USING REFPROP WITH OTHER PROGRAMS

The subroutines are contained in a number of files, which are placed in the
REFPROP\FORTRAN directory. The following files should be compiled and linked with
your own main program:

```
core_anc.for  core_bwr.for   core_cpp.for   core_de.for    core_ecs.for
core_feq.for  core_mlt.for   core_ph0.for   core_pr.for    core_stn.for
core_qui.for
flash2.for    flsh_sub.for   idealgas.for   mix_aga8.for   mix_hmx.for
prop_sub.for  realgas.for    sat_sub.for    setup.for      setup2.for
trns_ecs.for  trns_tcx.for   trns_vis.for   trnsp.for      utility.for
```

These 26 files were compiled with the Lahey compiler using a command similar
to this:

```
   lf95 core_anc.for -c -nco -nom -dll -win -ml msvb
```

and linked with the following:

```
   lf95 pass_ftn.obj setup.obj core_anc.obj core_bwr.obj core_cpp.obj core_de.obj core_ecs.obj
   core_feq.obj core_mlt.obj core_ph0.obj core_qui.obj core_stn.obj core_pr.obj flash2.obj
   flsh_sub.obj idealgas.obj mix_aga8.obj mix_hmx.obj prop_sub.obj realgas.obj sat_sub.obj
   setup2.obj trns_ecs.obj trns_tcx.obj trns_vis.obj trnsp.obj utility.obj   -nco -nom -dll -win -ml
   msvb -out refprop.dll
```

There is one additional file that comes with REFPROP. This file, called
pass_ftn.for, is used only when creating the DLL.

Several example files showing how to call the routines are located in your
REFPROP\Examples directory. EXAMPLE.FOR shows the common routines that you
might need. EX-MIX.FOR shows how to load a mixture .MIX file containing
information for specified blends of refrigerants (and air). The file
EX-PPF.FOR shows how to use the pseudo-pure fluid files.

In version 8.0, any fluids that are used must have a ".fld" extension, e.g.,
R134a.fld, or a ".ppf" extension, e.g., air.ppf. In 8.1 and later versions, the
.fld extension is no longer needed and assumed if missing. The file HMX.BNC
(containing mixture parameters) must also be present. It is suggested (but not
required) that these be put into a subdirectory called "fluids" under the
directory containing the subroutines and application program.

The subroutine SETUP must be called to initialize the pure fluid or mixture
components. The call to SETUP will allow the choice of one of three standard
reference states for entropy and enthalpy and will automatically load the
NIST-recommended models for the components as well as mixing rules. The
routine SETMOD allows the specification of other models. To define another
reference state, or to apply one of the standard states to a mixture of a
specified composition, the subroutine SETREF may be used. These additional
routines should be called only if the fluids and/or models (or reference state)
are changed. The sequence is:

```
call SETMOD    (optional)
call SETUP     (REQUIRED)         [or call SETMIX]
call SETKTV    (optional)
call SETREF    (optional)
```

Subroutine SETUP0 calls subroutine SETUP using the same techniques as those
used by the graphical interface and the Excel spreadsheet. Subroutine
SETMIX reads the *.MIX files and makes the appropriate call to subroutine
SETUP. Subroutine SETPATH sets the path where the *.fld files can be
found. Subroutine PUREFLD allows the user to calculate the properties of a
pure fluid when a mixture has been loaded and the fluid is one of the
constituents in the mixture.

Units. The subroutines use the following units for all inputs and outputs:

```
temperature                    K
pressure, fugacity             kPa
density                        mol/L
composition                    mole fraction
quality                        mole basis (moles vapor/total moles)
enthalpy, internal energy      J/mol
Gibbs, Helmholtz free energy   J/mol
entropy, heat capacity         J/(mol.K)
```

speed of sound	m/s
Joule-Thompson coefficient	K/kPa
d(p)/d(rho)	kPa.L/mol
d2(p)/d(rho)2	kPa.(L/mol)^2
viscosity	microPa.s (10^-6 Pa.s)
thermal conductivity	W/(m.K)
dipole moment	debye
surface tension	N/m

Note: The only exceptions to the above are the conversion utilities
XMASS and XMOLE. The graphical interface allows a wide variety
of units, but not the subroutines.

Naming conventions. The variable type of subroutine arguments can
generally be inferred from the first letter of the variable name:

a-g and o-z:	double precision
h:	double precision (i.e. enthalpy) or character
i-k and m,n:	integer
l:	logical (within subroutines only, no logicals in arguments), also used as integer for exponent of FEQ model

The property subroutines are written in ANSI standard Fortran 77 and are
compatible with Fortran 90. We are striving to make these as standard and
portable as possible, but every compiler has its own sensitive points.
Please report any compiler or linker errors or warnings.

Potential pitfalls:

The fluid data files are read using logical unit 12. Use of this unit
in your program may crash the program and should be avoided.

The following is a description of the high-level routines that would be
used in stand-alone applications. These routines will give you access
to all features in a model-independent fashion. (There are corresponding
low-level routines for some of these that call specific models. Please
do not incorporate the low-level routines into your applications--if
you do, you may find that future versions may not work the same.)

Warning and Error Messages:

The FORTRAN routines return warning and error codes and messages when the input
conditions are out of range or the calculations fail. The error codes are
integers returned in the subroutine argument ierr. When a calculation is
successful, the value of ierr is 0. Negative values of ierr signify a
warning-that is, a condition where calculations were completed, but the results
may be suspect. The most common warning would be for a temperature or pressure
slightly above the limits of the property model.

Errors imply conditions for which the calculations failed (such as an iteration
not converging) or for which calculations are not possible (such as a negative
pressure or a mixture composition not summing to one). Many property models
give nonsensical results when extrapolated to lower temperatures and/or higher
densities; thus errors are also issued when the temperature is below the limit
of the model or the density is above the limit. The codes for errors are
positive numbers.

More detailed messages (including, in some cases, a report of the input
condition(s) that resulted in the warning or error) are available via the output
variable herr, which may be printed or displayed using the subroutine ERRMSG.
In the graphical interface, errors are displayed in an Alert box. If the
calculation of a table results in a warning, a message is displayed at the
completion of the calculations, and the individual messages can be displayed by
clicking the mouse on the number of the affected row.

INITIALIZATION SUBROUTINES

```
      subroutine SETUP (nc,hfiles,hfmix,hrf,ierr,herr)
```

```
c
c  define models and initialize arrays
c
c  A call to this routine is required.
c
c  inputs:
c       nc--number of components (1 for pure fluid) [integer]
c           if called with nc=-1, the version number*100 will be returned in ierr
c   hfiles--array of file names specifying fluid/mixture components
c           [character*255 variable] for each of the nc components;
c           e.g., :fluids:r134a.fld (Mac) or fluids\r134a.fld (DOS) or
c           [full_path]/fluids/r134a.fld (UNIX)
c    hfmix--mixture coefficients [character*255]
c           file name containing coefficients for mixture model,
c           if applicable
c           e.g.,  fluids\hmx.bnc
c      hrf--reference state for thermodynamic calculations [character*3]
c           'DEF':  default reference state as specified in fluid file
c                   is applied to each pure component
c           'NBP':  h,s = 0 at pure component normal boiling point(s)
c           'ASH':  h,s = 0 for sat liquid at -40 C (ASHRAE convention)
c           'IIR':  h = 200, s = 1.0 for sat liq at 0 C (IIR convention)
c           other choices are possible, but these require a separate
c           call to SETREF
c  outputs:
c     ierr--error flag:  0 = successful
c                      101 = error in opening file
c                      102 = error in file or premature end of file
c                     -103 = unknown model encountered in file
c                      104 = error in setup of model
c                      105 = specified model not found
c                      111 = error in opening mixture file
c                      112 = mixture file of wrong type
c                      114 = nc<>nc from setmod
c                     -117 = binary pair not found, all parameters will be estimated
c                      117 = No mixture data are available, mixture is outside the
c                            range of the model and calculations will not be made
c     herr--error string (character*255 variable if ierr<>0)
c     [fluid parameters, etc. returned via various common blocks]




      subroutine SETUP0 (i,hfld,hfm,hrf,ierr,herr)
c
c     call the SETUP routine with the same inputs as SETUP except for
c     the hfld variable.  This subroutine is generally used in calls
c     to the REFPROP DLL since the call cannot handle a string array.
c     The hfld variable is a string of length 10000.  For a pure fluid,
c     it simply contains the name of the fluid file (with a path if needed).
c     For a mixture, it contains the names of the constituents in the
c     mixture separated by a |.  For the air mixture, this would be
c     something like (depends on the need for paths):
c     hfld='fluids\nitrogen.fld|fluids\argon.fld|fluids\oxygen.fld|'




      subroutine SETFLD (icomp,hfile,ierr,herr)
c
c  open a fluid file and read model coefficients (or get from block data)
c
c  inputs:
c    icomp--pointer specifying component number
c           zero and negative values are used for ECS reference fluid(s)
c    hfile--array of file names specifying fluid/mixture components
c           [character*255 variable] for each of the components;
c           --or--
c           when hf(i) is of the form:
c             BDATA:nn-nn-nn
c           use coefficients stored in block data for fluid with CAS
c           number specified by nn-nn-nn;
```

```
c           e.g. to use stored formulation for R134a,
c           hf(i) = 'BDATA:811-97-2'
c  outputs:
c     ierr--error flag:  0 = successful
c                      101 = error in opening file
c                      102 = error in file or premature end of file
c                     -103 = unknown model encountered in file
c                      104 = error in setup of model
c                      105 = specified model not found
c                      106 = cp0 equation not found
c     herr--error string (character*255 variable if ierr<>0)
c     [fluid parameters, etc. returned via various common blocks]




      subroutine SETMOD (nc,htype,hmix,hcomp,ierr,herr)
c
c  set model(s) other than the NIST-recommended ('NBS') ones
c
c  This subroutine must be called before SETUP; it need not be called
c  at all if the default (NIST-recommended) models are desired.
c
c  inputs:
c       nc--number of components (1 for pure fluid) [integer]
c    htype--flag indicating which models are to be set [character*3]
c           'EOS':  equation of state for thermodynamic properties
c           'ETA':  viscosity
c           'TCX':  thermal conductivity
c           'STN':  surface tension
c           'NBS':  reset all of the above model types and all
c                   subsidiary component models to 'NBS';
c                   values of hmix and hcomp are ignored
c     hmix--mixture model to use for the property specified in htype [character*3];
c           ignored if number of components = 1
c           some allowable choices for hmix:
c             'NBS':  use NIST recommendation for specified fluid/mixture
c             'HMX':  mixture Helmholtz model for thermodynamic properties
c             'ECS':  extended corresponding states for viscosity or therm. cond.
c             'STX':  surface tension mixture model
c    hcomp--component model(s) to use for property specified in htype [array (1..nc) of character*3]
c             'NBS':  NIST recommendation for specified fluid/mixture
c           some allowable choices for an equation of state:
c             'FEQ':  Helmholtz free energy model
c             'BWR':  pure fluid modified Benedict-Webb-Rubin (MBWR)
c             'ECS':  pure fluid thermo extended corresponding states
c           some allowable choices for viscosity:
c             'ECS':  extended corresponding states (all fluids)
c             'VS1':  the 'composite' model for R134a, R152a, NH3, etc.
c             'VS2':  Younglove-Ely model for hydrocarbons
c             'VS4':  Generalized friction theory of Quinones-Cisneros and Deiters
c             'VS5':  Chung et al. (1988) predictive model
c           some allowable choices for thermal conductivity:
c             'ECS':  extended corresponding states (all fluids)
c             'TC1':  the 'composite' model for R134a, R152a, etc.
c             'TC2':  Younglove-Ely model for hydrocarbons
c             'TC5':  Chung et al. (1988) predictive model
c           some allowable choices for surface tension:
c             'ST1':  surface tension as f(tau); tau = 1 - T/Tc
c
c  outputs:
c     ierr--error flag:  0 = successful
c                      113 = nc outside of bounds
c     herr--error string (character*255 variable if ierr<>0)
c     [fluid parameters, etc. returned via various common blocks]




      subroutine GERG04 (nc,iflag,ierr,herr)
c
c  set the pure model(s) to those used by the GERG 2004 formulation.
```

```
c
c  This subroutine must be called before SETUP; it need not be called
c  at all if the default (NIST-recommended) models are desired.
c  To turn off the GERG settings, call this routine again with iflag=0,
c  and then call the SETUP routine to reset the parameters of the equations
c  of state.
c
c  inputs:
c       nc--number of components (1 for pure fluid)
c    iflag--set to 1 to load the GERG 2004 equations, set to 0 for defaults
c  outputs:
c     ierr--error flag:  0 = successful
c     herr--error string returned from SETMOD




      subroutine SETREF (hrf,ixflag,x0,h0,s0,t0,p0,ierr,herr)
c
c  set reference state enthalpy and entropy
c
c  This subroutine must be called after SETUP; it need not be called at
c  all if the reference state specified in the call to SETUP is to be
c  used.
c
c  inputs:
c      hrf--reference state for thermodynamic calculations [character*3]
c           'NBP':  h,s = 0 at normal boiling point(s)
c           'ASH':  h,s = 0 for sat liquid at -40 C (ASHRAE convention)
c           'IIR':  h = 200, s = 1.0 for sat liq at 0 C (IIR convention)
c           'DEF':  default reference state as specified in fluid file
c                   is applied to each component (ixflag = 1 is used)
c           'OTH':  other, as specified by h0, s0, t0, p0 (real gas state)
c           'OT0':  other, as specified by h0, s0, t0, p0 (ideal gas state)
c           '???':  change hrf to the current reference state and exit.
c   ixflag--composition flag:  1 = ref state applied to pure components
c                              2 = ref state applied to mixture x0
c  following input has meaning only if ixflag = 2
c       x0--composition for which h0, s0 apply; array(1:nc) [mol frac]
c           this is useful for mixtures of a predefined composition,
c           e.g. refrigerant blends such as R410A
c  following inputs have meaning only if hrf = 'OTH'
c       h0--reference state enthalpy at t0,p0 {x0} [J/mol]
c       s0--reference state entropy at t0,p0 {x0} [J/mol-K]
c       t0--reference state temperature [K]
c           t0 = -1 indicates saturated liquid at normal boiling point
c                   (bubble point for a mixture)
c       p0--reference state pressure [kPa]
c           p0 = -1 indicates saturated liquid at t0 {and x0}
c           p0 = -2 indicates saturated vapor at t0 {and x0}
c  outputs:
c     ierr--error flag:  0 = successful
c                       22 = Tmin > Tref for IIR reference state
c                       23 = Tcrit < Tref for IIR reference state
c                       24 = Tmin > Tref for ASHRAE reference state
c                       25 = Tcrit < Tref for ASHRAE reference state
c                       26 = Tmin > Tnbp for NBP reference state
c                       27 = Tref, Pref for OTH ref state outside limits
c                      -28 = can't apply 'DEF' to mixture;
c                            will apply to pure components
c                      -29 = unknown reference state specified;
c                            will use 'DEF'
c     herr--error string (character*255 variable if ierr<>0)
c     [fluid parameters, etc. returned via various common blocks]




      subroutine SETMIX (hmxnme,hfmix,hrf,ncc,hfiles,x,ierr,herr)
c
c  open a mixture file (e.g., R410A.mix) and read constituents and
c  mole fractions
```

```
c
c  inputs:
c    hmxnme--mixture file name to be read in [character*255]
c    hfmix--mixture coefficients [character*255]
c           file name containing coefficients for mixture model
c    hrf--reference state for thermodynamic calculations [character*3]
c         (see info in subroutine setup for specifics)
c  outputs:
c     ncc--number of fluids in mixture
c     hfiles--array of file names specifying mixture components
c             that were used to call setup. [character*255 variable]
c     x--array of mole fractions for the specified mixture
c     ierr--error flag:  0 = successful
c                      101 = error in opening file
c                     -102 = mixture file contains mixing parameters
c                     -103 = composition not equal to one
c     herr--error string (character*255 variable if ierr<>0)




      subroutine SETPATH (hpth)
c
c  set the path where the fluid files are located
c
c  inputs:
c     hpth--location of the fluid files [character*255 variable]
c           The path does not need to contain the ending "\" and it can
c           point directly to the location where the DLL is stored if a
c           fluids subdirectory (with the corresponding fluid files) is
c           located there.
c           example:  hpth='C:\Program Files\Refprop'




      subroutine PUREFLD (icomp)
c
c  Change the standard mixture setup so that the properties of one fluid
c  can be calculated as if SETUP had been called for a pure fluid.
c  Calling this routine will disable all mixture calculations.
c  To reset the mixture setup, call this routine with icomp=0.
c
c  inputs:
c   icomp--fluid number in a mixture to use as a pure fluid




      subroutine SETNC (ncomp)
c
c  Allow the user to modify the value of nc (the number of components in
c  a mixture) so that a subset of the full setup can be used.
c  For example, a 5 component mixture could be set up, but nc could
c  be set to 3 to calculate properties for just the first three components.
c  This also allows a mixture to be setup with one or more pure fluids
c  loaded after the components in the mixture.  For example, R407C could
c  be loaded by calling setup with R32.fld, R125.fld, R134a.fld, and R407C.ppf.
c  The number of components would be set to 3, and PUREFLD would be called
c  to access the properties from component 4, the pseudo-pure fluid equation.
c
c  inputs:
c   ncomp--number of components in the mixture




      subroutine PREOS (i)
c
c  turn on or off the use of the PR cubic equation
c
c  inputs:
```

```
c        i--flag specifying use of PR:
c           0 - Use full equation of state (Peng-Robinson off)
c           1 - Use full equation of state with Peng-Robinson for sat. conditions
c               (not currently working)
c           2 - Use Peng-Robinson equation for all calculations
c           if i=-1, then i is returned with current usage of PR:  0, 1, or 2.
c
```

```
      subroutine SETAGA (ierr,herr)
c
c  set up working arrays for use with AGA8 equation of state
c
c  input:
c  outputs:
c     ierr--error flag:  0 = successful
c                        1 = error (e.g. fluid not found)
c     herr--error string (character*255 variable if ierr<>0)
c     [fluid parameters, etc. returned via various common blocks]
```

```
      subroutine UNSETAGA
c
c  Load original values into arrays changed in the call to SETAGA.  This
c  routine resets the values back to those loaded when SETUP was called.
```

```
      subroutine GETMOD (icomp,htype,hcode,hcite)
c
c  retrieve citation information for the property models used
c
c  inputs:
c    icomp--pointer specifying component number
c           zero and negative values are used for ECS reference fluid(s)
c    htype--flag indicating which model is to be retrieved [character*3]
c           'EOS':  equation of state for thermodynamic properties
c           'CP0':  ideal part of EOS (e.g. ideal-gas heat capacity)
c           'ETA':  viscosity
c           'VSK':  viscosity critical enhancement
c           'TCX':  thermal conductivity
c           'TKK':  thermal conductivity critical enhancement
c           'STN':  surface tension
c           'DE ':  dielectric constant
c           'MLT':  melting line (freezing line, actually)
c           'SBL':  sublimation line
c           'PS ':  vapor pressure equation
c           'DL ':  saturated liquid density equation
c           'DV ':  saturated vapor density equation
c  outputs:
c    hcode--component model used for property specified in htype
c
c           some possibilities for thermodynamic properties:
c           'FEQ':  Helmholtz free energy model
c           'BWR':  pure fluid modified Benedict-Webb-Rubin (MBWR)
c           'ECS':  pure fluid thermo extended corresponding states
c
c           some possibilities for viscosity:
c           'ECS':  extended corresponding states (all fluids)
c           'VS1':  the 'composite' model for R134a, R152a, NH3, etc.
c           'VS2':  Younglove-Ely model for hydrocarbons
c           'VS4':  generalized friction theory of Quinones-Cisneros and Dieters
c           'VS5':  Chung et al model
c
c           some possibilities for thermal conductivity:
c           'ECS':  extended corresponding states (all fluids)
c           'TC1':  the 'composite' model for R134a, R152a, etc.
```

```
c           'TC2':  Younglove-Ely model for hydrocarbons
c           'TC5':  predictive model of Chung et al. (1988)
c
c           some possibilities for surface tension:
c           'ST1':  surface tension as f(tau); tau = 1 - T/Tc
c
c    hcite--component model used for property specified in htype;
c           the first 3 characters repeat the model designation of hcode
c           and the remaining are the citation for the source




      subroutine SETKTV (icomp,jcomp,hmodij,fij,hfmix,ierr,herr)
c
c  set mixture model and/or parameters
c
c  This subroutine must be called after SETUP, but before any call to
c  SETREF; it need not be called at all if the default mixture
c  parameters (those read in by SETUP) are to be used.
c
c  inputs:
c    icomp--component i
c    jcomp--component j
c   hmodij--mixing rule for the binary pair i,j [character*3]
c           e.g. 'LJ1' (Lemmon-Jacobsen model)
c                'LM1' (modified Lemmon-Jacobsen model) or
c                'LIN' (linear mixing rules)
c           'RST' indicates reset all pairs to values from
c                 original call to SETUP (i.e. those read from file)
c                 [all other inputs are ignored]
c      fij--binary mixture parameters [array of dimension nmxpar;
c           currently nmxpar is set to 6]
c           the parameters will vary depending on hmodij;
c           for example, for the Lemmon-Jacobsen model (LJ1):
c             fij(1) = zeta
c             fij(2) = xi
c             fij(3) = Fpq
c             fij(4) = beta
c             fij(5) = gamma
c             fij(6) = 'not used'
c    hfmix--file name [character*255] containing generalized parameters
c           for the binary mixture model; this will usually be the same
c           as the corresponding input to SETUP (e.g.,':fluids:hmx.bnc')
c  outputs:
c     ierr--error flag:  0 = successful
c                      111 = error in opening mixture file
c                      112 = mixture file of wrong type
c                     -113 = illegal i,j specification
c                            (i = j or i > nc or j > nc)
c                     -114 = possibility of mismatch of interaction parameters
c     herr--error string (character*255 variable if ierr<>0)
c     [mixture parameters returned via various common blocks]




      subroutine GETKTV (icomp,jcomp,hmodij,fij,hfmix,hfij,hbinp,hmxrul)
c
c  retrieve mixture model and parameter info for a specified binary
c
c  This subroutine should not be called until after a call to SETUP.
c
c  inputs:
c    icomp--component i
c    jcomp--component j
c  outputs:
c   hmodij--mixing rule for the binary pair i,j (e.g. LJ1 or LIN)
c           [character*3]
c      fij--binary mixture parameters [array of dimension nmxpar;
c           currently nmxpar is set to 6]; the parameters will vary
c           depending on hmodij;
```

```
c    hfmix--file name [character*255] containing parameters for the
c           binary mixture model
c     hfij--description of the binary mixture parameters [character*8
c           array of dimension nmxpar]
c           for example, for the Lemmon-Jacobsen model (LJ1):
c             fij(1) = zeta
c             fij(2) = xi
c             fij(3) = Fpq
c             fij(4) = beta
c             fij(5) = gamma
c             fij(6) = 'not used'
c    hbinp--documentation for the binary parameters [character*255]
c           terminated with ASCII null character
c   hmxrul--description of the mixing rule [character*255]




      subroutine GETFIJ (hmodij,fij,hfij,hmxrul)
c
c  retrieve parameter info for a specified mixing rule
c
c  This subroutine should not be called until after a call to SETUP.
c
c  inputs:
c   hmodij--mixing rule for the binary pair i,j (e.g. LJ1 or LIN)
c          [character*3]
c  outputs:
c      fij--binary mixture parameters [array of dimension nmxpar;
c           currently nmxpar is set to 6]; the parameters will vary
c           depending on hmodij;
c     hfij--description of the binary mixture parameters [character*8
c           array of dimension nmxpar]
c   hmxrul--description of the mixing rule [character*255]
```

SATURATION-STATE SUBROUTINES

```
      subroutine SATT (t,x,kph,p,rhol,rhov,xliq,xvap,ierr,herr)
c
c  iterate for saturated liquid and vapor states given temperature
c  and the composition of one phase
c
c  inputs:
c        t--temperature [K]
c                       if t is negative, then use other variables as
c                       initial guesses at abs(t)
c        x--composition [array of mol frac] (phase specified by kph)
c      kph--phase flag: 1 = input x is liquid composition (bubble point)
c                       2 = input x is vapor composition (dew point)
c                       3 = input x is liquid composition (freezing point)
c                       4 = input x is vapor composition (sublimation point)
c  outputs:
c        p--pressure [kPa]
c     rhol--molar density [mol/L] of saturated liquid
c     rhov--molar density [mol/L] of saturated vapor
c           For a pseudo pure fluid, the density of the equilibrium phase
c           is not returned.  Call SATT twice, once with kph=1 to get
c           pliq and rhol, and once with kph=2 to get pvap and rhov.
c     xliq--liquid phase composition [array of mol frac]
c     xvap--vapor phase composition [array of mol frac]
c     ierr--error flag:   0 = successful
c                         1 = T < Tmin
c                         8 = x out of range
c                         9 = T and x out of range
c                       120 = CRITP did not converge
c                       121 = T > Tcrit
c                       122 = TPRHO-liquid did not converge (pure fluid)
```

```
c                      123 = TPRHO-vapor did not converge (pure fluid)
c                      124 = pure fluid iteration did not converge
c           following 3 error codes are advisory--iteration will either
c           converge on later guess or error out (ierr = 128)
c                     -125 = TPRHO did not converge for parent ph (mix)
c                     -126 = TPRHO did not converge for incipient (mix)
c                     -127 = composition iteration did not converge
c                      128 = mixture iteration did not converge
c     herr--error string (character*255 variable if ierr<>0)




      subroutine SATP (p,x,kph,t,rhol,rhov,xliq,xvap,ierr,herr)
c
c  iterate for saturated liquid and vapor states given pressure
c  and the composition of one phase
c
c  inputs:
c        p--pressure [kPa]
c        x--composition [array of mol frac] (phase specified by kph)
c      kph--phase flag:  1 = input x is liquid composition
c                        2 = input x is vapor composition
c                        3 = input x is liquid composition (freezing point)
c                        4 = input x is vapor composition (sublimation point)
c
c  outputs:
c        t--temperature [K]
c     rhol--molar density [mol/L] of saturated liquid
c     rhov--molar density [mol/L] of saturated vapor
c           For a pseudo pure fluid, the density of the equilibrium phase
c           is not returned.  Call SATP twice, once with kph=1 to get
c           tliq and rhol, and once with kph=2 to get tvap and rhov.
c     xliq--liquid phase composition [array of mol frac]
c     xvap--vapor phase composition [array of mol frac]
c     ierr--error flag:  0 = successful
c                        2 = P < Ptp
c                        4 = P < 0
c                        8 = x out of range
c                       12 = P and x out of range
c                      140 = CRITP did not converge
c                      141 = P > Pcrit
c                      142 = TPRHO-liquid did not converge (pure fluid)
c                      143 = TPRHO-vapor did not converge (pure fluid)
c                      144 = pure fluid iteration did not converge
c           following 3 error codes are advisory--iteration will either
c           converge on later guess or error out (ierr = 148)
c                     -144 = Raoult's law (mixture initial guess) did
c                            not converge
c                     -145 = TPRHO did not converge for parent ph (mix)
c                     -146 = TPRHO did not converge for incipient (mix)
c                     -147 = composition iteration did not converge
c                      148 = mixture iteration did not converge
c     herr--error string if ierr<>0 (character*255)




      subroutine SATD (rho,x,kph,kr,t,p,rhol,rhov,xliq,xvap,ierr,herr)
c
c  iterate for temperature and pressure given a density along the
c  saturation boundary and the composition
c
c  inputs:
c      rho--molar density [mol/L]
c        x--composition [array of mol frac]
c      kph--flag specifying desired root for multi-valued inputs
c           has meaning only for water at temperatures close to its triple point
c          -1 = return middle root (between 0 and 4 C)
c           1 = return highest temperature root (above 4 C)
c           3 = return lowest temperature root (along freezing line)
c  outputs:
```

```
c        t--temperature [K]
c        p--pressure [kPa]
c     rhol--molar density [mol/L] of saturated liquid
c     rhov--molar density [mol/L] of saturated vapor
c     xliq--liquid phase composition [array of mol frac]
c     xvap--vapor phase composition [array of mol frac]
c       kr--phase flag: 1 = input state is liquid
c                       2 = input state is vapor in equilibrium with liq
c                       3 = input state is liquid in equilibrium with solid
c                       4 = input state is vapor in equilibrium with solid
c     ierr--error flag:   0 = successful
c                         2 = D > Dmax
c                         8 = x out of range
c                        10 = D and x out of range
c                       160 = CRITP did not converge
c                       161 = SATD did not converge
c     herr--error string (character*255 variable if ierr<>0)
c
c  N.B. kr = 3,4 presently working only for pure components
c
c  either (rhol,xliq) or (rhov,xvap) will correspond to the input state
c  with the other pair corresponding to the other phase in equilibrium
c  with the input state




      subroutine SATH (h,x,kph,nroot,k1,t1,p1,d1,k2,t2,p2,d2,ierr,herr)
c
c  iterate for temperature, pressure, and density given enthalpy along
c  the saturation boundary and the composition
c
c  inputs:
c        h--molar enthalpy [J/mol]
c        x--composition [array of mol frac]
c      kph--flag specifying desired root
c           0 = return all roots along the liquid-vapor line
c           1 = return only liquid VLE root
c           2 = return only vapor VLE roots
c           3 = return liquid SLE root (melting line)
c           4 = return vapor SVE root (sublimation line)
c  outputs:
c    nroot--number of roots.  Value is set to one for kph=1,3,4 if ierr=0
c       k1--phase of first root (1-liquid, 2-vapor, 3-melt, 4-subl)
c       t1--temperature of first root [K]
c       p1--pressure of first root [kPa]
c       d1--molar density of first root [mol/L]
c       k2--phase of second root (1-liquid, 2-vapor, 3-melt, 4-subl)
c       t2--temperature of second root [K]
c       p2--pressure of second root [kPa]
c       d2--molar density of second root [mol/L]
c     ierr--error flag:   0 = successful
c                         2 = h < hmin
c                         4 = h > hmax
c                         8 = h > htrp (for subl input)
c                       160 = CRITP did not converge
c                       161 = SATH did not converge for one root
c                       162 = SATH did not converge for both roots
c     herr--error string (character*255 variable if ierr<>0)
c
c  The second root is always set as the root in the vapor at temperatures
c  below the maximum enthalpy on the vapor saturation line.  If kph is
c  set to 2, and only one root is found in the vapor (this occurs when h<hcrit)
c  the state point will be placed in k2,t2,p2,d2.  If kph=0 and this situation
c  occurred, the first root (k1,t1,p1,d1) would be in the liquid (k1=1, k2=2).
c
c  N.B. kph = 3,4 presently working only for pure components




      subroutine SATE (e,x,kph,nroot,k1,t1,p1,d1,k2,t2,p2,d2,ierr,herr)
```

```
c
c  iterate for temperature, pressure, and density given energy along
c  the saturation boundary and the composition
c
c  inputs:
c        e--molar energy [J/mol]
c        x--composition [array of mol frac]
c      kph--flag specifying desired root
c           0 = return all roots along the liquid-vapor line
c           1 = return only liquid VLE root
c           2 = return only vapor VLE roots
c           3 = return liquid SLE root (melting line)
c           4 = return vapor SVE root (sublimation line)
c  outputs:
c    see SATH for description of outputs




      subroutine SATS (s,x,kph,nroot,k1,t1,p1,d1,k2,t2,p2,d2,
     &                 k3,t3,p3,d3,ierr,herr)
c
c  iterate for temperature, pressure, and density given an entropy along
c  the saturation boundary and the composition
c
c  inputs:
c        s--molar entropy [J/mol-K]
c        x--composition [array of mol frac]
c      kph--flag specifying desired root
c           0 = return all roots along the liquid-vapor line
c           1 = return only liquid VLE root
c           2 = return only vapor VLE roots
c           3 = return liquid SLE root (melting line)
c           4 = return vapor SVE root (sublimation line)
c  outputs:
c    nroot--number of roots.  Set to one for kph=1,3,4 if ierr=0
c       k1--phase of first root (1-liquid, 2-vapor, 3-melt, 4-subl)
c       t1--temperature of first root [K]
c       p1--pressure of first root [kPa]
c       dl--molar density of first root [mol/L]
c       k2--phase of second root (1-liquid, 2-vapor, 3-melt, 4-subl)
c       t2--temperature of second root [K]
c       p2--pressure of second root [kPa]
c       d2--molar density of second root [mol/L]
c       k3--phase of third root (1-liquid, 2-vapor, 3-melt, 4-subl)
c       t3--temperature of third root [K]
c       p3--pressure of third root [kPa]
c       d3--molar density of third root [mol/L]
c     ierr--error flag:   0 = successful
c                         1 = no roots found for specified input phase
c                         2 = s < smin
c                         4 = s > smax
c                         8 = s > strp (for subl input)
c                       160 = CRITP did not converge
c                       161 = SATS did not converge for one root
c                       162 = SATS did not converge for two roots
c                       163 = SATS did not converge for all roots
c     herr--error string (character*255 variable if ierr<>0)
c
c  The second root is always set as the root in the vapor at temperatures
c  below the maximum entropy on the vapor saturation line.  If kph is
c  set to 2, and only one root is found in the vapor (this occurs when s<scrit)
c  the state point will be placed in k2,t2,p2,d2.  If kph=0 and this situation
c  occurred, the first root (k1,t1,p1,d1) would be in the liquid (k1=1, k2=2).
c
c  The third root is the root with the lowest temperature.  For fluids
c  with multiple roots:  When only one root is found in the vapor phase
c  (this happens only at very low temperatures past the region where three
c  roots are located), the value of the root is still placed in
c  k3,t3,p3,d3.  For fluids that never have more than one root (when there
c  is no maximum entropy along the saturated vapor line), the value of the
c  root is always placed in k1,t1,p1,d1.
```

```
c
c  N.B. kph = 3,4 presently working only for pure components




      subroutine CSATK (icomp,t,kph,p,rho,csat,ierr,herr)
c
c  compute the heat capacity along the saturation line as a function of
c  temperature for a given component
c
c  csat can be calculated two different ways:
c     Csat = Cp - T(DvDT)(DPDTsat)
c     Csat = Cp - beta/rho*hvap/(vliq - vvap)
c     where beta is the volume expansivity
c
c  inputs:
c    icomp--component number in mixture (1..nc); 1 for pure fluid
c        t--temperature [K]
c      kph--phase flag: 1 = liquid calculation
c                       2 = vapor calculation
c  outputs:
c        p--saturation pressure [kPa]
c      rho--saturation molar density [mol/L]
c     csat--saturation heat capacity [J/mol-K]




      subroutine DPTSATK (icomp,t,kph,p,rho,csat,dpt,ierr,herr)
c
c  compute the heat capacity and dP/dT along the saturation line as a
c  function of temperature for a given component.  See also subroutine CSATK.
c
c  inputs:
c    icomp--component number in mixture (1..nc); 1 for pure fluid
c        t--temperature [K]
c      kph--phase flag: 1 = liquid calculation
c                       2 = vapor calculation
c  outputs:
c        p--saturation pressure [kPa]
c      rho--saturation molar density [mol/L]
c     csat--saturation heat capacity [J/mol-K] (same as that called from CSATK)
c      dpt--dP/dT along the saturation line [kPa/K]
c           (this is not dP/dT "at" the saturation line for the single phase
c            state, but the change in saturated vapor pressure as the
c            saturation temperature changes.)




      subroutine CV2PK (icomp,t,rho,cv2p,csat,ierr,herr)
c
c  compute the isochoric heat capacity in the two phase (liquid+vapor)
c  region
c
c  inputs:
c    icomp--component number in mixture (1..nc); 1 for pure fluid
c        t--temperature [K]
c      rho--density [mol/l] if known
c           If rho=0, then a saturated liquid state is assumed.
c
c  outputs:
c     cv2p--isochoric two-phase heat capacity [J/mol-K]
c     csat--saturation heat capacity [J/mol-K]
c           (Although there is already a csat routine in REFPROP,
c            it is also returned here.  However, the calculation
c            speed is slower than csat.)
```

Two similar routines are provided for calculating surface tension.
SURTEN is more efficient if the liquid and vapor density and composition
are known (e.g., from a previous call to SATT). If these are not known,
then SURFT may be used.

```
      subroutine SURFT (t,rhol,xl,sigma,ierr,herr)
c
c  compute surface tension
c
c  inputs:
c       t--temperature [K]
c      xl--composition of liquid phase [array of mol frac]
c  outputs:
c     rhol--molar density of liquid phase [mol/L]
c           if rho > 0 use as input value
c                  < 0 call SATT to find density
c    sigma--surface tension [N/m]
c     ierr--error flag:   0 = successful
c                         1 = T < Tmin
c                         8 = x out of range
c                         9 = T and x out of range
c                       120 = CRITP did not converge
c                       121 = T > Tcrit
c                       122 = TPRHO-liquid did not converge in SATT
c                       123 = TPRHO-vapor did not converge in SATT
c                       124 = SATT pure fluid iteration did not converge
c                       128 = SATT mixture iteration did not converge
c     herr--error string if ierr<>0 (character*255)




      subroutine SURTEN (t,rhol,rhov,xl,xv,sigma,ierr,herr)
c
c  compute surface tension
c
c  inputs:
c       t--temperature [K]
c     rhol--molar density of liquid phase [mol/L]
c     rhov--molar density of vapor phase [mol/L]
c           if either rhol or rhov < 0 call SATT to find densities
c       xl--composition of liquid phase [array of mol frac]
c       xv--composition of vapor phase [array of mol frac]
c           (xv is optional input if rhol < 0 or rhov < 0)
c  outputs:
c    sigma--surface tension [N/m]
c     ierr--error flag:   0 = successful
c                         1 = T < Tmin
c                         8 = x out of range
c                         9 = T and x out of range
c                       120 = CRITP did not converge
c                       121 = T > Tcrit
c                       122 = TPRHO-liquid did not converge in SATT
c                       123 = TPRHO-vapor did not converge in SATT
c                       124 = SATT pure fluid iteration did not converge
c                       128 = SATT mixture iteration did not converge
c     herr--error string if ierr<>0 (character*255)
```

FLASH SUBROUTINES

So-called "flash" calculations involve a determination of the
thermodynamic state, given two independent variables plus composition.
In addition to the inputs of temperature and pressure, which is the most
common flash calculation, REFPROP provides routines for virtually all
combinations of temperature, pressure, or density as the first variable
and density, internal energy, enthalpy, entropy, or quality as the
second variable. The combination of enthalpy and entropy is also

supported.

Flash subroutines are provided for cases where the state is known to be
single phase (liquid or vapor) or two-phase (liquid plus vapor), and
also for the general case where the phase is not known. Because of the
many combinations and their parallel structure, these routines are
described in groups. The first two letters of the subroutine name
indicate the independent variables, where

```
      T = temperature [K]
      P = pressure [kPa]
      D = density [mol/L]
      E = internal energy [J/mol]
      H = enthalpy [J/mol]
      S = entropy [J/mol-K]
      Q = vapor quality [moles vapor/total moles]
          or [kg vapor/total kg] depending on the
          value of the input flag kq
```

GENERAL FLASH SUBROUTINES

For cases where the phase is not known, the following routines are available.

```
      subroutine TPFLSH (t,p,z,D,Dl,Dv,x,y,q,e,h,s,cv,cp,w,ierr,herr)
      subroutine TDFLSH (t,D,z,p,Dl,Dv,x,y,q,e,h,s,cv,cp,w,ierr,herr)
      subroutine THFLSH (t,h,z,kr,p,D,Dl,Dv,x,y,q,e,s,cv,cp,w,ierr,herr)
      subroutine TSFLSH (t,s,z,kr,p,D,Dl,Dv,x,y,q,e,h,cv,cp,w,ierr,herr)
      subroutine TEFLSH (t,e,z,kr,p,D,Dl,Dv,x,y,q,h,s,cv,cp,w,ierr,herr)
      subroutine PDFLSH (p,D,z,t,Dl,Dv,x,y,q,e,h,s,cv,cp,w,ierr,herr)
      subroutine PHFLSH (p,h,z,t,D,Dl,Dv,x,y,q,e,s,cv,cp,w,ierr,herr)
      subroutine PSFLSH (p,s,z,t,D,Dl,Dv,x,y,q,e,h,cv,cp,w,ierr,herr)
      subroutine PEFLSH (p,e,z,t,D,Dl,Dv,x,y,q,h,s,cv,cp,w,ierr,herr)
      subroutine HSFLSH (h,s,z,t,p,D,Dl,Dv,x,y,q,e,cv,cp,w,ierr,herr)
      subroutine ESFLSH (e,s,z,t,p,D,Dl,Dv,x,y,q,h,cv,cp,w,ierr,herr)
      subroutine DHFLSH (D,h,z,t,p,Dl,Dv,x,y,q,e,s,cv,cp,w,ierr,herr)
      subroutine DSFLSH (D,s,z,t,p,Dl,Dv,x,y,q,e,h,cv,cp,w,ierr,herr)
      subroutine DEFLSH (D,e,z,t,p,Dl,Dv,x,y,q,h,s,cv,cp,w,ierr,herr)
      subroutine TQFLSH (t,q,z,kq,p,D,Dl,Dv,x,y,e,h,s,cv,cp,w,ierr,herr)
      subroutine PQFLSH (p,q,z,kq,t,D,Dl,Dv,x,y,e,h,s,cv,cp,w,ierr,herr)
c
c  flash calculation given two independent variables and bulk composition
c
c  These routines accept both single-phase and two-phase states as the
c  input; if the phase is known, the specialized routines are faster
c
c  inputs--two of the following as indicated by the first two letters of
c          the subroutine name:
c        t--temperature [K]
c        p--pressure [kPa]
c        e--internal energy [J/mol]
c        h--enthalpy [J/mol]
c        s--entropy [[J/mol-K]
c        q--vapor quality [basis specified by kq]
c           q = 0 indicates saturated liquid
c           q = 1 indicates saturated vapor
c           q < 0 or q > 1 are not allowed and will result in warning
c
c  additional input--required for all routines
c        z--overall (bulk) composition [array of mol frac]
c
c  additional input--only for TQFLSH and PQFLSH
c       kq--flag specifying units for input quality
c           kq = 1 quality on MOLAR basis [moles vapor/total moles]
c           kq = 2 quality on MASS basis [mass vapor/total mass]
c
c  additional input--only for THFLSH, TSFLSH, and TEFLSH
c       kr--flag specifying desired root for multi-valued inputs:
c           1 = return lower density root
c           2 = return higher density root
```

```
c
c  outputs--one, two, or all of the following, depending on the inputs:
c        t--temperature [K]
c        p--pressure [kPa]
c        D--overall (bulk) molar density [mol/L]
c
c  additional outputs--common to all routines
c       Dl--molar density [mol/L] of the liquid phase
c       Dv--molar density [mol/L] of the vapor phase
c           if only one phase is present, Dl = Dv = D
c        x--composition of liquid phase [array of mol frac]
c        y--composition of vapor phase [array of mol frac]
c           if only one phase is present, x = y = z
c
c  additional output--common to all routines except TQFLSH and PQFLSH
c        q--vapor quality on a MOLAR basis [moles vapor/total moles]
c           q < 0 indicates subcooled (compressed) liquid
c           q = 0 indicates saturated liquid
c           q = 1 indicates saturated vapor
c           q > 1 indicates superheated vapor
c           q = 998 superheated vapor, but quality not defined (in most situations, t > Tc)
c           q = 999 indicates supercritical state (t > Tc) and (p > Pc)
c
c  additional outputs--common to all routines, except that input
c                      quantities are not repeated
c        e--overall (bulk) internal energy [J/mol]
c        h--overall (bulk) enthalpy [J/mol]
c        s--overall (bulk) entropy [J/mol-K]
c       cv--isochoric (constant V) heat capacity [J/mol-K]
c       cp--isobaric (constant p) heat capacity [J/mol-K]
c        w--speed of sound [m/s]
c           Cp, w are not defined for 2-phase states
c           in such cases, a flag = -9.99998d6 is returned
c     ierr--error flag:  0 = all inputs within limits
c                       <>0 = one or more inputs outside limits:
c                        -1 = 1.5*tmax > t > tmax
c                         1 = t < tmin or t > 1.5*tmax
c                         2 = D > Dmax or D < 0
c                        -4 = 2*pmax > p > pmax
c                         4 = p < 0 or p > 2*pmax
c                         8 = component composition < 0 or > 1
c                             and/or composition sum < 0 or > 1
c                        16 = p>pmelt
c                       -16 = t<ttrp (important for water)
c           if multiple inputs are outside limits, ierr = abs[sum(ierr)]
c           with the sign determined by the most severe excursion
c           (ierr > 0 indicate an error--calculations not possible,
c            ierr < 0 indicate a warning--results may be questionable)
c     herr--error string (character*255 variable if ierr<>0)
```

SINGLE-PHASE FLASH SUBROUTINES

These routines accept only single-phase states as inputs. They will be faster than the corresponding general routines, but will fail if called with an incorrect phase specification. The phase-specific subroutines also do not check limits, so may fail if called outside the range of the equation of state. The following single-phase routines are available.

```
      subroutine THFL1 (t,h,x,Dmin,Dmax,D,ierr,herr)
      subroutine TSFL1 (t,s,x,Dmin,Dmax,D,ierr,herr)
      subroutine TEFL1 (t,e,x,Dmin,Dmax,D,ierr,herr)
      subroutine PDFL1 (p,rho,x,t,ierr,herr)
      subroutine PHFL1 (p,h,x,kph,t,D,ierr,herr)
      subroutine PSFL1 (p,s,x,kph,t,D,ierr,herr)
      subroutine PEFL1 (p,e,x,kph,t,D,ierr,herr)
      subroutine HSFL1 (h,s,x,Dmin,Dmax,t,D,ierr,herr)
      subroutine DHFL1 (rho,h,x,t,ierr,herr)
```

```
      subroutine DSFL1 (rho,s,x,t,ierr,herr)
      subroutine DEFL1 (rho,e,x,t,ierr,herr)
c
c  inputs--two of the following as indicated by the first two letters of
c          the subroutine name:
c        t--temperature [K]
c        e--internal energy [J/mol]
c        h--enthalpy [J/mol]
c        s--entropy [[J/mol-K]
c      rho--molar density [mol/L]
c
c  additional inputs
c        x--overall (bulk) composition [array of mol frac]
c     Dmin--lower bound on density [mol/L]
c     Dmax--upper bound on density [mol/L]
c      kph--phase flag:  1 = liquid
c                        2 = vapor
c
c  outputs:
c        t--temperature [K] (present only for HSFL1)
c        D--molar density [mol/L]
c     ierr--error flag:  0 = successful
c     herr--error string (character*255 variable if ierr<>0)



The single-phase temperature-pressure flash is called many times by
other routines, and has been optimized for speed; it requires a specific
calling sequence.

      subroutine TPRHO (t,p,x,kph,kguess,rho,ierr,herr)
c
c  iterate for density as a function of temperature, pressure, and
c  composition for a specified phase
c
c**********************************************************************
c  WARNING:
c  Invalid densities will be returned for T & P outside range of validity,
c  i.e., pressure > melting pressure, pressure less than saturation
c  pressure for kph=1, etc.
c
c**********************************************************************
c  inputs:
c        t--temperature [K]
c        p--pressure [kPa]
c        x--composition [array of mol frac]
c      kph--phase flag:  1 = liquid
c                        2 = vapor
c                 N.B.:  0 = stable phase--NOT ALLOWED (use TPFLSH)
c                           (unless an initial guess is supplied for rho)
c                       -1 = force the search in the liquid phase (for metastable points)
c                       -2 = force the search in the vapor phase (for metastable points)
c   kguess--input flag:  1 = first guess for rho provided
c                        0 = no first guess provided
c      rho--first guess for molar density [mol/L], only if kguess = 1
c
c  outputs:
c      rho--molar density [mol/L]
c     ierr--error flag:  0 = successful
c                      200 = CRITP did not converge
c                      201 = illegal input (kph <= 0)
c                      202 = liquid-phase iteration did not converge
c                      203 = vapor-phase iteration did not converge
c     herr--error string (character*255 variable if ierr<>0)
```

TWO-PHASE FLASH SUBROUTINES

```
These routines accept only two-phase (liquid + vapor) states as inputs.
They will be faster than the corresponding general routines, but will
fail if called with an incorrect phase specification.  The
phase-specific subroutines also do not check limits, so may fail if
called outside the range of the equation of state.  The following
two-phase routines are available.

      subroutine TPFL2 (t,p,z,Dl,Dv,x,y,q,ierr,herr)
      subroutine DHFL2 (d,h,z,t,p,Dl,Dv,x,y,q,ierr,herr)
      subroutine DSFL2 (d,s,z,t,p,Dl,Dv,x,y,q,ierr,herr)
      subroutine DEFL2 (d,e,z,t,p,Dl,Dv,x,y,q,ierr,herr)

c  inputs--two of the following
c        t--temperature [K]
c        p--pressure [kPa]
c        d--bulk molar density [mol/L]
c        e--internal energy [J/mol]
c        h--enthalpy [J/mol]
c        s--entropy [[J/mol-K]
c
c  outputs:
c        t--temperature [K] (not present for TPFL2)
c        p--pressure [kPa] (not present for TPFL2)
c       Dl--molar density [mol/L] of the liquid phase
c       Dv--molar density [mol/L] of the vapor phase
c        x--composition of liquid phase [array of mol frac]
c        y--composition of vapor phase [array of mol frac]
c        q--vapor quality on a MOLAR basis [moles vapor/total moles]
c     ierr--error flag:  0 = successful
c     herr--error string (character*255 variable if ierr<>0)



The following two-phase flash routines have the option to pass the dew and
bubble point conditions as inputs if these values are known (from a
previous call to SATT or SATP, for example), these two-phase routines will
be significantly faster than the corresponding general FLSH routines
described above.  Otherwise, the general routines will be more reliable.

      subroutine THFL2 (t,h,z,ksat,pbub,pdew,Dlbub,Dvdew,ybub,xdew,
     &                  p,Dl,Dv,x,y,q,ierr,herr)
      subroutine TSFL2 (t,s,z,ksat,pbub,pdew,Dlbub,Dvdew,ybub,xdew,
     &                  p,Dl,Dv,x,y,q,ierr,herr)
      subroutine TEFL2 (t,e,z,ksat,pbub,pdew,Dlbub,Dvdew,ybub,xdew,
     &                  p,Dl,Dv,x,y,q,ierr,herr)
      subroutine TDFL2 (t,D,z,ksat,pbub,pdew,Dlbub,Dvdew,ybub,xdew,
     &                  p,Dl,Dv,x,y,q,ierr,herr)
      subroutine PDFL2 (p,d,z,ksat,tbub,tdew,Dlbub,Dvdew,ybub,xdew,
     &                  t,Dl,Dv,x,y,q,ierr,herr)
      subroutine PHFL2 (p,h,z,ksat,tbub,tdew,Dlbub,Dvdew,ybub,xdew,
     &                  t,Dl,Dv,x,y,q,ierr,herr)
      subroutine PSFL2 (p,s,z,ksat,tbub,tdew,Dlbub,Dvdew,ybub,xdew,
     &                  t,Dl,Dv,x,y,q,ierr,herr)
      subroutine PEFL2 (p,e,z,ksat,tbub,tdew,Dlbub,Dvdew,ybub,xdew,
     &                  t,Dl,Dv,x,y,q,ierr,herr)
      subroutine TQFL2 (t,q,z,kq,ksat,pbub,pdew,Dlbub,Dvdew,ybub,xdew,
     &                  p,Dl,Dv,x,y,ierr,herr)
      subroutine PQFL2 (p,q,z,kq,ksat,tbub,tdew,Dlbub,Dvdew,ybub,xdew,
     &                  t,Dl,Dv,x,y,ierr,herr)

c  inputs--two of the following
c        t--temperature [K]
c        p--pressure [kPa]
c        D--overall (bulk) molar density [mol/L]
c        e--internal energy [J/mol]
c        h--enthalpy [J/mol]
c        s--entropy [[J/mol-K]
c        q--vapor quality on a MOLAR basis [moles vapor/total moles]
c
c  additional input--only for TQFL2 and PQFL2
c       kq--flag specifying units for input quality
```

```
c           kq = 1 quality on MOLAR basis [moles vapor/total moles]
c           kq = 2 quality on MASS basis [mass vapor/total mass]
c
c  additional inputs
c        z--overall (bulk) composition [array of mol frac]
c     ksat--flag for bubble and dew point limits
c           0 = dew and bubble point limits computed within routine
c           1 = must provide values for following:
c     tbub--bubble point temperature [K] at (p,x=z)
c     tdew--dew point temperature [K] at (p,y=z)
c  --or--
c     pbub--bubble point pressure [kPa] at (t,x=z)
c     pdew--dew point pressure [kPa] at (t,y=z)
c  --and--
c    Dlbub--liquid density [mol/L] at bubble point
c    Dvdew--vapor density [mol/L] at dew point
c     ybub--vapor composition [array of mol frac] at bubble point
c     xdew--liquid composition [array of mol frac] at dew point
c
c  outputs--one of the following, depending on the inputs:
c        t--temperature [K]
c        p--pressure [kPa]
c
c  additional outputs--common to all routines
c       Dl--molar density [mol/L] of the liquid phase
c       Dv--molar density [mol/L] of the vapor phase
c        x--composition of liquid phase [array of mol frac]
c        y--composition of vapor phase [array of mol frac]
c        q--vapor quality on a MOLAR basis [moles vapor/total moles]
c           (not present for TQFL2 and PQFL2)
c     ierr--error flag:  0 = successful
c     herr--error string (character*255 variable if ierr<>0)




      subroutine DQFL2 (d,q,z,kq,t,p,Dl,Dv,x,y,ierr,herr)
c
c  flash calculation given bulk density, quality, and composition
c
c  This routine accepts only two-phase states as input; it is intended
c  primarily for use by the general density-quality flash routine
c  DQFLSH.  It may be called independently if the state is known to be
c  two-phase.  But beware--this routine does not check limits.
c
c  inputs:
c        d--overall (bulk) molar density [mol/L]
c        q--vapor quality on a MOLAR basis [moles vapor/total moles]
c        z--overall (bulk) composition [array of mol frac]
c       kq--flag specifying units for input quality
c           kq = 1 quality on MOLAR basis [moles vapor/total moles]
c           kq = 2 quality on MASS basis [mass vapor/total mass]
c
c  outputs:
c        t--temperature [K]
c        p--pressure [kPa]
c       Dl--molar density [mol/L] of the liquid phase
c       Dv--molar density [mol/L] of the vapor phase
c        x--composition of liquid phase [array of mol frac]
c        y--composition of vapor phase [array of mol frac]
c     ierr--error flag:  0 = successful
c     herr--error string (character*255 variable if ierr<>0)




      subroutine CSTAR (t,p,v,x,cs,ts,Ds,ps,ws,ierr,herr)
c
c  Calculate the critical flow factor, C*, for nozzle flow of a gas
c  (subroutine was originally named CCRIT)
c
c  inputs:
```

```
c        t--temperature [K]
c        p--pressure [kPa]
c        v--plenum velocity [m/s] (should generally be set to 0 for
c                                  calculating stagnation conditions)
c        x--composition [array of mol frac]
c
c  outputs:
c       cs--critical flow factor [dimensionless]
c       ts--nozzle throat temperature [K]
c       Ds--nozzle throat molar density [mol/L]
c       ps--nozzle throat pressure [kPa]
c       ws--nozzle throat speed of sound [m/s]
c     ierr--error flag:  0 = successful
c                      200 = CSTAR did not converge
c                      201 = Final state may be 2-phase
c     herr--error string (character*255 variable if ierr<>0)
c
```

THERMODYNAMIC PROPERTY SUBROUTINES AS F(T,rho,X)

The following routines provide thermodynamic properties as a function of temperature, density, and composition. Typically, one or more of these will be called after finding the temperature and/or density with a call to one of the saturation or flash routines. Note that these routines assume that valid inputs are supplied--no range checking is performed.

```
      subroutine CRITP (x,tcrit,pcrit,Dcrit,ierr,herr)
c
c  critical parameters as a function of composition
c
c  input:
c        x--composition [array of mol frac]
c  outputs:
c    tcrit--critical temperature [K]
c    pcrit--critical pressure [kPa]
c    Dcrit--critical density [mol/L]
c     ierr--error flag:  0 = successful
c                        1 = did not converge
c     herr--error string (character*255 variable if ierr<>0)
```

```
      subroutine THERM (t,rho,x,p,e,h,s,cv,cp,w,hjt)
c
c  compute thermal quantities as a function of temperature, density,
c  and compositions using core functions (Helmholtz free energy, ideal
c  gas heat capacity and various derivatives and integrals)
c
c  Based on derivations in Younglove & McLinden, JPCRD 23 #5, 1994,
c  Appendix A for pressure-explicit equations (e.g. MBWR) and
c  Baehr & Tillner-Roth, Thermodynamic Properties of Environmentally
c  Acceptable Refrigerants, Berlin:  Springer-Verlag (1995) for
c  Helmholtz-explicit equations (e.g. FEQ).
c
c  inputs:
c        t--temperature [K]
c      rho--molar density [mol/L]
c        x--composition [array of mol frac]
c  outputs:
c        p--pressure [kPa]
c        e--internal energy [J/mol]
c        h--enthalpy [J/mol]
c        s--entropy [J/mol-K]
c       Cv--isochoric heat capacity [J/mol-K]
c       Cp--isobaric heat capacity [J/mol-K]
c        w--speed of sound [m/s]
c      hjt--isenthalpic Joule-Thompson coefficient [K/kPa]
```

```
      subroutine THERM2 (t,rho,x,p,e,h,s,cv,cp,w,Z,hjt,A,G,
     &                   xkappa,beta,dPdrho,d2PdD2,dPT,drhodT,drhodP,
     &                   d2PT2,d2PdTD,spare3,spare4)
c
c  compute thermal quantities as a function of temperature, density,
c  and compositions using core functions (Helmholtz free energy, ideal
c  gas heat capacity and various derivatives and integrals)
c
c  this routine is the same as THERM, except that additional properties
c  are calculated
c
c  inputs:
c        t--temperature [K]
c      rho--molar density [mol/L]
c        x--composition [array of mol frac]
c  outputs:
c        p--pressure [kPa]
c        e--internal energy [J/mol]
c        h--enthalpy [J/mol]
c        s--entropy [J/mol-K]
c       Cv--isochoric heat capacity [J/mol-K]
c       Cp--isobaric heat capacity [J/mol-K]
c        w--speed of sound [m/s]
c        Z--compressibility factor (= PV/RT) [dimensionless]
c      hjt--isenthalpic Joule-Thompson coefficient [K/kPa]
c        A--Helmholtz energy [J/mol]
c        G--Gibbs free energy [J/mol]
c   xkappa--isothermal compressibility (= -1/V dV/dP = 1/rho dD/dP) [1/kPa]
c     beta--volume expansivity (= 1/V dV/dT = -1/rho dD/dT) [1/K]
c   dPdrho--derivative dP/drho [kPa-L/mol]
c   d2PdD2--derivative d^2P/drho^2 [kPa-L^2/mol^2]
c      dPT--derivative dP/dT [kPa/K]
c   drhodT--derivative drho/dT [mol/(L-K)]
c   drhodP--derivative drho/dP [mol/(L-kPa)]
c    d2PT2--derivative d2P/dT2 [kPa/K^2]
c   d2PdTD--derivative d2P/dTd(rho) [J/mol-K]
c   sparei--2 space holders for possible future properties



      subroutine THERM0 (t,rho,x,p0,e0,h0,s0,cv0,cp00,w0,A0,G0)
c
c  compute ideal gas thermal quantities as a function of temperature, density,
c  and compositions using core functions
c
c  this routine is the same as THERM, except it only calculates ideal gas
c  properties (Z=1) at any temperature and density
c
c  inputs:
c        t--temperature [K]
c      rho--molar density [mol/L]
c        x--composition [array of mol frac]
c  outputs:
c       p0--pressure [kPa]
c       e0--internal energy [J/mol]
c       h0--enthalpy [J/mol]
c       s0--entropy [J/mol-K]
c      Cv0--isochoric heat capacity [J/mol-K]
c     Cp00--isobaric heat capacity [J/mol-K]
c       w0--speed of sound [m/s]
c       A0--Helmholtz energy [J/mol]
c       G0--Gibbs free energy [J/mol]



      subroutine RESIDUAL (t,rho,x,pr,er,hr,sr,cvr,cpr,Ar,Gr)
```

```
c
c  compute the residual quantities as a function of temperature, density,
c  and compositions (where the residual is the property minus the ideal
c  gas portion).
c
c  inputs:
c        t--temperature [K]
c      rho--molar density [mol/L]
c        x--composition [array of mol frac]
c  outputs:
c       pr--residual pressure [kPa]  (p-rho*R*T)
c       er--residual internal energy [J/mol]
c       hr--residual enthalpy [J/mol]
c       sr--residual entropy [J/mol-K]
c      Cvr--residual isochoric heat capacity [J/mol-K]
c      Cpr--residual isobaric heat capacity [J/mol-K]
c       Ar--residual Helmholtz energy [J/mol]
c       Gr--residual Gibbs free energy [J/mol]



      subroutine ENTRO (t,rho,x,s)
c
c  compute entropy as a function of temperature, density and composition
c  using core functions (temperature derivative of Helmholtz free energy
c  and ideal gas integrals)
c
c  based on derivations in Younglove & McLinden, JPCRD 23 #5, 1994,
c  equations A5, A19 - A26
c
c  inputs:
c        t--temperature [K]
c      rho--molar density [mol/L]
c        x--composition [array of mol frac]
c  output:
c        s--entropy [J/mol-K]



      subroutine ENTHAL (t,rho,x,h)
c
c  compute enthalpy as a function of temperature, density, and
c  composition using core functions (Helmholtz free energy and ideal
c  gas integrals)
c
c  based on derivations in Younglove & McLinden, JPCRD 23 #5, 1994,
c  equations A7, A18, A19
c
c  inputs:
c        t--temperature [K]
c      rho--molar density [mol/L]
c        x--composition [array of mol frac]
c  output:
c        h--enthalpy [J/mol]



      subroutine ENERGY (t,rho,x,e)
c
c  compute energy as a function of temperature, density, and
c  composition using core functions (Helmholtz free energy and ideal
c  gas integrals)
c
c  based on derivations in Younglove & McLinden, JPCRD 23 #5, 1994
c
c  inputs:
c        t--temperature [K]
c      rho--molar density [mol/L]
c        x--composition [array of mol frac]
```

```
c  output:
c        e--energy [J/mol]




      subroutine CVCP (t,rho,x,cv,cp)
c
c  compute isochoric (constant volume) and isobaric (constant pressure)
c  heat capacity as functions of temperature, density, and composition
c  using core functions
c
c  based on derivations in Younglove & McLinden, JPCRD 23 #5, 1994,
c  equation A15, A16
c
c  inputs:
c        t--temperature [K]
c      rho--molar density [mol/L]
c        x--composition [array of mol frac]
c  outputs:
c       cv--isochoric heat capacity [J/mol-K]
c       cp--isobaric heat capacity [J/mol-K]




      subroutine CVCPK (icomp,t,rho,cv,cp)
c
c  compute isochoric (constant volume) and isobaric (constant pressure)
c  heat capacity as functions of temperature for a given component
c
c  analogous to CVCP, except for component icomp, this is used by transport
c  routines to calculate Cv & Cp for the reference fluid (component zero)
c
c  inputs:
c    icomp--component number in mixture (1..nc); 1 for pure fluid
c        t--temperature [K]
c      rho--molar density [mol/L]
c  outputs:
c       cv--isochoric heat capacity [J/mol-K]
c       cp--isobaric heat capacity [J/mol-K]




      subroutine GIBBS (t,rho,x,Ar,Gr)
c
c  compute residual Helmholtz and Gibbs free energy as a function of
c  temperature, density, and composition using core functions
c
c  N.B.  The quantity calculated is
c
c        G(T,rho) - G0(T,P*) = G(T,rho) - G0(T,rho) + RTln(RTrho/P*)
c
c        where G0 is the ideal gas state and P* is a reference pressure
c        which is equal to the current pressure of interest.  Since Gr
c        is used only as a difference in phase equilibria calculations
c        where the temperature and pressure of the phases are equal, the
c        (RT/P*) part of the log term will cancel and is omitted.
c
c        "normal" (not residual) A and G are computed by subroutine AG
c
c  based on derivations in Younglove & McLinden, JPCRD 23 #5, 1994,
c  equations A8 - A12
c
c  inputs:
c        t--temperature [K]
c      rho--molar density [mol/L]
c        x--composition [array of mol frac]
c  outputs:
c       Ar--residual Helmholtz free energy [J/mol]
c       Gr--residual Gibbs free energy [J/mol]
```

```
      subroutine AG (t,rho,x,a,g)
c
c  compute Helmholtz and Gibbs energies as a function of temperature,
c  density, and composition.
c
c  N.B.  These are not residual values (those are calculated by GIBBS).
c
c  inputs:
c        t--temperature [K]
c      rho--molar density [mol/L]
c        x--composition [array of mol frac]
c  outputs:
c        a--Helmholtz energy [J/mol]
c        g--Gibbs free energy [J/mol]



      subroutine PRESS (t,rho,x,p)
c
c  compute pressure as a function of temperature,
c  density, and composition using core functions
c
c  direct implementation of core function of corresponding model
c
c  inputs:
c        t--temperature [K]
c      rho--molar density [mol/L]
c        x--composition [array of mol frac]
c  output:
c        p--pressure [kPa]



      subroutine DPDD (t,rho,x,dpdrho)
c
c  compute partial derivative of pressure w.r.t. density at constant
c  temperature as a function of temperature, density, and composition
c
c  inputs:
c        t--temperature [K]
c      rho--molar density [mol/L]
c        x--composition [array of mol frac]
c  output:
c   dpdrho--dP/drho [kPa-L/mol]



      subroutine DPDDK (icomp,t,rho,dPdrho)
c
c  compute partial derivative of pressure w.r.t. density at constant
c  temperature as a function of temperature and density for a specified
c  component
c
c  analogous to DPDD, except for component icomp, this is used by transport
c  routines to calculate dP/dD for the reference fluid (component zero)
c
c  inputs:
c    icomp--component number in mixture (1..nc); 1 for pure fluid
c        t--temperature [K]
c      rho--molar density [mol/L]
c  output:
c   dPdrho--dP/drho [kPa-L/mol]
```

```
      subroutine DPDD2 (t,rho,x,d2PdD2)
c
c  compute second partial derivative of pressure w.r.t. density at const
c  temperature as a function of temperature, density, and composition
c
c  inputs:
c        t--temperature [K]
c      rho--molar density [mol/L]
c        x--composition [array of mol frac]
c  output:
c   d2pdD2--d^2P/drho^2 [kPa-L^2/mol^2]




      subroutine DPDT (t,rho,x,dpt)
c
c  compute partial derivative of pressure w.r.t. temperature at constant
c  density as a function of temperature, density, and composition
c
c  inputs:
c        t--temperature [K]
c      rho--molar density [mol/L]
c        x--composition [array of mol frac]
c  output:
c      dpt--dP/dT [kPa/K]




      subroutine DPDTK (icomp,t,rho,dpt)
c
c  compute partial derivative of pressure w.r.t. temperature at constant
c  density as a function of temperature and density for a specified component
c
c  analogous to DPDT, except for component icomp, this is used by transport
c  routines to calculate dP/dT
c
c  inputs:
c    icomp--component number in mixture (1..nc); 1 for pure fluid
c        t--temperature [K]
c      rho--molar density [mol/L]
c  output:
c      dpt--dP/dT [kPa/K]




      subroutine DDDP (t,rho,x,drhodp)
c
c  compute partial derivative of density w.r.t. pressure at constant
c  temperature as a function of temperature, density, and composition
c
c  inputs:
c        t--temperature [K]
c      rho--molar density [mol/L]
c        x--composition [array of mol frac]
c  output:
c   drhodp--drho/dP [mol/(L-kPa)]




      subroutine DDDT (t,rho,x,drhodt)
c
c  compute partial derivative of density w.r.t. temperature at constant
c  pressure as a function of temperature, density, and composition
c
c  inputs:
c        t--temperature [K]
c      rho--molar density [mol/L]
```

```
c       x--composition [array of mol frac]
c  output:
c   drhodt--drho/dT [mol/(L-K)]
c
c   d(rho)/d(T) = -d(rho)/dP x dP/dT = -dP/dT / (dP/d(rho))




      subroutine DHD1(t,rho,x,dhdt_d,dhdt_p,dhdd_t,dhdd_p,dhdp_t,dhdp_d)
c
c  compute partial derivatives of enthalpy w.r.t. t, p, or rho at constant
c  t, p, or rho as a function of temperature, density, and composition
c
c  inputs:
c       t--temperature [K]
c     rho--molar density [mol/L]
c       x--composition [array of mol frac]
c  outputs:
c   dhdt_d--dH/dT at constant density [J/(mol-K)]
c   dhdt_p--dH/dT at constant pressure [J/(mol-K)]
c   dhdd_t--dH/drho at constant temperature [(J/mol)/(mol/L)]
c   dhdd_p--dH/drho at constant pressure [(J/mol)/(mol/L)]
c   dhdp_t--dH/dP at constant temperature [J/(mol-kPa)]
c   dhdp_d--dH/dP at constant density [J/(mol-kPa)]




      subroutine FGCTY2 (t,rho,x,f,ierr,herr)
c
c  compute fugacity for each of the nc components of a mixture by
c  analytical differentiation of the dimensionless residual Helmholtz energy
c
c  based on derivations in the GERG-2004 document for natural gas
c
c  inputs:
c       t--temperature [K]
c     rho--molar density [mol/L]
c       x--composition [array of mol frac]
c  outputs:
c       f--array (1..nc) of fugacities [kPa]




      subroutine PHIDERV (i,j,t,rho,x,dadn,dnadn,ierr,herr)
c
c  calculate various derivatives needed for VLE determination
c
c  based on derivations in the GERG-2004 document for natural gas
c
c  inputs:
c       i--component number of which to take derivative
c       j--component number of which to take derivative (can be set to zero)
c       t--temperature (K)
c     rho--density (mol/L)
c       x--composition [array of mol frac]
c  outputs: (where n is mole number)
c     dadn--n*partial(alphar)/partial(ni)                (Eq. 7.16 in GERG)
c    dnadn--partial(n*alphar)/partial(ni)                (Eq. 7.15 in GERG)
c
c     dtdn--n*[partial(Tred)/partial(ni)]/Tred           (Eq. 7.19 in GERG)
c     dvdn--n*[partial(Vred)/partial(ni)]/Vred           (Eq. 7.18 in GERG)
c           (=-n*[partial(Dred)/partial(ni)]/Dred)
c    daddn--del*n*partial(darddel)/partial(ni)           (Eq. 7.17 in GERG)
c          where darddel=partial(alphar)/partial(del)
c   d2adnn--n*partial^2(n*alphar)/partial(ni)/partial(nj) (Eq. 7.46 in GERG)
c  the following are at constant tau and/or del
c    dadxi--partial(alphar)/partial(xi)                  (Eq. 7.21g in GERG)
c   sdadxi--sum[xi*partial(alphar)/partial(xi)]          (Eq. 7.21g in GERG)
c   dadxij--partial^2(alphar)/partial(xi)/partial(xj)    (Eq. 7.21i in GERG)
```

```
c    daddx--del*partial^2(alphar)/partial(xi)/partial(del) (Eq. 7.21j in GERG)
c  daddxii--del*partial^3(alphar)/partial(xi)/partial(xj)/partial(del)
c
c  other calculated variables:
c   d2addn--del*par.(n*(par.(alphar)/par.(n)/par.(del)    (Eq. 7.50 in GERG)
c   d2adtn--tau*par.(n*(par.(alphar)/par.(n)/par.(tau)    (Eq. 7.51 in GERG)
c   d2adxn--par.(n*(par.(alphar)/par.(n)/par.(xj)         (Eq. 7.52 in GERG)




      subroutine FGCTY (t,rho,x,f)
c
c  old routine to compute fugacity for each of the nc components of a mixture
c  by numerical differentiation (using central differences) of the
c  dimensionless residual Helmholtz energy
c
c  based on derivations in E.W. Lemmon, MS Thesis, University of Idaho
c  (1991); section 3.2
c
c  inputs:
c        t--temperature [K]
c      rho--molar density [mol/L]
c        x--composition [array of mol frac]
c  outputs:
c        f--array (1..nc) of fugacities [kPa]




      subroutine CHEMPOT (t,rho,x,u,ierr,herr)
c
c  compute the chemical potentials for each of the nc components of a
c  mixture
c
c  inputs:
c        t--temperature [K]
c      rho--molar density [mol/L]
c        x--composition [array of mol frac]
c  outputs:
c        u--array (1..nc) of the chemical potentials [J/mol]




      subroutine FUGCOF (t,rho,x,phi,ierr,herr)
c
c  compute the fugacity coefficient for each of the nc components of a
c  mixture
c
c  inputs:
c        t--temperature [K]
c      rho--molar density [mol/L]
c        x--composition [array of mol frac]
c  outputs:
c      phi--array (1..nc) of the fugacity coefficients




      subroutine VIRB (t,x,b)
c
c  compute second virial coefficient as a function of temperature
c  and composition.
c
c  inputs:
c        t--temperature [K]
c        x--composition [array of mol frac]
c  outputs:
c        b--second virial coefficient [L/mol]
```

```
      subroutine DBDT (t,x,dbt)
c
c  compute the 1st derivative of B (B is the second virial coefficient) with
c  respect to T as a function of temperature and composition.
c
c  inputs:
c        t--temperature [K]
c        x--composition [array of mol frac]
c  outputs:
c      dbt--1st derivative of B with respect to T [L/mol-K]




      subroutine DBDT2 (t,x,dbt2)
c
c  compute the 2nd derivative of B (B is the second virial coefficient) with
c  respect to T as a function of temperature and composition.
c
c  inputs:
c        t--temperature [K]
c        x--composition [array of mol frac]
c  outputs:
c      dbt2--2nd derivative of B with respect to T [L/mol-K^2]




      subroutine VIRC (t,x,c)
c
c  compute the third virial coefficient as a function of temperature
c  and composition.
c
c  inputs:
c        t--temperature [K]
c        x--composition [array of mol frac]
c  outputs:
c        c--third virial coefficient [(L/mol)^2]




      subroutine DCDT (t,x,dct)
c
c  compute the 1st derivative of C (C is the third virial coefficient) with
c  respect to T as a function of temperature and composition.
c
c  inputs:
c        t--temperature [K]
c        x--composition [array of mol frac]
c  outputs:
c      dct--1st derivative of C with respect to T [(L/mol)^2-K]




      subroutine DCDT2 (t,x,dct2)
c
c  compute the 2nd derivative of C (C is the third virial coefficient) with
c  respect to T as a function of temperature and composition.
c
c  inputs:
c        t--temperature [K]
c        x--composition [array of mol frac]
c  outputs:
c      dct2--2nd derivative of C with respect to T [(L/mol-K)^2]
```

```
      subroutine VIRD (t,x,d)
c
c  compute the fourth virial coefficient as a function of temperature
c  and composition.
c
c  inputs:
c        t--temperature [K]
c        x--composition [array of mol frac]
c  outputs:
c        d--fourth virial coefficient [(L/mol)^3]




      subroutine VIRBA (t,x,ba)
c
c  compute second acoustic virial coefficient as a function of temperature
c  and composition.
c
c  inputs:
c        t--temperature [K]
c        x--composition [array of mol frac]
c  outputs:
c        ba--second acoustic virial coefficient [L/mol]




      subroutine VIRCA (t,x,ca)
c
c  compute third acoustic virial coefficient as a function of temperature
c  and composition.
c
c  inputs:
c        t--temperature [K]
c        x--composition [array of mol frac]
c  outputs:
c        ca--third acoustic virial coefficient [(L/mol)^2]




      subroutine B12 (t,x,b)
c
c  compute b12 as a function of temperature and composition.
c
c  inputs:
c        t--temperature [K]
c        x--composition [array of mol frac]
c  outputs:
c        b--b12 [(L/mol)^2]




      subroutine EXCESS (t,p,x,kph,rho,vE,eE,hE,sE,aE,gE,ierr,herr)
c
c  compute excess properties as a function of temperature, pressure,
c  and composition.
c
c  inputs:
c        t--temperature [K]
c        p--pressure [kPa]
c        x--composition [array of mol frac]
c      kph--phase flag:  1 = liquid
c                        2 = vapor
c                        0 = stable phase
c  outputs:
c       rho--molar density [mol/L] (if input less than 0, used as initial guess)
c        vE--excess volume [L/mol]
c        eE--excess energy [J/mol]
c        hE--excess enthalpy [J/mol]
```

```
c        sE--excess entropy [J/mol-K]
c        aE--excess Helmholtz energy [J/mol]
c        gE--excess Gibbs energy [J/mol]
c      ierr--error flag:  0 = successful
c                        55 = T,p inputs in different phase for the pure fluids
c      herr--error string (character*255 variable if ierr<>0)




      subroutine FPV (t,rho,p,x,f)
c
c  Compute the supercompressibility factor, Fpv.
c
c  inputs:
c        t--temperature [K]
c        p--pressure [kPa]
c      rho--molar density [mol/L]
c        x--composition [array of mol frac]
c  outputs:
c        f--Fpv = sqrt[Z(60 F, 14.73 psia)/Z(T,P)]




      subroutine RMIX2 (x,Rgas)
c
c  Return the gas "constant" as a combination of the gas constants for
c  the pure fluids
c
c  inputs:
c        x--composition [array of mol frac]
c  outputs:
c     Rgas--gas constant [J/mol-K]




      subroutine THERM3 (t,rho,x,
     &           xkappa,beta,xisenk,xkt,betas,bs,xkkt,throtte,pi,spht)
c
c  Compute miscellaneous thermodynamic properties
c
c  inputs:
c        t--temperature [K]
c      rho--molar density [mol/L]
c        x--composition [array of mol frac]
c
c  outputs:
c   xkappa--Isothermal compressibility [1/kPa]
c     beta--Volume expansivity [1/K]
c   xisenk--Isentropic expansion coefficient [-]
c      xkt--Isothermal expansion coefficient [-]
c    betas--Adiabatic compressibility [1/kPa]
c       bs--Adiabatic bulk modulus [kPa]
c     xkkt--Isothermal bulk modulus [kPa]
c   throtte--Isothermal throttling coefficient [L/mol]
c       pi--Internal pressure [kPa]
c     spht--Specific heat input [J/mol]




      subroutine HEAT (t,rho,x,hg,hn,ierr,herr)
c
c  Compute the ideal gas gross and net heating values.
c
c  inputs:
c        t--temperature [K]
c      rho--molar density [mol/L]
c        x--composition [array of mol frac]
c
```

```
c  outputs:
c       hg--gross (or superior) heating value [J/mol]
c       hn--net (or inferior) heating value [J/mol]
c     ierr--error flag:  0 = successful
c                        1 = error in chemical formula
c                        2 = not all heating values available
c     herr--error string (character*255 variable if ierr<>0)
```

TRANSPORT PROPERTY SUBROUTINE

```
      subroutine TRNPRP (t,rho,x,eta,tcx,ierr,herr)
c
c  compute the transport properties of thermal conductivity and
c  viscosity as functions of temperature, density, and composition
c
c  inputs:
c        t--temperature [K]
c      rho--molar density [mol/L]
c        x--composition array [mol frac]
c  outputs:
c      eta--viscosity (uPa.s)
c      tcx--thermal conductivity (W/m.K)
c     ierr--error flag:  0 = successful
c                      -31 = temperature out of range for conductivity
c                      -32 = density out of range for conductivity
c                      -33 = T and D out of range for conductivity
c                      -41 = temperature out of range for viscosity
c                      -42 = density out of range for viscosity
c                      -43 = T and D out of range for viscosity
c                      -51 = T out of range for both visc and t.c.
c                      -52 = D out of range for both visc and t.c.
c                      -53 = T and/or D out of range for visc and t.c.
c                       39 = model not found for thermal conductivity
c                       40 = model not found for thermal conductivity or viscosity
c                       49 = model not found for viscosity
c                       50 = ammonia/water mixture (no properties calculated)
c                       51 = exactly at t, rhoc for a pure fluid; k is infinite
c                  -58,-59 = ECS model did not converge
c     herr--error string (character*255 variable if ierr<>0)
```

MISCELLANEOUS PROPERTIES

The following routines return the dielectric constant, melting line, and sublimation line. These properties are not available for all fluids in REFPROP.

```
      subroutine DIELEC (t,rho,x,de)
c
c  compute the dielectric constant as a function of temperature, density,
c  and composition.
c
c  inputs:
c        t--temperature [K]
c      rho--molar density [mol/L]
c        x--composition [array of mol frac]
c   output:
c       de--dielectric constant
```

```
      subroutine MELTT (t,x,p,ierr,herr)
c
c  compute the melting line pressure as a function of temperature
c  and composition.
```

```
c
c  inputs:
c        t--temperature [K]
c        x--composition [array of mol frac]
c   output:
c        p--melting line pressure [kPa]
c        ierr--error flag:  0 = successful
c        herr--error string (character*255 variable if ierr<>0)




      subroutine MELTP (p,x,t,ierr,herr)
c
c  compute the melting line temperature as a function of pressure
c  and composition.
c
c  inputs:
c        p--melting line pressure [kPa]
c        x--composition [array of mol frac]
c   output:
c        t--temperature [K]
c        ierr--error flag:  0 = successful
c        herr--error string (character*255 variable if ierr<>0)




      subroutine SUBLT (t,x,p,ierr,herr)
c
c  compute the sublimation line pressure as a function of temperature
c  and composition.
c
c  inputs:
c        t--temperature [K]
c        x--composition [array of mol frac]
c   output:
c        p--sublimation line pressure [kPa]
c        ierr--error flag:  0 = successful
c        herr--error string (character*255 variable if ierr<>0)




      subroutine SUBLP (p,x,t,ierr,herr)
c
c  compute the sublimation line temperature as a function of pressure
c  and composition.
c
c  inputs:
c        p--sublimation line pressure [kPa]
c        x--composition [array of mol frac]
c   output:
c        t--temperature [K]
c        ierr--error flag:  0 = successful
c        herr--error string (character*255 variable if ierr<>0)



UTILITY SUBROUTINES

The following "utility" routines provide fluid constants, such as the
molar mass, and provide conversions between mass and molar quantities.

      subroutine INFO (icomp,wmm,ttrp,tnbpt,tc,pc,Dc,Zc,acf,dip,Rgas)
c
c  provides fluid constants for specified component
c
c  input:
c    icomp--component number in mixture; 1 for pure fluid
c  outputs:
c     wmm--molecular weight [g/mol]
```

```
c     ttrp--triple point temperature [K]
c    tnbpt--normal boiling point temperature [K]
c       tc--critical temperature [K]
c       pc--critical pressure [kPa]
c       Dc--critical density [mol/L]
c       Zc--compressibility at critical point [pc/(Rgas*Tc*Dc)]
c      acf--acentric factor [-]
c      dip--dipole moment [debye]
c     Rgas--gas constant [J/mol-K]



      subroutine NAME (icomp,hname,hn80,hcas)
c
c  provides name information for specified component
c
c  input:
c    icomp--component number in mixture; 1 for pure fluid
c  outputs:
c    hname--component name [character*12]
c     hn80--component name--long form [character*80]
c     hcas--CAS (Chemical Abstracts Service) number [character*12]



      function WMOL (x)
c
c  molecular weight for a mixture of specified composition
c
c  input:
c        x--composition array [array of mol frac]
c
c  output (as function value):
c     WMOL--molar mass [g/mol], a.k.a. "molecular weight"



      subroutine XMASS (xmol,xkg,wmix)
c
c  converts composition on a mole fraction basis to mass fraction
c
c  input:
c     xmol--composition array [array of mol frac]
c  outputs:
c      xkg--composition array [array of mass frac]
c     wmix--molar mass of the mixture [g/mol], a.k.a. "molecular weight"



      subroutine XMOLE (xkg,xmol,wmix)
c
c  converts composition on a mass fraction basis to mole fraction
c
c  input:
c      xkg--composition array [array of mass frac]
c  outputs:
c     xmol--composition array [array of mol frac]
c     wmix--molar mass of the mixture [g/mol], a.k.a. "molecular weight"



      subroutine LIMITX (htyp,t,D,p,x,tmin,tmax,Dmax,pmax,ierr,herr)
c
c  returns limits of a property model as a function of composition
c  and/or checks input t, D, p against those limits
c
c  Pure fluid limits are read in from the .fld files; for mixtures, a
```

```
c  simple mole fraction weighting in reduced variables is used.
c
c  Attempting calculations below the mininum temperature and/or above
c  the maximum density will result in an error.  These will often
c  correspond to a physically unreasonable state; also many equations of
c  state do not extrapolate reliably to lower T's and higher D's.
c
c  A warning is issued if the temperature is above the maximum but below
c  1.5 times the maximum; similarly pressures up to twice the maximum
c  result in only a warning. Most equations of state may be
c  extrapolated to higher T's and P's.  Temperatures and/or pressures
c  outside these extended limits will result in an error.
c
c  When calling with an unknown temperature, set t to -1 to avoid performing
c  the melting line check
c
c  inputs:
c     htyp--flag indicating which models are to be checked [character*3]
c           'EOS':  equation of state for thermodynamic properties
c           'ETA':  viscosity
c           'TCX':  thermal conductivity
c           'STN':  surface tension
c        t--temperature [K]
c        D--molar density [mol/L]
c        p--pressure [kPa]
c        x--composition array [mol frac]
c     N.B.--all inputs must be specified, if one or more are not
c           available, (or not applicable as in case of surface tension)
c           use reasonable values, such as:
c           t = tnbp
c           D = 0
c           p = 0
c  outputs:
c     tmin--minimum temperature for model specified by htyp [K]
c     tmax--maximum temperature [K]
c     Dmax--maximum density [mol/L]
c     pmax--maximum pressure [kPa]
c     ierr--error flag:  0 = all inputs within limits
c                       <>0 = one or more inputs outside limits:
c                        -1 = 1.5*tmax > t > tmax
c                         1 = t < tmin or t > 1.5*tmax
c                         2 = D > Dmax or D < 0
c                        -4 = 2*pmax > p > pmax
c                         4 = p < 0 or p > 2*pmax
c                         8 = component composition < 0 or > 1
c                             and/or composition sum < 0 or > 1
c                        16 = p>pmelt
c                       -16 = t<ttrp (important for water)
c           if multiple inputs are outside limits, ierr = abs[sum(ierr)]
c           with the sign determined by the most severe excursion
c           (ierr > 0 indicate an error--calculations not possible,
c            ierr < 0 indicate a warning--results may be questionable)
c     herr--error string (character*255 variable if ierr<>0)




      subroutine LIMITK (htyp,icomp,t,D,p,tmin,tmax,Dmax,pmax,ierr,herr)
c
c  returns limits of a property model (read in from the .fld files) for
c  a mixture component and/or checks input t, D, p against those limits
c
c  This routine functions in the same manner as LIMITX except that the
c  composition x is replaced by the component number icomp.
c
c  Attempting calculations below the minimum temperature and/or above
c  the maximum density will result in an error.  These will often
c  correspond to a physically unreasonable state; also many equations of
c  state do not extrapolate reliably to lower T's and higher D's.
c
c  A warning is issued if the temperature is above the maximum but below
c  1.5 times the maximum; similarly pressures up to twice the maximum
```

```
c  result in only a warning. Most equations of state may be
c  extrapolated to higher T's and P's.  Temperatures and/or pressures
c  outside these extended limits will result in an error.
c
c  inputs:
c     htyp--flag indicating which models are to be checked [character*3]
c           'EOS':  equation of state for thermodynamic properties
c           'ETA':  viscosity
c           'TCX':  thermal conductivity
c           'STN':  surface tension
c    icomp--component number in mixture; 1 for pure fluid
c        t--temperature [K]
c        D--molar density [mol/L]
c        p--pressure [kPa]
c     N.B.--all inputs must be specified, if one or more are not
c           available, (or not applicable as in case of surface tension)
c           use reasonable values, such as:
c           t = tnbp
c           D = 0
c           p = 0
c  outputs:
c     tmin--minimum temperature for model specified by htyp [K]
c     tmax--maximum temperature [K]
c     Dmax--maximum density [mol/L]
c     pmax--maximum pressure [kPa]
c     ierr--error flag:  0 = all inputs within limits
c                       <>0 = one or more inputs outside limits:
c                        -1 = 1.5*tmax > t > tmax
c                         1 = t < tmin or t > 1.5*tmax
c                         2 = D > Dmax or D < 0
c                        -4 = 2*pmax > p > pmax
c                         4 = p < 0 or p > 2*pmax
c                        16 = p>pmelt
c                       -16 = t<ttrp (important for water)
c           if multiple inputs are outside limits, ierr = abs[sum(ierr)]
c           with the sign determined by the most severe excursion
c           (ierr > 0 indicate an error--calculations not possible,
c            ierr < 0 indicate a warning--results may be questionable)
c     herr--error string (character*255 variable if ierr<>0)




      subroutine LIMITS (htyp,x,tmin,tmax,Dmax,pmax)
c
c  returns limits of a property model as a function of composition
c
c  Pure fluid limits are read in from the .fld files; for mixtures, a
c  simple mole fraction weighting in reduced variables is used.
c
c  inputs:
c     htyp--flag indicating which models are to be checked [character*3]
c           'EOS':  equation of state for thermodynamic properties
c           'ETA':  viscosity
c           'TCX':  thermal conductivity
c           'STN':  surface tension
c        x--composition array [mol frac]
c  outputs:
c     tmin--minimum temperature for model specified by htyp [K]
c     tmax--maximum temperature [K]
c     Dmax--maximum density [mol/L]
c     pmax--maximum pressure [kPa]




      subroutine ERRMSG (ierr,herr)
c
c  write error messages to default output; this subroutine should be
c  called immediately after any call to a subroutine which potentially
c  can error out
c
```

```
c  inputs:
c     ierr--error flag:  0 = successful (no message will be written)
c                       <0 = warning
c                       >0 = error
c     herr--error string (character*255 variable)
c
c  outputs:
c     if iprnterr in common block prnterr is equal to zero:
c     error string written to default output
c
c     if iprnterr is equal to 1:
c     error string written to screen if ierr is positive
c
c     if iprnterr is equal to -1:
c     error string written to screen
c
c     if iprnterr is equal to 3, -3:
c     same as 1, -1, but program also pauses
c
c     Note:  no information should be written to the screen
c     when compiling the DLL.




      subroutine QMASS (qmol,xl,xv,qkg,xlkg,xvkg,wliq,wvap,ierr,herr)
c
c  converts quality and composition on a mole basis to a mass basis
c
c  inputs:
c     qmol--molar quality [moles vapor/total moles]
c           qmol = 0 indicates saturated liquid
c           qmol = 1 indicates saturated vapor
c           0 < qmol < 1 indicates a two-phase state
c           qmol < 0 or qmol > 1 are not allowed and will result in warning
c       xl--composition of liquid phase [array of mol frac]
c       xv--composition of vapor phase [array of mol frac]
c  outputs:
c      qkg--quality on mass basis [mass of vapor/total mass]
c     xlkg--mass composition of liquid phase [array of mass frac]
c     xvkg--mass composition of vapor phase [array of mass frac]
c     wliq--molecular weight of liquid phase [g/mol]
c     wvap--molecular weight of vapor phase [g/mol]
c     ierr--error flag:  0 = all inputs within limits
c           -19:  input q < 0 or > 1
c     herr--error string (character*255 variable if ierr<>0)




      subroutine QMOLE (qkg,xlkg,xvkg,qmol,xl,xv,wliq,wvap,ierr,herr)
c
c  converts quality and composition on a mass basis to a molar basis
c
c  inputs:
c      qkg--quality on mass basis [mass of vapor/total mass]
c           qkg = 0 indicates saturated liquid
c           qkg = 1 indicates saturated vapor
c           0 < qkg < 1 indicates a two-phase state
c           qkg < 0 or qkg > 1 are not allowed and will result in warning
c     xlkg--mass composition of liquid phase [array of mass frac]
c     xvkg--mass composition of vapor phase [array of mass frac]
c  outputs:
c     qmol--quality on mass basis [mass of vapor/total mass]
c       xl--molar composition of liquid phase [array of mol frac]
c       xv--molar composition of vapor phase [array of mol frac]
c     wliq--molecular weight of liquid phase [g/mol]
c     wvap--molecular weight of vapor phase [g/mol]
c     ierr--error flag:  0 = all inputs within limits
c           -19:  input q < 0 or > 1
c     herr--error string (character*255 variable if ierr<>0)
```