I. Programmer's Manual

II. Programmer's Manual

A. Structure and Philosophy of Basic Modules

B. Files Used by the Programs

ACES II makes use of a small number of unformatted files which contain various pieces of information relating to the calculation. For the most part, these files are direct access files which have an implicit logical record structure. This means that the programs which read from and/or write to these files treat the information in chunks (logical records) which do not correspond directly to the physical structure of the file on the disk drive. Translation between the logical and physical structure of the files is handled by specialized I/O routines (GETLST, PUTLST, GETREC and PUTREC). The following section lists all of the permanent files used by ACES II.

JOBARC : This file contains a considerable amount of information about the calculation, such as basis function and symmetry information, SCF eigenvectors, density matrices *etc.* All records resident on this file are associated with a character string which is used to reference them (see the description of the PUTREC and GETREC subroutines in the library routines handbook). Thus far, the programmers of ACES II have followed a philosophy which limits the size of records written to JOBARC to be of relatively small size (no larger than the square of the number of basis functions), with the result that this file is rarely larger than a few megabytes in size. JOBARC is a direct access file with an imposed logical record structure, with information regarding the translation held in the JOBARC common block and stored in the JASMRY file. Entries are read by the GETREC subroutine and written by the PUTREC subroutine. Subroutine JASMRY dumps a list of the contents of the file.

JASMRY : See JOBARC, above. This file contains a single record which is read and written by the DUMPJA subroutine.

OPTARC : This file contains information about the progress of optimizations and is a historical carry-over from the old days. Eventually, this file will probably be phased out with its information being moved over to JOBARC. This file has a sequential structure with relatively large records, and the number of records always equals the number of optimization cycles plus one. OPTARC records are written by subroutine ARCHIVE in joda.f and are read by subroutine RETRIEVE in the same program.

JODADONE : This file contains a single empty sequential record and is written by XJODA when an optimization has terminated. Its presence is used by XACES2 to test for the convergence of an optimization. It is written by subroutine EFOL of joda.f.

FCMINT : This file contains the full square internal coordinate force constant matrix and can be used to initialize the Hessian in geometry optimizations.

MOINTS, GAMLAM, MOINTS, SECDER, DERINT : These five files are actually treated as a single file by the program system, and contain the integral and amplitude lists used by the post-SCF program modules. As such, these files can become quite large and will always take up most of the disk space in actual calculations. The only difference between these files is the lists which are stored on them. MOINTS contains list numbers 1–100; GAMLAM

contains lists 101–200; MOABCD contains lists 201–300; SECDER contains lists 301–400; and DERINT contains lists 401–500. These files are composed of direct access records with an imposed logical record structure, and all I/O is supervised by the GETLST and PUTLST subroutines and their dependent routines FETCH and DUMP.

C. Job archive (JOBARC) file contents

This JOBARC file serves as a repository of many diverse data about the calculation being run. Records are read and written by nearly every module in ACES II. This section is intended as a catalog of the records which *may* be found on the JOBARC file, depending on the job type, etc. With each record is given a brief description, the size of the record, and which module first write it. Cross references are often given to related records.

Terms of reference

The "computational basis" or "computational order" is the order of centers and basis functions used by the integral package, and thus the rest of the calculation.

The "ZMAT basis" or "ZMAT order" refers to centers arranged in the order of the user's ZMAT file. Basis functions are rearranged to correspond to the ZMAT center ordering and may be further rearranged within a center to facilitate symmetry checking operations.

For transformation matrices, we try to indicate which bases label the rows and columns of the matrix in order to avoid confusion in their application. For non-unitary transformations, we also specify which side the transformation acts on.

SOs are symmetry-adapted orbitals, the basis in which the bulk of ACES II operates. AOs are atomic orbitals which are used mainly for symmetry analysis within the codes.

The sizes of records are described symbolically using the following definitions:

NAO	Number of atomic orbitals
NMO	Number of molecular orbitals
NAtoms	Number of centers specified in the ZMAT
NIrreps	Number of irreps in computational point group
NOrbits	Number of "orbits" (symmetry-unique centers)
NSO	Numer of symmetry-adapted orbitals

The actual variables employed in the code may be different.

C..1 ANGMOMBF

l-quantum number of each AO basis function in the computational basis. s=0, p=1, ... See also ANMOMBF0.

Data type: integer. Dimension: NAO. Written by: xvmol2ja.

C..2 ANMOMBF0

l-quantum number of each AO basis function in the ZMAT order. s=0, p=1, ... See also ANGMOMBF.

Data type: integer. Dimension: NAO. Written by: xvmol2ja.

C..3 AO2SO

The transformation matrix connecting the symmetry-adapted orbitals (columns of AO2SO) with the atomic orbitals (rows of AO2SO). This matrix multiplied on the *left* of an SO basis object will transform it to the AO basis (note that the record is misnamed in this respect). This matrix is in general *not* unitary and frequently not even square. The inverse transformation is given in the AO2SOINV record.

Data type: floating point. Dimension: NAO×NSO. Written by: xvmol2ja.

C..4 AO2SOINV

The transformation matrix connecting the atomic orbitals (columns of AO2SOINV) with the symmetry-adapted orbitals (rows of AO2SOINV). This matrix multiplied on the *left* of an AO basis object will transform it to the SO basis. This matrix is in general *not* unitary and frequently not even square. The inverse transformation is given in the AO2SOINV record.

Data type: floating point. Dimension: NSO×NAO. Written by: xvmol2ja.

C..5 ATOMCHRG

Atomic number of all atoms specified in Z-matrix. Dummy atoms have a value of zero.

Data type: floating point. Dimension: NATOMS. Written by: xjoda.

C..6 ATOMMASS

Atomic masses of all atoms specified in Z-matrix. Dummy atoms have zero mass.

Data type: floating point. Dimension: NATOMS. Written by: xjoda.

C..7 BRUKTEST

A flag which tells the xaces ii drive program if a Brueckner orbital-based calculation has converged to the tolerance specified by the BRUCK_CONV keyword. A value of 1 indicates convergence, a value of 0 forces another iteration.

Data type: integer. Dimension: 1. Written by: xvcc.

C..8 CENTERBF

Atomic center to which each AO basis function belongs. Both centers and basis functions are in the computational order. See also CNTERBF0.

Data type: integer. Dimension: NAO. Written by: xvmol2ja.

C..9 CMP2ZMAT

The transformation matrix connecting the computational ordered SO basis (columns of CMP2ZMAT) to the ZMAT ordered AO basis (rows of CMP2ZMAT). This matrix operates on the left. This matrix is in general *not* unitary and frequently not even square. The inverse transformation is given in the ZMAT2CMP record.

Data type: floating point. Dimension: NAO×NSO. Written by: xvmol2ja.

C..10 CNTERBF0

Atomic center to which each AO basis function belongs. Both centers and basis functions are in the ZMAT order. See also CENTERBF.

Data type: integer. Dimension: NAO. Written by: xvmol2ja.

C..11 COORD

Cartesian coordinates for all atoms in Z-matrix, according to the computational orientation.

Data type: floating point. Dimension: NATOMS*3. Written by: xjoda

C..12 FULLAOSO

The transformation matrix connecting the atomic orbitals (columns of FULLAOSO) with the symmetry-adapted orbitals (rows of FULLAOSO). This matrix multiplied on the *left* of an AO basis object will transform it to the SO basis. This matrix is in general *not* unitary. This matrix differs from AO2SOINV in that it is always a full square – it includes in the SO basis, functions such as spherical harmonic contaminants, which are deleted in the computational basis. The inverse transformation is given in the FULLSOAO record.

Data type: floating point. Dimension: NAO×NAO. Written by: xvmol2ja.

C..13 FULLSOAO

The transformation matrix connecting the symmetry-adapted orbitals (columns of FULLSOAO) with the atomic orbitals (rows of FULLSOAO). This matrix multiplied on the *left* of an SO basis object will transform it to the AO basis. This matrix is in general *not* unitary. This matrix differs from AO2SO in that it is always a full square – it includes in the SO basis, functions such as spherical harmonic contaminants, which are deleted in the computational basis. The inverse transformation is given in the FULLAOSO record.

Data type: floating point. Dimension: NAO×NAO. Written by: xvmol2ja.

C..14 GRADIENT

This record contains the Cartesian gradient. The entries are ordered $\partial E/\partial x$, $\partial E/\partial y$, $\partial E/\partial z$, atom-by-atom. The atoms follow the computational order.

Data type: integer. Dimension: 3*NATOMS. Written by: xvdint.

C..15 HESSIANM

This record contains the Hessian matrix. The ordering is $F_{1x1x}, F_{1x1y}, F_{1x1z}, F_{1x2x} \cdots F_{nx1x}, F_{nx1y} \cdots$. Atoms follow the computational order. The record is written by xvdint and xcphf.

Data type: integer. Dimension: 9*NATOMS*NATOMS. Written by: see above.

C..16 IFLAGS

The job control parameters which can be specified by the ZMAT file. This record is initially written by xjoda, but may be modified by subsequent programs in the execution sequence.

Data type: integer. Dimension: 100. Written by: see above.

C..17 LINEAR

A value of 1 indicates that the molecule is linear, 0 otherwise.

Data type: integer. Dimension: 1. Written by: xjoda.

C..18 MAP2ZMAT

Maps atomic centers from the computational ordering to the ZMAT ordering. If i is a computational center, MAP2ZMAT(i) is the number of the same center in the ZMAT file.

Data type: integer. Dimension: NAtoms. Written by: xvmol2ja.

C..19 MODROPA

The alpha molecular spin-orbitals which will be dropped prior to the electron correlation calculation. The orbitals are indexed by eigenvalue, with the smallest eigenvalue (most strongly bound) being 1 and the largest eigenvalue equal to the number of basis functions.

Data type: integer. Dimension: NUMDROPA (see below). Written by: xjoda.

C..20 MODROPB

The beta molecular spin-orbitals which will be dropped prior to the electron correlation calculation. While this record is present on JOBARC, it is never read, as the program system does not allow different values of MODROPA and MODROPB.

Data type: integer. Dimension: NUMDROPB. Written by: xjoda.

C..21 NAOBASFN

Number of atomic orbitals in the basis. Because the SO basis may have functions deleted from it (i.e. spherical harmonic contaminants), the AO basis may be larger than the SO basis. See also NBASTOT.

Data type: integer. Dimension: 1. Written by: xvmol2ja.

C..22 NAOBFORB

Number of atomic orbitals in each "orbit" (symmetry-unique center). This is the AO basis analog of NBASATOM.

Data type: integer. Dimension: NOrbits. Written by: xvmol2ja.

C..23 NBASATOM

Number of symmetry-adapted orbitals in each "orbit" (symmetry-unique center). This is the SO basis analog of NAOBFORB.

Data type: integer. Dimension: NOrbits. Written by: xvmol2ja.

C..24 NATOMS

The number of atoms in the Z-matrix (includes dummy atoms).

Data type: integer. Dimension: 1. Written by: xjoda.

C..25 NREALATM

The number of non-dummy centers in the Z-matrix.

Data type: integer. Dimension: 1. Written by: xjoda.

C..26 NUCREP

The nuclear repulsion energy in atomic units.

Data type: floating point. Dimension: 1. Written by: xjoda.

C..27 NBASTOT

Number of symmetry-adapted orbitals in the basis. Because the SO basis may have functions deleted from it (i.e. spherical harmonic contaminants), the AO basis may be larger than the SO basis. See also NAOBASFN.

Data type: integer. Dimension: 1. Written by: xvmol2ja.

C..28 NMPROTON

The number of protons in the molecule.

Data type: integer. Dimension: 1. Written by: xjoda.

C..29 NOCCORB

The number of occupied molecular orbitals used in the post-SCF calculation according to spin.

Data type: integer. Dimension: 2. Written by: xintprc.

C..30 NUCREP

Nuclear repulsion energy in Hartrees (atomic units).

Data type: floating point. Dimension: 1. Written by: xvmol2ja.

C..31 NUMBASIR

Number of SOs in each irrep.

Data type: integer. Dimension: NIrreps. Written by: xvmol2ja.

C..32 NUMDROPA

Number of alpha molecular spin-orbitals which will be dropped prior to the electron correlation calculation.

Data type: integer. Dimension: 1. Written by: xjoda.

C..33 NUMDROPB

Number of beta molecular spin-orbitals which will be dropped prior to the electron correlation calculation. While this record is present on JOBARC, it is never read, as the program system does not allow different values of NUMDROPA and NUMDROPB.

Data type: integer. Dimension: 1. Written by: xjoda.

C..34 NVRTORB

The number of virtual molecular orbitals used in the post-SCF calculation according to spin.

Data type: integer. Dimension: 2. Written by: xintprc.

C..35 OCCUPYA

Orbital occupancy vector for alpha spin-orbitals. Originally written by xjoda as specified by ZMAT input.

Data type: integer. Dimension: NIrreps. Written by: xjoda.

C..36 OCCUPYB

Orbital occupancy vector for beta spin-orbitals. Originally written by xjoda as specified by ZMAT input.

Data type: integer. Dimension: NIrreps. Written by: xjoda.

C..37 ORIENTMT

Matrix which relates computational and canonical orientations.

Data type: floating point. Dimension: 9. Written by: xjoda.

C..38 QRHFIRR

The symmetries of the orbitals which are involved in the QRHF orbital occupation alteration. A minus sign indicates that a β orbital of the associated symmetry will be depopulated, while a positive sign indicates that an α orbital will be populated.

Data type: floating point. Dimension: QRHFTOT. Written by: xjoda.

C..39 QRHFLOC

The offset for orbitals which are involved in the QRHF orbital occupation alteration. For depopulation calculations, a value of 1 indicates the occupied orbital with the largest eigenvalue in the symmetry block; larger values give offsets with respect to the Fermi level. For electron addition, 1 refers to the lowest unoccupied orbital within the symmetry block, and larger values give offsets as above.

Data type: floating point. Dimension: QRHFTOT. Written by: xjoda.

C..40 QRHFTOT

The number of orbitals which will undergo a change in occupation number in the QRHF procedure.

Data type: floating point. Dimension: 1. Written by: xjoda.

C..41 S2SCF

The value of $\langle S^2 \rangle$ for the reference wavefunction.

Data type: floating point. Dimension: 1. Written by: xvscf.

C..42 SCFENEG

The SCF energy.

Data type: floating point. Dimension: 1. Written by: xvscf.

C..43 SCFEVALA

The diagonal elements of the α Fock matrix, expressed in the molecular orbital basis. Ordered according to the post-SCF orbital numbering scheme.

Data type: floating point. Dimension: NMO. Written by: xftran*.

C..44 SCFEVALB

The diagonal elements of the β Fock matrix, expressed in the molecular orbital basis. Ordered according to the post-SCF orbital numbering scheme.

Data type: floating point. Dimension: NMO. Written by: xftran*.

C..45 SCFEVCA0

The α molecular orbitals, ordered according to the SCF orbital numbering scheme.

Data type: floating point. Dimension: NAO*NAO. Written by: xvscf.

C..46 SCFEVCB0

The β molecular orbitals, ordered according to the SCF orbital numbering scheme.

Data type: floating point. Dimension: NAO*NAO. Written by: xvscf.

C..47 SCFEVECA

The α molecular orbitals, ordered according to the post-SCF orbital numbering scheme, as used in post-SCF calculations.

Data type: floating point. Dimension: NMO*NAO. Written by: xftran*.

C..48 SCFEVECB

The β molecular orbitals, ordered according to the post-SCF orbital numbering scheme, as used in post-SCF calculations.

Data type: floating point. Dimension: NMO*NAO. Written by: xftran*.

C..49 SCFKICK

A flag which tells the xaces ii driver program if an SCF orbital instability has been found which can be followed within the current symmetry and spin restrictions. A value of 1 forces reexecution of the SCF code with a rotated eigenvector, a value of 0 terminates the job. Used only in jobs which invoke HFSTABILITY=FOLLOW.

Data type: integer. Dimension: 1. Written by: xintprc.

C..50 TOTENERG

The total energy. This record is written by many programs, beginning with xvscf and keeps a running tab on the total energy.

Data type: floating point. Dimension: 1. Written by: see above.

C..51 UHFRHF

This flag gives the type of SCF calculation which will be performed. 0 indicates RHF, 1 indicates UHF and 2 indicates ROHF. UHFRHF is originally written by xjoda, but subsequent programs may modify it.

Data type: floating point. Dimension: 1. Written by: see above.

C..52 ZMAT2CMP

The transformation matrix connecting the ZMAT ordered AO basis (columns of ZMAT2CMP) to the computationally ordered SO basis (rows of ZMAT2CMP). This matrix operates on the left. This matrix is in general *not* unitary and frequently not even square. The inverse transformation is given in the CMP2ZMAT record.

Data type: floating point. Dimension: NSO×NAO. Written by: xvmol2ja.

D. Integral Lists

The following section describes the contents of the MOINTS, GAMLAM, MOABCD, SECDER and DERINT files, which are used to store two-electron integrals and various other quantities.

```
PART I.  Lists used in standard calculation types (single point energy,
first and second derivative calculations)

-------------------------------------------------------------
   J      I           Quantity           Storage Mode
                                          ALL ; FOR
                                               EACH
-------------------------------------------------------------
 1-NIRREP  5          <AB|IJ>             A,I ; B,J (NOT ANTISYMMETRIZED)
 1-NIRREP  6          <ab|ij>             a,i ; b,j (NOT ANTISYMMETRIZED)

 1-NIRREP 11          <IJ||KL>            I<J ; K<L                    ***
 1-NIRREP 12          <ij||kl>            i<j ; k<l                    ***
 1-NIRREP 13          <Ij||Kl>            I,j ; K,l
```

```
1-NIRREP  7          <IJ||KA>            I<J ; K,A
1-NIRREP  8          <ij||ka>            i<j ; k,a                    ***
1-NIRREP  9          <Ij|Ak>             I,j ; A,k                    ***
1-NIRREP 10          <Ij|Ka>             I,j ; K,a

1-NIRREP 14          <AB||IJ>            A<B ; I<J
1-NIRREP 15          <ab||ij>            a<b ; i<j                    ***
1-NIRREP 16          <Ab|Ij>             A,b ; I,j
1-NIRREP 17          <Ab|Ij>             b,j ; A,I                    ***
1-NIRREP 18          <Ab|Ij>             A,I ; b,j
1-NIRREP 19          <AB||IJ>            A,I ; B,J
1-NIRREP 20          <ab||ij>            a,i ; b,j                    ***
1-NIRREP 21          <Ab|Ij>             A,j ; b,I
1-NIRREP 22          <Ab|Ij>             b,I ; A,j                    ***

1-NIRREP 23          <IA||JB>            B,I ; A,J
1-NIRREP 24          <ia||jb>            b,i ; a,j                    ***
1-NIRREP 25          <iA|jB>             B,i ; A,j
1-NIRREP 26          <aI|bJ>             b,I ; a,J                    ***

1-NIRREP 27          <AB||CI>            A<B ; C,I
1-NIRREP 28          <ab||ci>            a<b ; c,i                    ***
1-NIRREP 29          <Ab|Ic>             A,b ; I,c                    ***
1-NIRREP 30          <Ab|Ci>             A,b ; C,i

1-NIRREP 34        T2(IJ,AB)             A,J ; B,I
1-NIRREP 35        T2(ij,ab)             a,j ; b,i                    ***
1-NIRREP 36        T2(Ij,Ab)             b,j ; A,I                    ***
1-NIRREP 37        T2(Ij,Ab)             A,I ; b,j
1-NIRREP 38        T2(Ij,Ab)             b,I ; A,j                    ***
1-NIRREP 39        T2(Ij,Ab)             A,j ; b,I

1-NIRREP 40        T2(IJ,AB) (increment)A,J ; B,I      (rings)        ***
1-NIRREP 41        T2(ij,ab) (increment)a,j ; b,i      (rings)        ***
1-NIRREP 42        T2(Ij,Ab) (increment)A,I ; b,j      (rings)
1-NIRREP 43        T2(Ij,Ab) (increment)A,j ; b,I      (rings)

1-NIRREP 44        T2(IJ,AB)             A<B ; I<J
1-NIRREP 45        T2(ij,ab)             a<b ; i<j                    ***
1-NIRREP 46        T2(Ij,Ab)             A,b ; I,j

1-NIRREP 48        D(IJ,AB)              A<B ; I<J
```

```
1-NIRREP 49         D(ij,ab)             a<b ; i<j                      ***
1-NIRREP 50         D(Ij,Ab)             A,b ; I,j

1-NIRREP 51         W(MN,IJ) intermed.   M<N ; I<J                      ***
1-NIRREP 52         W(mn,ij) intermed.   m<n ; i<j                      ***
1-NIRREP 53         W(Mn,Ij) intermed.   M,n ; I,j
1-NIRREP 54         W(MB,EJ) intermed.   E,M ; B,J
1-NIRREP 55         W(mb,ej) intermed.   e,m ; b,j                      ***
1-NIRREP 56         W(Mb,Ej) intermed.   E,M ; b,j
1-NIRREP 57         W(mB,eJ) intermed.   e,m ; B,J                      ***
1-NIRREP 58         W(mB,Ej) intermed.   E,m ; B,j
1-NIRREP 59         W(Mb,eJ) intermed.   e,M ; b,J                      ***

1-NIRREP 61         T2(IJ,AB) increment  A<B ; I<J                      ***
                    L2(IJ,AB) increment  A<B ; I<J in Lambda iterations ***
                    T2(IJ,AB)[3]         A<B ; I<J  iterative         ROHF
1-NIRREP 62         T2(ij,ab) increment  a<b ; i<j                       ***
                    L2(ij,ab) increment  a<b ; i<j in Lambda iterations ***
                    T2(ij,ab)[3]         a<b ; i<j  iterative         ROHF
1-NIRREP 63         T2(Ij,Ab) increment  A,b ; I,j
                    L2(Ij,Ab) increment  A,b ; I,j in Lambda iterations
                    T2(Ij,Ab)[3]         A,b ; I,j  iterative         ROHF

1-NIRREP 64         Reciprocal D(IJ,AB)  A<B ; I<J
 9       64         Reciprocal D(I,A)      A,I
1-NIRREP 65         Reciprocal D(ij,ab)  a<b ; i<j                      ***
 9       65         Reciprocal D(i,a)      a,i                          ***
1-NIRREP 66         Reciprocal D(Ij,Ab)  A,b ; I,j

1-3      70-89      Reserved for RLE Jacobi iterates

 1       90         T1AA                   A,I
 2       90         T1BB                   a,i                          ***
 3       90         T1AA AND L1AA INCREM   A,I
 4       90         T1BB AND L1BB INCREM   a,i                          ***
 5       90         AB MO OVERLAP MATRIX   I,j  (o-o)                   ***
 6       90         AB MO OVERLAP MATRIX   A,j  (v-o)                   ***
 7       90         AB MO OVERLAP MATRIX   I,b  (o-v)                   ***
 8       90         AB MO OVERLAP MATRIX   A,b  (v-v)                   ***
 9       90         T1AA[2]; iterative     A,I                          ROHF
10       90         T1BB[2]; iterative     a,i                          ROHF
```

```
 1      91          F(MI) intermed.         M,I
 2      91          F(mi) intermed.         m,i                          ***
 3      91          f(M,I) (fock matrix)    M,I  (only non-HF methods)
 4      91          f(m,i) (fock matrix)    m,i  (only non-HF methods) ***

 1      92          F(EA) intermed.         E,A
 2      92          F(ea) intermed.         e,a                          ***
 3      92          f(E,A) (fock matrix)    E,A  (only non-HF methods)
 4      92          f(e,a) (fock matrix)    e,a  (only non-HF methods) ***

 1      93          F(AI) intermed.         A,I
 2      93          F(ai) intermed.         a,i                          ***
 3      93          f(A,I) (fock matrix)    A,I  (only non-HF methods)
 4      93          f(a,i) (fock matrix)    a,i  (only non-HF methods) ***

1-NIRREP 94         T2(IJ,AB)[2] iterative  A<B ; I<J                 ROHF
1-NIRREP 95         T2(ij,ab)[2] iterative  a<b ; i<j                 ROHF
1-NIRREP 96         T2(Ij,Ab)[2] iterative  A,b ; I,j                 ROHF

1-NIRREP 98         T2(Ij,Ab) cumulative    A,b ; I,j  (MBPT S^2)     ROHF

1-NIRREP 107        G(IJ,KA)              I<J ; K,A                    ***
                    W(IJ,KA)              I<J ; K,A in VLAMCC          ***
1-NIRREP 108        G(ij,ka)              i<j ; k,a                    ***
                    W(ij,ka)              i<j ; k,a in VLAMCC          ***
1-NIRREP 109        G(Ij,Ak)              I,j ; A,k                    ***
                    W(Ij,Ak)              I,j ; A,k in VLAMCC          ***
1-NIRREP 110        G(Ij,Ka)              I,j ; K,a
                    W(Ij,Ka)              I,j ; K,a in VLAMCC

1-NIRREP 111        G(IJ,KL)              I<J ; K<L                    ***
1-NIRREP 112        G(ij,kl)              i<j ; k<l                    ***
1-NIRREP 113        G(Ij,Kl)              I,j ; K,l
1-NIRREP 114        G(IJ,AB)              I<J ; A<B
1-NIRREP 115        G(ij,ab)              i<j ; a<b                    ***
1-NIRREP 116        G(Ij,Ab)              I,j ; A,b
1-NIRREP 117        G(iA,bJ)              b,j ; A,I                    ***
  resorted in VDENS to                    b,I ; A,j                    ***
1-NIRREP 118        G(Ia,Bj)              B,J ; a,i
  resorted in VDENS to                    B,i ; a,J
1-NIRREP 123        G(IA,JB)              B,I ; A,J
  resorted in VDENS to                    B,J ; A,I
```

```
 1-NIRREP 124        G(ia,jb)             b,i ; a,j                    ***
   resorted in VDENS to                   b,j ; a,i                    ***
 1-NIRREP 125        G(iA,jB)             B,i ; A,j
   resorted in VDENS to                   B,j ; A,i
 1-NIRREP 126        G(Ia,Jb)             b,I ; a,J                    ***
   resorted in VDENS to                   b,J ; a,I                    ***

 1-NIRREP 127        G(AB,CI)             A<B ; C,I                    ***
                     W(AB,CI)             A<B ; C,I in VLAMCC          ***
 1-NIRREP 128        G(ab,ci)             a<b ; c,i                    ***
                     W(ab,ci)             a<b ; c,i in VLAMCC          ***
 1-NIRREP 129        G(Ab,Ic)             A,b ; I,c                    ***
                     W(Ab,Ic)             A,b ; I,c in VLAMCC          ***
 1-NIRREP 130        G(Ab,Ci)             A,b ; C,i
                     W(Ab,Ci)             A,b ; C,i in VLAMCC

 1-NIRREP 131        G(AB,CD)             A<B ; C<D                    ***
 1-NIRREP 132        G(ab,cd)             a<b ; c<d                    ***
 1-NIRREP 133        G(Ab,Cd)             A,b ; C,d

 1-NIRREP 134        L2(IJ,AB)            A,J ; B,I
 1-NIRREP 135        L2(ij,ab)            a,j ; b,i                    ***
 1-NIRREP 136        L2(Ij,Ab)            b,j ; A,I                    ***
 1-NIRREP 137        L2(Ij,Ab)            A,I ; b,j
 1-NIRREP 138        L2(Ij,Ab)            b,I ; A,j                    ***
 1-NIRREP 139        L2(Ij,Ab)            A,j ; b,I

 1-NIRREP 144        L2(IJ,AB)            A<B ; I<J
 1-NIRREP 145        L2(ij,ab)            a<b ; i<j                    ***
 1-NIRREP 146        L2(Ij,Ab)            A,b ; I,j


Derivatives of one-electron integrals in AO basis:

 1-NIRREP 150        S(mu,nu)^chi         ipert ; mu >= nu
 1-NIRREP 151        f(mu,nu^(chi) (alpha)ipert ; mu >= nu
 1-NIRREP 152        f(mu,nu^(chi) (beta) ipert ; mu >= nu             ***
 1-3      153        dipole(mu,nu)        ixyz ; mu >= nu
 1-NIRREP 154        A^(chi)*DREL  (alpha)ipert ; mu >= nu
 1-NIRREP 155        A^(chi)*DREL  (beta) ipert ; mu >= nu             ***
```

```
Relaxed density matrix (correlation correction only):

 1        160        D(I,J)              occ.-occ. block of relaxed
                                         density matrix (alpha)
 2        160        D(i,j)              occ.-occ. block of relaxed
                                         density matrix (beta)        ***
 3        160        D(A,B)              virt.-virt. block of relaxed
                                         density matrix (alpha)
 4        160        D(a,b)              virt.-virt. block of relaxed
                                         density matrix (beta)        ***
 5        160        D(A,I)              occ.-virt. block of relaxed
                                         density matrix (alpha)
 6        160        D(a,i)              occ.-virt. block of relaxed
                                         density matrix (beta)        ***

Intermediate matrix I(p,q) (correlation correction only):

 1        161        I(I,J)              occ.-occ. block of intermediate
                                         matrix (alpha)
 2        161        I(i,j)              occ.-occ. block of intermediate
                                         matrix (beta)                ***
 3        161        I(A,B)              virt.-virt. block of intermediate
                                         matrix (alpha)
 4        161        I(a,b)              virt.-virt. block of intermediate
                                         matrix (beta)                ***
 5        161        I(A,I)              virt.-occ. block of intermediate
                                         matrix (alpha)
 6        161        I(a,i)              virt.-occ. block of intermediate
                                         matrix (beta)                ***

Gamme-intermediates created in the triple code for CCSD(T), ...

 1-NIRREP 162        G^trip(IJ,KA)        I<J ; K,A                   ***
 1-NIRREP 163        G^trip(ij,ka)        i<j ; k,a                   ***
 1-NIRREP 164        G^trip(Ij,Ak)        I,j ; A,k                   ***
 1-NIRREP 165        G^trip(Ij,Ka)        I,j ; K,a
 1-NIRREP 166        G^trip(AB,CI)        A<B ; C,I                   ***
 1-NIRREP 167        G^trip(ab,ci)        a<b ; c,i                   ***
 1-NIRREP 168        G^trip(Ab,Ic)        A,b ; I,c                   ***
 1-NIRREP 169        G^trip(Ab,Ci)        A,b ; C,i

Derivatives of Overlap and Fock matrices and CPHF coefficients in MO basis:
```

```
 1-NIRREP 170        S(I,J)^chi
 1-NIRREP 171        S(i,j)^chi                                          ***
 1-NIRREP 172        S(A,B)^chi
 1-NIRREP 173        S(a,b)^chi                                          ***
 1-NIRREP 174        S(A,I)^chi
 1-NIRREP 175        S(a,i)^chi                                          ***
 1-NIRREP 176        F(I,J)^chi
 1-NIRREP 177        F(i,j)^chi                                          ***
 1-NIRREP 178        F(A,B)^chi
 1-NIRREP 179        F(a,b)^chi                                          ***
 1-NIRREP 180        F(A,I)^chi
 1-NIRREP 181        F(a,i)^chi                                          ***
 1-NIRREP 182        U(A,I)^chi
 1-NIRREP 183        U(a,i)^chi                                          ***
 1-NIRREP 184        additional lists used in CPHF (RHF and UHF)
 1-NIRREP 185        additional lists used in CPHF (UHF only)            ***
 1-NIRREP 186        A^(chi)*D (I,J)
 1-NIRREP 187        A^(chi)*D (i,j)                                     ***
 1-NIRREP 188        A^(chi)*D (A,I)
 1-NIRREP 189        A^(chi)*D (a,i)                                     ***

  1     190          L1AA                    A,I
  2     190          L1BB                    a,i                         ***

  1     191          G(MI) intermed.         M,I
  2     191          G(mi) intermed.         m,i                         ***

  1     192          G(A,E) intermed.        E,A
  2     192          G(a,e) intermed.        e,a                         ***

 1-NIRREP 231          <AB||CD>              A<B ; C<D                   ***
                       W(AB,CD)              A<B ; C<D DURING VLAMCC     ***
 1-NIRREP 232          <ab||cd>              a<b ; c<d                   ***
                       W(ab,cd)              a<b ; c<d DURING VLAMCC     ***
 1-NIRREP 233          <Ab|Cd>               A,b ; C,d
                       <Ab|Cd>               A<=b; C,d single point RHF
                       W(Ab,Cd)              A,b ; C,d DURING VLAMCC

Some additional lists are used in MBPT(2) second derivative calculations.
These are:

 1-NIRREP 314       d<IJ||AB>/dx         A<B ; I<J,IPERT               ***
```

```
 1-NIRREP 315       d<ij||ab>/dx         a<b ; i<j,IPERT                ***
 1-NIRREP 316       d<Ij||Ab>/dx         A,b ; I,j,IPERT
```

In analytic gradient calculations, the direct access files are reinitialized and restructured by the modules XANTI and XBCKTRN, which process and transform the two-particle reduced density matrix. These programs use the MOINTS and GAMLAM files in the following way

a) Lists used by ANTI

left index	right index	quantity	symmetry type	storage	
1	1-NSYM	G(PQ,rs)	AAAA	P<Q ; r<s	
2	1-NSYM	G(PQ,rs)	AABB	P<Q ; r<s	
3	1-NSYM	G(PQ,rs)	ABAB	P,Q ; r,s	
4	1-NSYM	G(PQ,rs)	ABCD	P,Q ; r,s	
1	51-50+NSYM	G(pq,RS)	AAAA	p<q ; R<S	***
2	51-50+NSYM	G(pq,RS)	AABB	p<q ; R<S	***
3	51-50+NSYM	G(pq,RS)	ABAB	p,q ; R,S	***
4	51-50+NSYM	G(pq,RS)	ABCD	p,q ; R,S	***
6	1-NSYM	G(PQ,RS)	AAAA	P<Q ; R<S	***
7	1-NSYM	G(PQ,RS)	AABB	P<Q ; R<S	***
8	1-NSYM	G(PQ,RS)	ABAB	P,Q ; R,S	***
9	1-NSYM	G(PQ,RS)	ABCD	P,Q ; R,S	***
6	51-50+NSYM	G(pq,rs)	AAAA	p<q ; r<s	***
7	51-50+NSYM	G(pq,rs)	AABB	p<q ; r<s	***
8	51-50+NSYM	G(pq,rs)	ABAB	p,q ; r,s	***
9	51-50+NSYM	G(pq,rs)	ABCD	p,q ; r,s	***

b) Lists used in BCKTRN and ABACUS

1	1-NSYM	G(mu,nu,sigma,rho)	AAAA	mu < nu ; sigma < rho
2	1-NSYM	G(mu,nu,sigma,rho)	AABB	mu < nu ; sigma < rho
3	1-NSYM	G(mu,nu,sigma,rho)	ABAB	mu,nu ; sigma,rho
4	1-NSYM	G(mu,nu,sigma,rho)	ABCD	mu,nu ; sigma,rho

In the above, the number of possible sublists defined by the right index of the list depends upon the symmetry type. Obviously, there are NIRREP different sublists for symmetry type 1 : 1,1,1,1; ....; NIRREP,NIRREP,NIRREP,NIRREP. For the second symmetry type, there are NIRREP*(NIRREP-1)/2 different sublists, namely those for each possible pair of irreps with A < B (A faster running than B). We have thus 1,1,2,2; 1,1,3,3; 2,2,3,3; .... NIRREP-1,NIRREP-1,NIRREP, NIRREP. The symmetry type ABAB involves also NIRREP*(NIRREP-1)/2 sublists. They are however given in such a way that the main loop is over IRREP(AB), and the second, faster loop over B with A faster than B and IRREP(A) < IRREP(B). The fourth symmetry type, ABCD, is by far the most complicated. The main loop runs first over IRREP(CD)=IRREP(AB) with a subloop over IRREP(C) and IRREP(D) in such a way that C is faster than D and IRREP(C) < IRREP(D). For a given CD (note again IRREP(AB) = IRREP(CD)), the fastest loops runs over all possible combinations of IRREP(A) and IRREP(B) with A faster than B, IRREP(A) < IRREP(B) and also (IRREP(A), IRREP(B)) > IRREP(C),IRREP(D). The loop structure is set up for example in a very clear way in the routine GAMSRT which might be consulted for further questions regarding the set up of the loops and dealing with the AO-two-particle density matrix.

PART II. Extra lists used in excitation energy calculations

J	I	Quantity	Storage Mode ALL ; FOR EACH		
1-NIRREP	434	C(IJ,AB)	A,J ; B,I	[INCREMENT]	***
1-NIRREP	435	C(ij,ab)	a,j ; b,i	[INCREMENT]	***
1-NIRREP	436	C(Ij,Ab)	b,j ; A,I	[INCREMENT]	***
1-NIRREP	437	C(Ij,Ab)	A,I ; b,j	[INCREMENT]	
1-NIRREP	438	C(Ij,Ab)	b,I ; A,j	[INCREMENT]	***
1-NIRREP	439	C(Ij,Ab)	A,j ; b,I	[INCREMENT]	
1-NIRREP	444	C(AB,IJ)	A<B ; I<J		***
1-NIRREP	445	C(ab,ij)	a<b ; i<j		***
1-NIRREP	446	C(Ab,Ij)	A,b ; I,j		
1-NIRREP	448	D(IJ,AB)	A<B ; I<J		

```
1-NIRREP 449         D(ij,ab)           a<b ; i<j                        ***
1-NIRREP 450         D(Ij,Ab)           A,b ; I,j
1-NIRREP 461         C(AB,IJ)           A<B ; I<J  [INCREMENT]           ***
1-NIRREP 462         C(ab,ij)           a<b ; i<j  [INCREMENT]           ***
1-NIRREP 463         C(Ab,Ij)           A,b ; I,j  [INCREMENT]
1-NRECS  470      C vectors (one/record)
1-NRECS  471      HC vectors (one/record)
1        472      Converged right C vector (on one record)
2        472      Converged left C vector (on one record)
1        490         C1(AI)             AI  ;
2        490         C1(ai)             ai  ;
3        490         C1(AI)             AI  ;       [INCREMENT]
4        490         C1(ai)             ai  ;       [INCREMENT]          ***
1        491          X(IJ)             IJ  ;
2        491          X(ij)             ij  ;                            ***
1        492          X(AB)             AB  ;
2        492          X(ab)             ab  ;                            ***
1        493          X(AI)             AB  ;
2        493          X(ai)             ab  ;                            ***
1-NIRREP  94          TDA eigenvectors (one/record)
1-NIRREP  95          TDA eigenvalues (one/record)
```

E. Common Blocks

The following is a listing of all common blocks which are automatically initialized in the CRAPSI procedure, and which programs make use of these subroutines. One should not assume that any of these common blocks have been initialized unless the program uses them. Initialization of these common blocks depends on logic based on the third argument passed to CRAPSI, as well as the presence or absence of specific files on disk.

```
      COMMON // ICORE(1)
```

This is the blank common block which is used for purposes of memory allocation, and serves to provide an address for the first element of the ICORE array. The value of ICORE(1) is never used. Used in all programs except aces ii.f, joda.f, vmol.f and vprops.f.

```
      COMMON /CACHEINF/ CACHNUM,CACHNMP1,CACHDIR(100),CACHPOS(100),
     &                  CACHFILE(100),CACHMOD(100),OLDEST
```

This common block contains information about the contents of the software I/O cache. CACHNUM : The total number of records held in the cache; CACHNMP1 : CACHNUM+1; CACHDIR : The record and file held in each of the 100 positions [these values are bitpacked

as follows : Record number (Bits 1-IBITWD*4-3); File number (remaining bits)]; CACHPOS : The location in ICORE (*relative to ICORE(0) !!!* where the record is held; CACHFILE : The actual unit number of the file associated with the record (*i.e. not in general an integer between 1 and 5*; CACHMOD : A flag indicating whether the record has been modified during its stay in core memory, with a value of 1 indicating modification, 0 otherwise; OLDEST : The location of the oldest record in the cache, which will be the first one flushed to disk when a new record must be read from the physical disk. All variables are integers. Used in post-SCF programs only. (Ed. note : The author of this section must admit that he has learned a great deal in writing this entry, even though he also authored all of the code for the I/O cache!)

```
COMMON /FLAGS/ IFLAGS(100)
```

The IFLAGS array contains the specified and/or default values of the ACES2 keywords, and is set in subroutine GTFLGS of joda.f. All values are integers. Used in all program modules.

```
COMMON /INFO/ NOCCO(2),NVRTO(2)
```

The information contained in this common block is perhaps obvious. The NOCCO and NVRTO arrays give the number of occupied and virtual orbitals by spin, irrespective of the symmetry. All variables are again integers. symmetry. Used in post-SCF programs only.

```
COMMON /IOPOS/ ICRSIZ,ICHCSZ,IOFF(2),LENREC
```

This common block contains information about memory usage and details of the software cache maintained by the ACES II I/O subsystem. ICRSIZ : The total working core area (in integer words) after subsidiary arrays have been allocated (*this value is always less than that specified by the* MEMORY_SIZE *keyword*; ICHCSZ : The size of a cache record in integer words; IOFF(2) : Not used; LENREC : The length of a physical disk record. The values of ICHCSZ and LENREC are always the same, and both variables persist for purely historical reasons. Used in all programs except aces ii.f, joda.f, vmol.f and vprops.f.

```
COMMON /ISTART/ I0
```

This is the number of integer words *relative to ICORE(0)* where the program working area begins. Thus, the program workspace begins at ICORE(I0), which is the address which must be passed to dependent subroutines. In general, this is necessary only in the main program unit. Used in all programs except aces ii.f, joda.f, vmol.f and vprops.f.

```
      COMMON /JOBARC/ MARKER(LENGTH),LOC(LENGTH),SIZE(LENGTH),NRECS,
     &                IRECWD,IRECLN
```

This common block contains information about the JOBARC and JAINDX files. MARKER : The eight character string associated with the logical record; LOC : The absolute word address of the first element of the record on the JOBARC file; SIZE : The size of the logical record in integer words; NRECS : The total number of physical records in the JOBARC file; IRECWD : The total number of integer words per physical record; IRECLN : Apparently not used. LENGTH is declared in parameter statements in all routines which are related to JOBARC/JAINDX I/O (DUMPJA, GETREC, JASMRY, PUTREC and ZEROJA). All variables are integers, as usual. Used in all programs.

```
      COMMON /LISTS/ MOIO(10,500),MOIOWD(10,500),MOIOSZ(10,500),
     &               MOIODS(10,500),MOIOFL(10,500)
```

This common block contains all information needed to retrieve a list from disk. The five variables are indexed by the list number (right index) and the "sublist" value, which is usually (but not always!) the symmetry species (left index). The following information is stored : MOIO : The direct access physical record on which the list begins; MOIOWD : The total number of words which make up the list; MOIOSZ : The distribution size of the list [length of the left-hand two indices, which is equivalent to IRPDPD(IRREP,ISYTYP(1,LIST)), where IRREP is the left-hand index of MOIOSZ]; MOIODS : The number of distributions of the list [length of the right-hand two indices, which is equivalent to IRPDPD(IRREP,ISYTYP(2,LIST))]; MOIOFL : A value between 1 to 5 which refers to the specific file where the list resides [1=MOINTS (lists 1-100); 2=GAMLAM (lists 101-200); 3=MOABCD (lists 201- 300); 4=DERGAM (lists 301-400); 5=SECDER (lists 401-500)]. All values are integers. Used in post-SCF programs only.

```
      COMMON /MACHSP/ IINTLN,IFLTLN,IINTFP,IALONE,IBITWD
```

This common block contains machine-specific information which is related to the machine word size. The meaning of the variables is as follows : IINTLN : Length of an integer in bytes; IFLTLN : Length of a floating point number in bytes; IINTFP : The ratio IFLTLN/IINTLN ; IALONE : The integer value with all bits set to "1" (*i.e.* 255 for IINTLN=4 and 65535 for IINTLN=8); IBITWD : 8*IINTLN/4. The most commonly used variable in this common block is undoubtedly IINTFP, as it is extremely useful in the coding of memory allocation. All variables are integers. Used in all programs.

```
      COMMON /MACHSP2/ MASK1,MASK2,ISHFSZ
```

This common block contains information required for bit packing and bit unpacking of the CACHDIR variable used in common block /CACHINF/. MASK1 : 2**ISHFSZ-1, MASK2 : 7; ISHFSZ = 4*IBITWD-3. Used in post-SCF programs only.

```
      COMMON /SYM/ POP(8,2),VRT(8,2),NT(2),NFMI(2),NFEA(2)
```

This common block contains information about the distribution of molecular orbitals among the various symmetry species. POP : The number of occupied orbitals by symmetry (left index) and spin (right index); VRT : The number of virtual orbitals by symmetry (left index) and spin (right index); NT : The lengths of both $\alpha\alpha$ and $\beta\beta$ virtual-occupied vectors *which are overall totally symmetric*; NFMI : The lengths of both $\alpha\alpha$ and $\beta\beta$ occupied-occupied vectors *which are overall totally symmetric*; NFEA : The lengths of both $\alpha\alpha$ and $\beta\beta$ virtual-virtual vectors *which are overall totally symmetric*. All variables are integers. Used in post-SCF programs only.

```
COMMON /SYMINF/ NSTART,NIRREP,IRREPS(255,2),DIRPRD(8,8)
```

This common block contains some rudimentary information which pertains to molecular symmetry. The meaning of the variables is as follows : NSTART : Not used; NIRREP : The number of symmetry species in the computational point group; IRREPS(255,2) : Not used; DIRPRD(8,8) : The direct product table for the symmetry species of the computational point group. All variables are integers. Used in post-SCF programs only.

```
COMMON /SYMPOP/ IRPDPD(8,22),ISYTYP(2,500),ID(18)
```

This common block contains information which is extremely useful in the coding of contractions. The meaning of the variables is slightly complicated, but worth learning if any serious coding is being planned. The IRPDPD array contains information about the total population of all possible two index arrays, by symmetry block. The right-hand index of IRPDPD refers to the specific type of two-index quantity (such as ab, $a\bar{i}$, $\bar{i}\bar{j}$ *etc.*), while the left-hand index refers to the symmetry block. The 22 different binary combinations are listed in the subsection entitled "Symmetry Types" below. However, these need not be consulted that frequently, since the ISYTYP(2,500) array contains these values for all lists on disk. The left-hand index of ISYTYP refers to the "side" of the list (left-hand or right-hand set of two indices, as stored) and the right-hand index refers to the list number. For example, ISYTYP(1,44) tells you the symmetry type (an integer between 1 and 22) for the left-hand two indices of list 44 (which happens to be $a \leq b$). Hence, to determine the number of $a \leq b$ combinations belonging to symmetry species 3 (this means that $\Gamma_a \otimes \Gamma_b = 3$), one need only inspect the value of IRPDPD(3,ISYTYP(1,44)). Use of these variables is perhaps most easily learned by studying pieces of the ACES II source code, particularly programs such as vcc.f, lambda.f and dens.f. The variable ID is not used. All variables are integers. Used in post-SCF programs only.

This concludes the list of common blocks which are initialized by the CRAPSI subroutine and are therefore used in most programs. Of course, a number of other common blocks are used, but these are necessarily specific to the program in which they are used and knowledge of their content is usually not necessary for program interfacing or developement.

F. Symmetry Information

Some useful information regarding the molecular symmetry is stored on the JOBARC file and is used by various programs in the ACES II program system. In order to understand old code or to write new code which uses these data structures, a careful reading of this section is strongly recommended.

For each molecule, two point groups are considered. The first is the actual point group to which the molecule belongs, and is called the "full" point group (FPG). The second point group is an Abelian subgroup of the FPG, and is termed the "computational" point group (CPG), since it is the group in which the electronic structure calculation is actually performed. The FPG and the CPG are both determined by the *JODA* program in subroutine SYMMETRY, and this information is written to the output file. For each group, a large quantity of information regarding the effects of the operations on the atoms and the operations themselves are also determined by *JODA* in subroutine SYMDRV and its dependents.

F..1 Point Groups

The FPG and CPG are stored on JOBARC in records FULLPTGP and COMPPTGP, respectively. The order of each group is stored in records FULLORDR and COMPORDR, respectively. It is usually necessary to know the order of the point group before reading other JOBARC records dealing with symmetry.

F..2 Symmetry Operations

Subroutine CHRTAB in *JODA* forms $3x3$ Cartesian representations for *all* symmetry operations in the FPG and the CPG and writes them to the JOBARC file records FULLSYOP and COMPSYOP, respectively. In subsequent programs, these operations are indexed by number and are ordered by class as follows:

C_1 group

Operation 1 is the identity.

C_s group

Operation 1 is the identity, operation 2 is σ_{xy}.

C_i group

Operation 1 is the identity, operation 2 is i.

C_n groups

Operation j corresponds to C_n^j.

D_n groups (n even)

Operation Number	Operation Type
1 thru $n-2$	$C_n^{i\ **}$
$n-1$	C_2 $(=C_n^{\frac{n}{2}})$
n thru $n-1+\frac{n}{2}$	C_2'
n+$\frac{n}{2}$ thru $2n-1$	C_2''
$2n$	E

S_{2n} groups

Operation Number	Operation Type
1 thru n	$S_{2n}^{2i-1\ *}$
$n+1$ thru n	$C_n^{i\ *}$

D_n groups (n odd)

Operation Number	Operation Type
1 thru $\frac{n}{2}$ - 1	$C_n^{i\ **}$
$\frac{n}{2}$ thru $2n-1$	C_2
$2n$	E

C_{nh} groups [1].

Operation Number	Operation Type
1 thru n	S_n
$n+1$ thru $2n$	$C_n^{i\ *}$

C_{nv} groups (n even)

Operation Number	Operation Type
1 thru $n-2$	$C_n^{i\ *}$
$n-1$	C_2
n thru $n-1+\frac{n}{2}$	σ_v
$n+\frac{n}{2}$ thru $2n-1$	σ_d
$2n$	E

C_{nv} groups (n odd)

[1] The S_n operations are ordered as follows: S_n, S_n^{n+2}, S_n^3, $S_n^{n+4}\cdots S_n^{2n-1}$ (n odd); S_n, S_q, S_n^3, S_q^{q+2}, S_n^5, $S_q^3\cdots S_n^{n-1}$, E (n even, where $q=\frac{n}{2}$)

Operation Number	Operation Type
1 thru $\frac{n}{2}-1$	C_n^{i} *
$\frac{n}{2}$ thru $2n-1$	σ_v
$2n$	E

D_{nh} groups (n even) [2]

Operation Number	Operation Type
1 thru $n-2$	C_n^{i} **
$n-1$	C_2
n thru $2n-3$	S_n
$2n-2$	i
$2n-1$ thru $2n+\frac{n}{2}-2$	C_2'
$2n+\frac{n}{2}-1$ thru $3n-2$	C_2''
$3n-1$ thru $3n+\frac{n}{2}-2$	σ_v
$3n+\frac{n}{2}-1$ thru $4n-2$	σ_d
$4n-1$	σ_h
$4n$	E

D_{nh} groups (n odd) [3].

Operation Number	Operation Type
1 thru $n-1$	C_n^{i} ** pair of operations.
n thru $2n-2$	S_n
$2n-1$ thru $3n-2$	C_2
$3n-1$ thru $4n-2$	σ_v
$4n-1$	σ_h
$4n$	E

D_{nd} groups (n even)

[2] S_n operations ordered as in C_{nh}, n even.
[3] S_n operations ordered as in C_{nh}, n odd

Operation Number	Operation Type
1 thru $n-2$	C_n^{i} **
$n-1$	$C_2 (= C_n^{\frac{n}{2}})$
n thru $2n-2$	S_{2n}^{2i-1} **
$2n-1$	i
$2n$ thru $3n-1$	C_2'
$3n$ thru $4n-1$	σ_d
$4n$	E

D_{nd} groups (n odd)

Operation Number	Operation Type
1 thru $n-1$	C_n^{i} **
n thru $2n-2$	S_{2n}^{2i-1} **
$2n-1$	i
$2n$ thru $3n-1$	C_2
$3n$ thru $4n-1$	σ_d
$4n$	E

T group

Operation Number	Operation Type
1 thru 4	C_3
5 thru 8	C_3^2
9 thru 11	C_2
12	E

T_d group

Operation Number	Operation Type
1 thru 6	S_4
7 thru 9	C_2
10 thru 17	C_3
18 thru 23	σ_d
24	E

O_h group

Operation Number	Operation Type
1 thru 3	C_4
4 thru 6	S_4
7 thru 9	σ_h
10 thru 12	C_4^2
13 thru 15	C_4
16 thru 18	S_4
19 thru 24	C_2
25 thru 30	σ_d
31 thru 38	C_3
39 thru 46	S_6
47	i
48	E

O group

Operation Number	Operation Type
1 thru 3	C_4
4 thru 6	C_2
7 thru 9	C_4
10 thru 15	C_2'
16 thru 23	C_3
24	E

T_h group

Operation Number	Operation Type
1 thru 3	σ_d
4 thru 6	C_2
7 thru 10	C_3
11 thru 14	C_3^2
15 thru 18	S_6
19 thru 22	S_6^2
23	i
24	E

I_h group

Operation Number	Operation Type
1 thru 6	C_5
7 thru 12	C_5^2
13 thru 18	S_{10}
19 thru 24	S_{10}^3
25 thru 30	C_5
31 thru 36	C_5^2
37 thru 42	S_{10}
43 thru 48	S_{10}^3
49 thru 68	C_3
69 thru 88	S_6
89 thru 103	C_2
104 thru 118	σ_v
119	i
120	E

I group

Operation Number	Operation Type
1 thru 6	C_5
7 thru 12	C_5^2
13 thru 18	C_5
19 thru 24	C_5^2
25 thru 44	C_3
45 thru 59	C_2
60	E

F..3 Permutation Vectors

For each symmetry operation in the CPG and FPG, a vector is constructed which associates each atom in the Z-matrix with its image under the transformation. Atoms are numbered according to their absolute position in the Z-matrix. For example, if operation i maps atom 4 into atom 17, the value of the fourth element of the permutation vector is 17. These vectors are stored in JOBARC records FULLPERM and COMPPERM, respectively. The set of permutation vectors is stored on disk as a single vector of length $N_{atoms}h$, where h is the order of the point group. Each permutation vector occurs sequentially in the composite record, and follows the ordering of the symmetry operations given in the preceding subsection.

F..4 Orbit-Specific Information

A group of symmetry equivalent atoms is called an *orbit*. Analysis of symmetry in terms of the orbits of a molecule is very useful and is used extensively in ACES II. Information on JOBARC dealing with the orbits is summarized in the following. Records FULLNORB and COMPNORB contain the value of the number of orbits in the FPG and CPG, respectively. Records FULLSTGP and COMPSTGP contain the site group (the subgroup under which the atoms in the orbit are mapped into themselves by all operations) of each orbit. Records FULLPOPV and COMPPOPV contain the number of atoms in each orbit. Finally, the records FULLMEMB and COMPMEMB order the Z-matrix centers by orbit, so that the FULLMEMB and FULLPOPV (or COMPMEMB and COMPPOPV) records can be used together to list the individual atoms in each orbit.

F..5 Symmetry Types

The post-SCF program modules of ACES II assign an integer value to specific binary combinations of orbital types (occupied or virtual; α or β) which is then used to determine symmetry information (see, for example, the description of the SYMPOP common block). A complete listing of the assignments for these values is given below

"Symmetry Number"	Distribution Type
1	$a < b$
2	$\bar{a} < \bar{b}$
3	$i < j$
4	$\bar{i} < \bar{j}$
5	$a \leq b$
6	$\bar{a} \leq \bar{b}$
7	$i \leq j$
8	$\bar{i} \leq \bar{j}$
9	ai
10	$\bar{a}\bar{i}$
11	$a\bar{i}$
12	$\bar{a}i$
13	$a\bar{b}$
14	$i\bar{j}$
15	$a\bar{b}$
16	ia
17	$\bar{i}\bar{a}$
18	$i\bar{a}$
19	ab
20	$\bar{a}\bar{b}$
21	ij
22	$\bar{i}\bar{j}$

Note that entries 13 and 15 are identical, which is simply a result of carelessness in the initial coding. Although not all possible combinations are given, these are the only combinations which are used in the distributions which make up the integral lists.