss, 23
 - WagnerPruss, 23
- WagnerPruss.cxx, 66
- WagnerPruss.h, 66
- WagnerPruss::tab62aS, 13
 - ci, 13
 - di, 13
 - ni, 13
 - ti, 13
- WagnerPruss::tab62bS, 14
 - alphi, 14
 - beti, 14
 - di, 14
 - epsi, 14
 - gami, 14
 - ni, 14
 - ti, 15
- WagnerPruss::tab62cS, 15
 - Ai, 15
 - ai, 15
 - beti, 16
 - Bi, 16
 - bi, 16
 - Ci, 16
 - Di, 16
 - ni, 16
- water, 26
 - ~water, 27
 - equa_of_s, 28
 - mixrat, 27
 - n, 27
 - nFairchild, 28
 - nWPruss, 28
 - T, 28
 - vir, 28
 - water, 27
- water.h
 - Fairchild, 68
 - Virial, 68
 - WPruss, 68
- water.cxx, 67
 - FAREN2KELVIN, 67
 - FEETPERMETER, 67
 - GRAINSPERGRAMM, 67
 - KELVIN2FAREN, 67
- water.h, 68
 - eos, 68
- waven
 - transition, 19
- width
 - transition, 19
- width_s
 - transition, 19