USER-SPECIFIC ROUTINES

The body of the code required to compile `multimode.4.9.0.f' is contained in the routines `XXX.vscf.4.9.0.f' or earlier modules. Apart from the key module `memo.vscf.4.9.f', the remainder of these routines are `black box' and should not require any further change. In order to produce his own working version of `multimode', the user must modify the routine `memo.vscf.4.9.f', and supply his own routine `user.XXX.f', tailored to the molecule under inspection. For reaction path studies, he will also have to supply a routine `react.XXX.f'. A brief summary of `memo.vscf.4.9.f', `user.XXX.f' and `react.XXX.f' is contained in Parts A, B and C, respectively, of this document. The user is strongly advised to study Parts B and C of this document together with the relevant routines which are supplied for the test programs appearing in GUIDE.DOC.

A. memo.vscf.4.9.f

The module `memo.vscf.4.9.f' sets the size of RAM to be used, and it also opens and assigns all input/output files. The RAM is set by the statement

```
PARAMETER (MAXSIZ=XXXX)
```

If the current RAM specified by MAXSIZ is exceeded during a run of `multimode', the program stops with the message:

`MEMORY EXHAUSTED'

followed by the current and projected size of MAXSIZ. The user should recompile `memo.vscf.4.9.f' with the new value of MAXSIZ and try again.

There are many files used in `multimode'. All files are opened in `memo.f'; the user must modify these OPEN statements in order to write them onto a disc of his/her choosing. All files except units 1 and 2 are written UNFORMATTED. The definitions of the files are as follows:

```
Unit 1:   Contains the standard input data for `multimode' in FREE FORMAT
Unit 2:   Contains the standard FORMATTED output from `multimode'
Unit 7:   Temporary file used to hold potential parameters in direct dynamics
Unit 20:  Lanczos file holding hamiltonian half-matrix < LAN20
Unit 21:  Basic rovibrational integrals for one-mode coupling
Unit 22:  Basic rovibrational integrals for two-mode coupling
Unit 23:  Basic rovibrational integrals for three-mode coupling
Unit 24:  Basic rovibrational integrals for four-mode coupling
Unit 25:  Basic rovibrational integrals for five-mode coupling
```

Unit 26: Basic rovibrational integrals for six-mode coupling
Unit 29: Temporary file holding VSCF coefficients for VSCF-CI
Unit 30: Temporary file holding VCI coefficients
Units 31 to 39: Basic rovibrational integrals for coriolis terms
Unit 40: Contains the final VSCF information in each UNFORMATTED record
State number, assignment, J, Ka, Kc, energy

Units 41 to 44: Scratch files used in Dump & Restart procedure
Unit 45: Contains the final CI information in each UNFORMATTED record
State number, assignment, J, Ka, Kc, energy
For BOTH units 40 and 45,
State number is an (integer) quantity representing the numbering of the VSCF or CI state
Assignment is an (integer) sequence of NMODE numbers which represnt the number of quanta in each Normal mode (see INPUT Section for definition of NMODE)
J is the (integer) total angular momentum quantum number
Ka is the (integer) asymmetric top quantum number along the principal Z-axis
Kc is the (integer) asymmetric top quantum number along the principal X-axis
energy is the (double precision) energy of the VSCF or CI state in cm-1 relative to the reference energy EVLJ0 (see INPUT Section). In the event that EVLJ0 is given a value of ZERO, the Band origin of the zero point level is written as its absolute energy in cm-1. In this case, all other energies are relative to this zero point energy
Units 46 to 50: Lanczos files holding Hamiltonian half-matrix > LAN20
Units 51 to 57: Lanczos work files
Unit 58: VCI dump scratch file
Unit 59: VCI dump scratch file
Unit 60: VCI dump file
Unit 61: Rigid-rotor energies at each one-mode grid point
Unit 62: Rigid-rotor energies at each two-mode grid point
Unit 63: Rigid-rotor energies at each three-mode grid point
Unit 64: Rigid-rotor energies at each four-mode grid point
Unit 65: Scratch file
Unit 70: Scratch file
Unit 71: Potential energy at each one-mode grid point
Unit 72: Potential energy at each two-mode grid point
Unit 73: Potential energy at each three-mode grid point
Unit 74: Potential energy at each four-mode grid point
Unit 75: Potential energy at each five-mode grid point
Unit 76: Potential energy at each six-mode grid point

Unit 81: Vibrational Coriolis coupling data at each one-mode grid point
Unit 82: Vibrational Coriolis coupling data at each two-mode grid point
Unit 83: Vibrational Coriolis coupling data at each three-mode grid point
Unit 84: Vibrational Coriolis coupling data at each four-mode grid point
Unit 91: Rotational Coriolis coupling data at each one-mode grid point
Unit 92: Rotational Coriolis coupling data at each two-mode grid point
Unit 93: Rotational Coriolis coupling data at each three-mode grid point
Unit 94: Rotational Coriolis coupling data at each four-mode grid point

For the sets of files (units 21, 22, 23, 24, 25, 26), (units 61, 62, 63, 64), (units 71, 72, 73, 74, 75, 76), (units 81, 82, 83, 84), (units 91, 92, 93, 94), the sizes grow permutationally from one-mode to four-mode (or six-mode) coupling.

B. user.XXX.f

The modules user.XXX.f contain nine routines which may be modified by the user in order to interface with his/her (a) potential energy, (b) dipole moment or (c) Reaction Path routines, and (d) produce the correct symmetry-adapted normal coordinate vectors and principal axis geometries in cases of symmetric or spherical top molecules.

1. USERIN
2. GETPOT
3. GETQPT
4. GETDIP
5. GETQDT
6. GETAPT
7. MINPOT
8. ROTATE
9. RTGEOM

(a) Potential Energy interface routines

The calling sequences to these routines are:

1.   SUBROUTINE USERIN

USERIN can be used to read (for example) any data that the user may require to define his potential. If the user wishes to append the standard input data file to `multimode' with his own data, and/or output any information to the

standard output file produced by `multimode', he may use INP and/or IOUT in the COMMON block

```
COMMON/FILASS/IOUT,INP
```

respectively. If no such data are required, then USERIN may be treated as a dummy routine which immediately exits, but it is ALWAYS called by `multimode'.

---

2. SUBROUTINE GETPOT(V,NATOM,XX,RR)

where V is a DOUBLE PRECISION variable which must contain the potential energy in Hartrees on exit. NATOM is an INTEGER variable denoting the number of atoms in the molecule, and XX, RR are DOUBLE PRECISION arrays of dimensions

```
DIMENSION XX(NATOM,3),RR(NATOM,NATOM)
```

XX contains the instantaneous Cartesian coordinates of the NATOM atoms, where XX(NATOM,1), XX(NATOM,2), XX(NATOM,3) are the x-, y-, z-coordinates. RR is reserved space for the internuclear distances between the atoms. If bond distances are required, the user may call the utility routine BONDS from within GETPOT.

```
SUBROUTINE BONDS(NATOM,RR,XX)
```

GETPOT is used as an interface with the user's own routine(s) which evaluate the potential at the Cartesian coordinates held in XX(NATOM,3) (or at the internuclear bond distances held in RR(NATOM,NATOM) which are returned from a user-initiated call to BONDS, if required). The Cartesian coordinates XX(NATOM,3) are passed to GETPOT for instantaneous configurations of the nuclei.

If the arrays XX or RR are required by the user in his own routine(s), they must be passed, along with NATOM, in the calling sequence(s) to his routine(s), and must be dimensioned: XX(NATOM,3),RR(NATOM,NATOM) in the called routine(s).

The COMMON block

```
COMMON/FUNDAM/WAVENM,ATTOJ,BOHR,ELMASS,RAD
```

contains some useful fundamental constants:

```
WAVENM:  Conversion from Hartrees to wavenumbers
ATTOJ:   Conversion from atojoules to hartrees
BOHR:    Conversion from Bohr to Angstroms
ELMASS:  Conversion from Unified atomic mass units to electron mass units
RAD:     Conversion from radians to degrees
```

which have been initialized to the latest IUPAC standards. These are for use

by the user, if required. These constants require, for example, that Hartrees * WAVENM = wavenumbers, etc.

Subroutine GETPOT is called to evaluate the potential which is defined to be in internal coordinates (IWHICH > 0)

---

3. SUBROUTINE GETQPT(V,NMODE,QQ,XTANH)

where V is a DOUBLE PRECISION variable which must contain the potential energy in Hartrees on exit. NMODE is an INTEGER variable denoting the number of vibrational normal modes in the molecule, and QQ and XTANH are DOUBLE PRECISION arrays of dimensions

```
DIMENSION QQ(NMODE),XTANH(NMODE)
```

QQ contains the NMODE instantaneous mass-weighted normal coordinates Q, and XTANH contains the parameters GAMMA(NMODE), defined for the asymptotic value of the Gauss quadrature point Q(MAX). This parameter is derived from input asymptotic values of the quantity GAMMA.Q(MAX). This is then used to redefine the coordinate

```
Y = tanh(GAMMA.QQ)
```

This coordinate is Morse-like at large displacements and may be preferred to the normal coordinate Q. The user may wish to take advantage of this parameter during the construction of his potential energy routine.

Subroutine GETQPT is called to evaluate the potential defined to be in normal coordinates (IWHICH < 0, MOLPRO = 0). This choice gives the user full control over the potential function, rather than relying on the supplied format for IWHICH=0 (see INPUT.DOC).

****

(b) Dipole moment interface routines

---

4. SUBROUTINE GETDIP(V,NATOM,XX,RR,IDIP)

where V is a DOUBLE PRECISION variable which must contain the dipole moment component IDIP in hartrees on exit. NATOM is an INTEGER variable denoting the number of atoms in the molecule, and XX, RR are DOUBLE PRECISION arrays of dimensions

```
DIMENSION XX(NATOM,3),RR(NATOM,NATOM)
```

XX contains the instantaneous Cartesian coordinates of the NATOM atoms, where XX(NATOM,1), XX(NATOM,2), XX(NATOM,3) are the x-, y-, z-coordinates. RR is reserved space for the internuclear distances between the atoms. If bond distances are required, the user may call the utility routine BONDS from within GETDIP. Apart from the parameter IDIP, GETDIP is identical in form to the routine GETPOT in 2 above. The parameter IDIP takes on the values 1,2 for triatomic molecules, or 1,2,3 for larger molecules, and represents the A1, B2, B1 (or A2) components of the dipoles in C2v.

Subroutine GETDIP is called to evaluate the dipole moment, defined to be in internal coordinates, during a Restart (IWHICH > 0, LDUMP = 1), following an earlier Dump of consecutive functions (IDUMP > 0) or specific functions (IDUMP < 0).

---

5. SUBROUTINE GETQDT(V,NMODE,QQ,IDIP)

where V is a DOUBLE PRECISION variable which must contain the dipole moment component IDIP in Hartrees on exit. NMODE is an INTEGER variable denoting the number of vibrational normal modes in the molecule, and QQ is a DOUBLE PRECISION array of dimension

```
DIMENSION QQ(NMODE)
```

QQ contains the NMODE instantaneous mass-weighted normal coordinates Q. Apart from the parameters IDIP and XTANH, GETQDT is identical in form to the routine GETQPT, but the Morse-like coordinate

```
Y = tanh(GAMMA.QQ)
```

is not used within GETQDT. The parameter IDIP takes on the values 1,2 for triatomic molecules, or 1,2,3 for larger molecules, and represents the A1,B2,B1 (or A2) components of the dipoles in C2v.

Subroutine GETQDT is called to evaluate the dipole moment, defined to be in normal coordinates, during a Restart (IWHICH < 0, MOLPRO = 0, LDUMP = 1), following an earlier Dump of consecutive functions (IDUMP > 0) or specific functions (IDUMP < 0).

****

(c) Reaction Path interface routines

---

6. SUBROUTINE GETAPT(V,NATOM,XX,RR,QQ,NMODE,ITAU)

where V is a DOUBLE PRECISION variable which must contain the potential energy in Hartrees on exit. NATOM and NMODE are INTEGER variables denoting the number of atoms and number of vibrational normal modes in the molecule, respectively. XX, RR and QQ are DOUBLE PRECISION arrays of dimensions

```
     DIMENSION XX(NATOM,3),RR(NATOM,NATOM),QQ(NMODE)
```

XX contains the instantaneous Cartesian coordinates of the NATOM atoms, where XX(NATOM,1), XX(NATOM,2), XX(NATOM,3) are the x, y, z coordinates. RR is reserved space for the internuclear distances between the atoms. QQ contains the NMODE instantaneous mass-weighted normal coordinates Q. During a Reaction Path analysis, the Reaction Path coordinate (denoted by the parameter IREACT on input), is moved to the position NMODE. The 3N-7 normal coordinates orthogonal to the Reaction Path coordinate therefore occupy positions QQ(1) to QQ(NMODE-1). The parameter ITAU is the instantaneous point along the Reaction Path for which the potential is required.

Subroutine GETAPT is called to evaluate the potential defined as ab initio for a Reaction Path (IWHICH > 0, IREACT < 0).

---

7. SUBROUTINE MINPOT(TAU,NATOM,XR,RR,TAUE)

where NATOM is an integer variable denoting the number of atoms in the molecule and XR, RR are DOUBLE PRECISION arrays of dimensions

```
     DIMENSION XR(NATOM,3),RR(NATOM,NATOM)
```

The parameter TAU is the torsional displacement in radians from the initial point along the Reaction Path. The parameter TAUE is used to return the actual value of the torsional angle corresponding to TAU. The array RR is reserved space for the internuclear distances between the atoms, if required. The array XR is returned to the calling routine `react.XXX.f' (below) containing the Cartesian coordinates corresponding to the minimum energy for the displacement torsional angle TAU.

Subroutine MINPOT is called to evaluate the minimum potential, defined to be in internal coordinates, for a displacement of TAU radians along the Reaction Path (IWHICH > 0, IREACT > 0).

****

(d) Normal coordinate and principal axis rotation routines

The calling sequences to these routines are:

8. SUBROUTINE ROTATE(NATOM,XO,OMEGA,NMODE,XL,WAVENM)

with the interface

```
     DIMENSION XO(NATOM,3),OMEGA(NMODE),XL(NATOM,NMODE,3)
```

---

9. SUBROUTINE RTGEOM(NATOM,XX,E,WAVENM)

with the interface

```
     DIMENSION XX(NATOM,3),E(3)
     COMMON/MOMI/XK(3,3),XMU(3,3)
```

In the majority of cases when asymmetric tops will be under observation, routines ROTATE and RTGEOM will simply EXIT.

In cases where there are E symmetries (symmetric tops) or F symmetries (spherical tops), the user must supply routines to get correct symmetry-adapted Normal Coordinate eigenvectors (ROTATE) and/or a principal axis geometry which reflects the symmetry of the molecule (RTGEOM).

In ROTATE, XO(NATOM,3) is the equilibrium geometry, OMEGA(NMODE) are the omegas of the Normal Coordinate analysis, and XL(NATOM,NMODE,3) are the eigenvectors of the Normal Coordinate analysis. It will be observed, in certain special cases, that two or three values of OMEGA will be identical (subject to a suitable tolerance). In this case, the eigenvectors of the Normal coordinate analysis for these degenerate modes will be scrambled. If it is desired to analyze the rovibrational energies in Cs or higher symmetry, these vectors must be 'cleaned'. Depending on how the user has defined the equilibrium geometry of the molecule which reflects this symmetry, he must choose two displacements which must be equal (subject to sign) in his chosen coordinate system. He then rotates the degenerate vectors in the usual way. In cases when there are three degenerate levels, a further rotation will usually be required, having first 'cleaned' two vectors. Subroutine ROTATE contains code for two rotations of the vectors of triply-degenerate OMEGAs, and this can be modified for any system that will be encountered by 'Multimode'.

In RTGEOM, XX(NATOM,3) is the (unrotated) principlal axis geometry, E(3) are the eigenvalues, and XK(3,3) are the eigenvectors of the diagonalization of the moment of inertia matrix (1/2E corresponds to the rotational constant, and XK are stored in columns). XMU(3,3) is not used. It may be necessary, as in ROTATE above, to select a principal axis geometry which reflects the symmetry in which the user wishes to work. The geometry produced by 'Multimode' may not be suitable for his purposes, in which case, the user must rotate to a geometry which reflects the symmetry of his choice. He will then use this geometry to define the symmetrization of the Normal Coordinate vectors in ROTATE above. Subroutine RTGEOM contains code for the rotation of the principal axis geometry, and this can be modified for any system that will be encountered by 'Multimode'.

The code `multimode' contains several machine-specific calls to timer routines. Currently computers supported are CRAY, IBM, and SGI. The user must comment out or delete timing routines to machines not being used. Also, there are calls to 'FLUSH' that are currently commented out. This routine flushes the output buffer.

Note...`multimode' calls timer routines for CRAY, IBM, SGI (these are also available from us).

C. react.XXX.f

Depending upon the setting of IREACT > 0 (analytic) or IREACT < 0 (ab initio), the user can elect to perform Reaction Path analyses in three distinct ways. Two of these assume that an analytical potential function exists in internal coordinates (IWHICH > 0), and IREACT > 0 points to the Reaction Path mode amongst the NMODE (3N-6) modes of the molecule. This ordering MUST be associated with the chosen starting value of the torsional angle that defines the Reaction Path. Two modules are supplied with `multimode.4.9.0' for the study of the Reaction Path of hydrogen peroxide. One (`react.vscf.eckart.f') minimizes the energy at each value of the torsional angle, in 1/2 degree intervals, and the Eckart conditions between successive structures along the path are obeyed (see J. Chem. Phys., 113. 987 (2000) and references therein). The Reaction Path commences at the equilibrium configuration aligned along principal axes, and the geometries are reflected about `cis' and `trans', thus making full use of symmetry. For studies of this type the logical variable ECKART in COMMON block /ECKCON/

```
     ECKART = .TRUE.
```

must be set within `react.vscf.eckart.f'.

A similar approach is carried out in `react.vscf.noneckart.f', but in this case each minimum-energy structure is rotated to principal axes only, and the Eckart conditions are not obeyed (strictly speaking there is no published reference for this approach, but see Molec. Phys., 101, 3513 (2003) for an outline of the theory). For studies of this type the logical variable

```
     ECKART = .FALSE.
```

must be set within `react.vscf.noneckart.f'.

Finally, the potential may be obtained ab initio at each point along the path, by pointing -IREACT to the Reaction Path mode (IREACT < 0). This method will then entail finding the minimum-energy structure ab initio, together with some form of potential in normal coordinates. In `react.vscf.abinitio.f' this takes the form of Cartesian first and second derivatives at the minimum-energy geometries which are used to determine a harmonic normal coordinate potential for methanol (see Molec. Phys., 101, 3513 (2003) and references therein). Anharmonicity can be added if required (see Phys. Chem. Chem. Phys., 3, 1958 (2001) and Spectrochimica Acta, A59, 1881 (2003) and references therein). For studies of this type, the logical variable

```
     ECKART = .FALSE.
```

has been set within `react.vscf.abinitio.f', although the user is free to use the Eckart conditions if required. In this case, the logical variable

```
     ECKART = .TRUE.
```

must be set within `react.vscf.abinitio.f'. For all `react.XXX.f' modules, the following calling sequence/interface must be satisfied:

```
      SUBROUTINE REACT(NATOM,NMODE,XM,X0,OMEGA,XL,XLP,XX,XXP,RR,XR,XP,
     1TEMP,BB,BBS,B,BS,BSS,IREACT)

      LOGICAL ECKART

      DIMENSION XM(NATOM),X0(NATOM,3),XL(NATOM,NMODE,3,722)
      DIMENSION XLP(NATOM,NMODE,3,722)
      DIMENSION XR(NATOM,3),OMEGA(NMODE,722),RR(NATOM,NATOM)
      DIMENSION XX(NATOM,3,722),XXP(NATOM,3,722)
      DIMENSION XP(3*NATOM,3*NATOM,722),TEMP(3*NATOM,3*NATOM)
      DIMENSION BB(NMODE,NMODE,3,722),BBS(NMODE,NMODE,722)
      DIMENSION B(NMODE,3,3,722),BS(NMODE,3,722),BSS(NMODE,722)

      COMMON/SCOORD/DSTAU(722),SPLUS(722),SMINUS(722)
      COMMON/ECKCON/ECKART
      COMMON/REACTN/IDUMMY,MMTAU,INIT,INCTAU
```

To use the supplied projection operator, the following are also required:

```
     DIMENSION SUP1(3,3),SUP2(3,3),SUP3(10),SUP4(4,4),SUP5(4,4)
     DIMENSION SUP6(20)
     COMMON/MOMI/XK(3,3),XMU(3,3)
```

and for ab initio potentials, the following variable is required in `react.vscf.abinitio.f':

```
     DIMENSION PAROT(3,3,722)
```

The projection operator is set up in `react.XXX.f' and the routine PROJEC is then called with the sequence:

```
     CALL PROJEC(NATOM,NMODE,XM,XX(1,1,ITAU),OMEGA(1,ITAU),
    1XL(1,1,1,ITAU),XR,RR,XP(1,1,ITAU),TEMP,PAROT(1,1,ITAU),ITAU)
```

where ITAU is the point numbering along the path. Definitions of the variables are as follows (assuming IMPLICIT DOUBLE PRECISION (A-H,O-Z)):

NATOM is the number of atoms.
NMODE is the total number of modes (3N-6).
IREACT contains the mode number corresponding to the path coordinate.
XM(NATOM) contains the atomic masses.
X0(NATOM,3) contains the reference geometry, input on `fort.1', but may be overwritten with a more convenient geometry along the path. It is used to define the one-dimensional modals.
XL(NATOM,NMODE,3,722) will contain the projected normal coordinate vectors at points along the path (minimum spacing 1/2 degree). XL(NATOM,NMODE,3,1) will contain the `reference' vectors corresponding to X0(NATOM,3).
XLP(NATOM,NMODE,3,733) will contain derivatives of the projected normal coordinate vectors with respect to the path coordinate (arc length for the true path coordinate in `react.vscf.eckart.f', or the torsional angle `tau' in the non-Eckart routines `react.vscf.noneckart.f' and `react.vscf.abinitio.f').
XR(NATOM,3) is a work array to hold a temporary path geometry.
OMEGA(NMODE,722) will hold the eigenfunctions of the projected force constant matrix at points along the path. OMEGA(NMODE,1) will contain those corresponding to the `reference' vectors in XL(NATOM,NMODE,3,1).
RR(NATOM,NATOM) is provided to contain bond distances, if required.
XX(NATOM,3,722) will contain the path Cartesian coordinates, either satisfying the Eckart conditions with the previous point as in `react.vscf.eckart.f', or
aligned along principal axes, as in `react.vscf.noneckart.f' and `react.vscf.abinitio.f'.
XXP(NATOM,3,722) will contain the (normalised) derivatives of the path geometries with respect to the path coordinate (arc length for the true path coordinate in `react.vscf.eckart.f'), or the (unnormalized) derivatives of the path geometries with respect to the path coordinate (the torsional angle `tau' in the non-eckart routines `react.vscf.noneckart.f' and

`react.vscf.abinitio.f').
XP(3*NATOM,3*NATOM,722) will contain the projection operator at points along the path.
TEMP(3*NATOM,3*NATOM) is for temporary storage of the second derivative matrix.
PAROT(3,3,722) holds the 3*3 rotation matrix in `react.vscf.abinitio.f' which transforms the geometry at each point along the path to principal axes. SUP1 to SUP6 are work arrays required for the projection algorithm.

BB(NMODE,NMODE,3,722),BBS(NMODE,NMODE,722),B(NMODE,3,3,722),
BS(NMODE,3,722),BSS(NMODE,722) are all black box arrays, and MUST be declared. They are used to store the final Coriolis terms.

The variables in COMMON block /SCOORD/ must also be set (see subroutines TORSPT and MILLER in `curve.vscf.4.9.0.f').
SPLUS(722) holds the difference in path coordinate (arc length or degrees) between point ITAU and ITAU+1 for all points ITAU along the path.
SMINUS(722) holds the difference in path coordinate (arc length or degrees) between point ITAU and ITAU-1 for all points ITAU along the path.
DSTAU(722) holds the Jacobian d(s)/d(tau), where `s' represents the path coordinate (arc length or angle) and `tau' is the torsional integration angle in radians.

The variables XK(3,3) and XMU(3,3) in COMMON block /MOMI/ are used to hold moment-of-inertia data in the projection algorithm.

The variable INIT in COMMON block /REACTN/ must be set to the point number ITAU corresponding to `cis' or `trans'; the point ITAU=1 is reserved for the `reference' point, and in the non-Eckart routines, point ITAU=2 is one of `cis' or `trans'. Strictly speaking the terms `cis' and `trans' refer to the value of the dihedral angle (0 or 180 degrees) which defines the torsional motion (cf the dihedral angle H1-C-O-H in methanol).

For Reaction Path studies for which a global potential exists (IREACT>0), the supplied routines `react.vscf.eckart.f' and `react.vscf.noneckart.f' follow basically the same approach. A torsional angle `tau' ranging from 0 to 180 degrees (corresponding to ITAU=2 to ITAU=362) is successively frozen in increments of 1/2 degree, and the routine MINPOT (above) is called to determine a `nice' geometry corresponding to the minimum-energy for eact `tau' in turn. This geometry is then translated to the centre-of-mass, and rotated, either to satisfy the Eckart conditions (in `react.vscf.eckart.f') or to the principal axis system (in `react.vscf.noneckart.f').

For ab initio potentials (IREACT<0), the following files must be set up:

`geom.dat' holds the cartesian coordinates at the (minimised) points along the path, stored as rows x,y,x for the NATOM columns. In the supplied routine `react.vscf.abinitio.f', the starting point is also the `reference' point, and so points ITAU=1 and ITAU=2 are equivalent. The first two structures in `geom.dat' are therefore identical, and the remaining structures are at 1/2 degree intervals. The first point corresponds to a dihedral H1-C-O-H angle of 180 degrees.

`deriv.dat' holds the first derivatives of the potential, stored as rows dV/dx, dV/dy, dV/dz for the NATOM columns. In the supplied routine `react.vscf.abinitio.f', they are stored for points ITAU=2 onwards, and correspond to the equivalent cartesian geometries in `geom.dat'.

`fcm.dat' holds the hessian (second derivative) matrix of the potential at all points ITAU=1 onwards, corresponding to the equivalent cartesian geometries in `geom.dat'. In the supplied routine `react.vscf.abinitio.f', point ITAU=1 is used to define the ordering of the normal coordinates which must be maintained at all subsequent points ITAU=2 onwards.

`v.dat' holds the (minimised) energies at all points ITAU=1 onwards.
`ve.dat' holds the single (equilibrium) potential.
`we.dat' holds the 3N-6 values of `omega' in cm-1 at equilibrium.

Regardless of the method chosen to define the path, extreme care must be taken in order to ensure that the path points and their derivatives, the projected 3N-7 normal coordinate vectors and their derivatives, all vary smoothly from start to finish (0 to 360 degrees). In the supplied routines, this is done by assuming that these quantities will only vary slightly in going from point ITAU to point ITAU+1 (1/2 degree). Therefore, provided that each quantity is itself normalised, the dot product of quantities at successive points will be almost unity, provided the path is smooth. If the dot product of successive points is far from unity, then `corrective action' must be performed. For the geometries at point ITAU+1, the most likely thing to have occurred is that (for `abinitio' or 'noneckart' algorithms) there has been an over-rotation in transforming to principal axes; if this over-rotation is 180 degrees, this can simply be corrected by a simple phase change. For over-rotations of 90 degrees or 270 degrees, axis-switching corrections must take place. Both of these corrections are present in the supplied routines. Once the geometries have been made to follow a continuous path, the derivatives should themselves follow a continuous path. The 3N-7 projected normal coordinate vectors should also follow a smooth path as they depend on the geometries and derivatives at point ITAU+1; however this may not always be the case. The trivial reason for this is that the diagonalisation within the Wilson FG has produced a

phase-change which must be corrected. There is a more serious reason, however; the ordering of the vectors depends on that obtained at the starting point (the 3N-6 vectors at the unique point ITAU=1). Point ITAU=2 corresponds to the first set of 3N-7 projected vectors, and the corresponding structure is either identical to that at ITAU=1 (for `abinitio') or very nearly identical to it (for `eckart' or noneckart'). The ordering of the corresponding 3N-7 vectors will therefore be the same. For points ITAU=3 onwards however, certain of these vectors may interchange. Taking the dot product of these (normalised) vectors will usually give an answer close to unity, unless two vectors have interchanged. The vectors (and omegas) at point ITAU+1 must therefore be switched. In extreme cases when there are several vectors corresponding to levels of similar energies (omega), a switched level may switch yet again. In this case the corrective action is to switch them back in two steps. All of this strategy is present in the supplied routines.