Computing for Science (CFS) Ltd.,
CCLRC Daresbury Laboratory.

Generalised Atomic and Molecular Electronic Structure System

G A M E S S - U K

USER'S GUIDE and REFERENCE MANUAL

Version 8.0 June 2008

PART 10. THE UTILITY FUNCTIONS

M.F. Guest, J. Kendrick, J.H. van Lenthe and P. Sherwood

Contents

1 Introduction

The Utility functions associated with the program provide facilities for

1. data handling commonly required when manipulating GAMESS–UK generated datasets, and is oriented towards manipulation of the 2-electron integral files in particular. The program will copy, list, check, summarise, edit etc such files.
2. generation of the Library file required when performing non-local pseudopotential calculations.
3. generation of plotting output using the grids of densities, potentials etc generated under previous control of the GRAPHICS directive and assumed resident on a Dumpfile.
4. manipulation of orbitals (vectors) with extended flexibility (SERVEC) e.g., if orthogonalisation of the 1-electron integrals is needed. This module may be invoked as part of the Utilities module, when an existing dumpfile is provided (or the 1-electron integrals are not needed) and later in the input after the basis-set and geometry information is read. SERVEC may prove useful for Valence Bond calculations (although not exclusively) and is only included when VB is specified when GAMESS is configured.

Note that these functions are performed independently of the computation associated with a specific molecular calculation, with the data input requirements quite separate from that described in sections 3-6 above. The following points should be noted:

- Certain of the File Manipulation Utilities require input and output areas to be simultaneously operational. When using a GAMESS–UK dataset it is possible for input and output to be taken from the same dataset. When an block is written to a dataset, all blocks except that explicitly written remain unaltered.
- The restrictions applicable to use the same dataset for input and output within certain of the Utilities (specifically COPY, EDIT and COPYDUMP) concern the possibility of overlapping fields. The following rules are applicable:
  - Operations involving non-overlapping data areas on the same dataset are always legal.
  - Operations involving overlapping fields are valid if the starting block number of the output area is less than or equal to the starting block number of the input area.
  - Operations involving overlapping fields are not valid when the starting block number of the input area is less than that of the output area, and yet the output area commences within the input area.
- The module is a collection of routines, which may be called in any order, as dictated by the user. Each routine will require some input data, and will perform a task. When the work of a given routine is complete, control is handed to the next routine nominated by the user. On successful completion, all routines reply with the printed message:

```
'NAME' COMPLETED
```

where NAME represents the task. The following paragraphs describe the tasks and their purpose.

2 Utility Directives

The Utilities module is requested by a single data line containing the data string UTILITY in the first data field, and should immediately follow any pre-directives. Subsequent utility functions are user-driven through the sequence of directives outlined below.

3 File Manipulations

3.1 The LIST Utility

Data for the LIST utility consists of one dataline, read to variables TEXT, DDLIST, IBLK, NBLK, TYPE using format (2A,2I,A).

- TEXT should be set to the character string LIST.
- DDLIST should be set to the LFN of the input dataset.
- IBLK is an integer used to specify the starting block number of the input dataset. Instead of using an integer to specify the IBLK parameter, the character * may be used, and in this case listing commences from the 'current position' of the input file. The syntax * + m or * - m, where m is an integer may be used to specify IBLK, and will be taken to mean the current block number + or - m respectively.
- NBLK is an integer used to specify the number of blocks to be listed. If NBLK is set to zero (or omitted), listing continues until an ENDFILE block is reached.
- TYPE should be set to the appropriate character string characterising the format of the 2-electron file to be LISTed - valid strings include 2E (for conventional integral format), P (for P-supermatrix files) and JK (for JK-supermatrix files).

A list of the 2-electron integrals, with associated indices (labelled I, J, K and L) stored in the dataset assigned by the file DDLIST will be printed. Listing commences from the block specified by IBLK, and continues for NBLK blocks (or to the first ENDFILE if NBLK=0).

Example 1

All datasets are positioned at block 1 when the service program is first entered. If the following dataline is presented:

```
LIST ED6 * + 1 0 2E
```

the file assigned to ED6 will be listed from block 2 (* + 1=2) until the first ENDFILE block is encountered.

Example 2

```
LIST MT5 98 5 P
```

This example list the P-Supermatrix file assigned to MT5 from block 98 for 5 blocks. The file will be positioned at block 103 on completion of the list.

3.2 The PUNCH Utility

Data for the PUNCH directive consists of one dataline read to variables TEXT, DDPUNC, IBLK, NBLK using the format (2A,2I).

- TEXT should be set to the character string PUNCH.
- DDPUNC specifies the LFN of the input dataset. This file must be associated to the FILE specified in the JCL by the pre-directives.
- IBLK is an integer used to specify the starting block number of the input file. Alternatively the '*' or '* + m' or '* - m' formats may be used, as described in the LIST directive.
- NBLK is an integer used to specify the number of blocks to be 'punched'. Alternatively, if NBLK is zero (or omitted), the file will be punched until an ENDFILE block is reached

The 2-electron integrals, with associated indices, as stored on the file specified by the DDPUNC starting at block IBLK for NBLK blocks (or until an ENDFILE block is detected if NBLK=0) are punched. Each 'punched' line is produced by the Fortran code of the form:

```
      WRITE(7,1)ISEQ,N,(I(M),J(M),K(M),L(M),G(M),M=1,N)
1     FORMAT(I4,I2,2(4I4,F17.10)
```

where

- ISEQ is a card sequence number.
- N is an integer whose value may be 1 or 2.
- I,J,K,L,G are arrays to hold the indices and values respectively of a maximum of two integrals.

The PUNCH directive should be used sparingly, since the output generated by this directive is considerable even for small NBLK numbers.

Example 1

```
PUNCH MT1 75 5
```

The file assigned to MT1 will be 'punched' starting at block 75 for 5 blocks. The dataset will be positioned at block 80 on termination of the PUNCH task.

Example 2

```
PUNCH ED2 * 0
```

The file assigned to ED2 will be 'punched' commencing from the current position (usually block 1 if first access) until the ENDFILE block has been encountered.

3.3 The SUMMARY Utility

Data for the SUMMARY utility consists of one dataline read to variables TEXT, DDSUM, IBLK, NBLK using format (2A,2I).

- TEXT should be set to the character string SUMMARY.
- DDSUM should be set to the file of the dataset to be summarised.
- IBLK is an integer used to specify the starting block of the input file. Alternatively the '*' or '* + m' or '* - m' formats may be used, see LIST routine.
- NBLK is an integer used to specify the number of blocks to be summarised. Alternatively, if NBLK is set to zero (or omitted), the file will be summarised until an ENDFILE block is reached.

The SUMMARY routine causes one printed line per block, and each block is assumed to be in MAINFILE format. The printing consists of the first and last 2-electron integral in the block (with associated indices) plus the number of integrals stored in the block.

Example

```
SUMMARY MT4 1 0
```

The file assigned to MT4 is summarised from block 1 until the first ENDFILE block is encountered.

3.4 The CHECKSUM Utility

Data for the CHECKSUM routine consists of one dataline read to variable TEXT, DDCHEK, IBLK, NBLK using the format (2A,2I).

- TEXT should be set to the character string CHECKSUM.

- DDCHEK should be set to the LFN used to assign the dataset to be 'checksummed'.
- IBLK is an integer used to specify the starting block number of he input file. Alternatively the '*' or '* + m' or '* - m' formats may be used, as described by the LIST directive.
- NBLK is an integer used to specify the number of blocks to be 'checksummed', or if set to zero (or omitted), the dataset will be 'checksummed' until an ENDFILE block is detected.

The CHECKSUM routine scans the nominated blocks of the stated dataset. Blocks with an invalid Checksum words cause the message:

```
BLOCK x CHECKSUM ERROR
```

to be printed, where x is the block number. ENDFILE blocks give rise to the message:

```
BLOCK x ENDFILE
```

The CHECKSUM routine may be used on any GAMESS–UK generated dataset.

Example

```
CHECKSUM ED2 1 999
```

The file assigned to ED2 will be 'checksummed' from block 1 for 999 blocks. The dataset will be positioned at block 1000 on completion of the operation.

3.5 The CHECK Utility

The CHECK routine provides a more complete check facility for the MAINFILE than is available when using the CHECKSUM directive. Data consists of one line read to variables TEXT, DDCHEK, IBLK, NBLK, FP, FN using format (2A,2I,2F).

- TEXT should be set to the character string CHECK.
- DDCHEK specifies the file name of the input dataset.
- IBLK is an integer used to specify the starting block of the input dataset. Alternatively the '*' or '* + m' or '* - m' formats may be used, as described in LIST routine.
- NBLK is an integer used to specify the number of blocks to be 'checked'. If NBLK zero, checking continues until the first ENDFILE block is detected.
- FP is a +ve real variable whose value is used to monitor +ve valued 2-electron integrals. If such an integral is found to be greater than FP, its value plus associated indices is printed.
- FN is a +ve real variable. If the absolute value of a -ve value 2-electron integral is found to be greater than FN, its value plus its associated indices is printed. If FN is omitted, then FP may also be omitted, the default value 5.0 being chosen for the latter.

The CHECK routine reads the nominated blocks of a given dataset, printing messages if unusual conditions are met. The printing is of the general form:

```
BLOCK x message
```

where x is the block number, and 'message' denotes explanatory text. The text messages may be one of the following:

Message	Explanation
CHECKSUM ERROR	A block containing an invalid checksum word has been detected.
NOT MAINFILE	Indicates that whilst the block is in standard format, it is not in MAINFILE format.
ENDFILE	An ENDFILE block has been detected.
INVALID NUMBER OF INTEGRALS	The 2-electron integral counter word has been set -ve, or to a value greater than 340.
INVALID FP	A 2-electron integral has been detected which is not in a normalised floating point representation.
G,I,J,K,L= f,i,j,k,l	The characters f,i,j,k,l correspond to the value and associated indices respectively of a 2-electron integral. A +ve integral whose value is greater than FP, or a -ve integral whose absolute value is greater than FN has been detected.

Example

```
CHECK MT5 * 0 8 .3
```

The file associated with MT5 will be checked from the current position to the first ENDFILE block. Positive integrals greater than 8 or -ve integrals whose absolute values are greater than .3 will be flagged.

3.6 The SCAN Utility

Data for the SCAN directive consists of one line, read to variables TEXT, DDSCAN, IBLK using format (2A,2I).

- TEXT should be set to the character string SCAN.
- DDSCAN specifies the file name of the dataset to be 'scanned'.
- IBLK is an integer used to specify the starting block for the scan operation. The formats '*' or '* + m' or '* - m' may also be used, as described in the LIST directive.

The SCAN routine will read the nominated dataset from the block specified by the IBLK parameter, until the first ENDFILE block is detected. The dataset will be positioned immediately after the ENDFILE block on termination of the operation. The SCAN routine will read blocks containing invalid checksum words without diagnosing an error.

Example

A dataset has been assigned to a file MT4, and is to be positioned immediately after the third ENDFILE block. The following data should be presented:

```
SCAN MT4 1
SCAN MT4 *
SCAN MT4 *
```

3.7 The COPY Utility

Data for the COPY routine consists of one line, read to variables TEXT, DDIN, IBLK, DDOUT, JBLK, NBLK using format (2A,I,A,2I).

- TEXT should be set to the character string COPY.
- DDIN specifies the file name of the input dataset.
- IBLK is an integer used to specify the starting block of the input dataset. The '*' or '* + m' or '* - m' formats also may be used, as described in the LIST directive.
- DDOUT specifies the file name of the output dataset.
- JBLK is an integer used to specify the starting block of the output dataset. The '*' or '* + m' or '* - m' formats may also be used, as described in the LIST directive.
- NBLK is an integer specifying the number of blocks to be copied. If the value zero is entered, the copy is terminated when an ENDFILE block is detected on the input file. In this case the ENDFILE block is copied to the output file.

Example 1

```
COPY ED2 11 MT2 * 0
```

The dataset assigned to the file ED2 will be copied from block 11 until the first ENDFILE block is detected, the copied blocks being routed to a dataset assigned to the file MT2 starting at the current position.

Example 2

```
COPY ED4 500 ED4 499 10
```

The above example illustrates a valid 'intra' dataset copy with overlapping input and output areas. The net effect is to move the information one block down the dataset.

Example 3

```
COPY ED4 500 ED4 501 10
```

The above example illustrates an invalid 'intra' dataset copy. An error will be diagnosed when block 501 is read, since it will have been overwritten by the copied block 500.

3.8 The EDIT Utility

The EDIT utility is used to copy, in a selective manner, MAINFILE datasets. The selectivity arises because integrals whose absolute value are less than an input threshold are not copied to the output dataset. The first line of data is read to variables TEXT, DDOUT, JBLK, IACC using format (2A,2I).

- TEXT should be set to the character string EDIT.
- DDOUT specifies the file name of the output dataset.
- JBLK is an integer specifying the starting block of the output dataset. Alternatively the formats '*' or '* + m' or '* - m' may be used, as described in the LIST directive.
- IACC the quantity 10**(-IACC) is calculated, and used as the threshold.

Subsequent datalines define the input files, each line being read to variables DDIN, IBLK, LBLK using format (A,2I). An 'edited' copy of the dataset specified by DDIN starting at block IBLK and continuing up to but not including LBLK will be routed for output. The number of blocks scanned will be equal to LBLK-IBLK. Note that '*' or '* + m' or '* - m' formats may be used to specify IBLK, and that if LBLK is set to zero, editing will continue until the first ENDFILE block is detected on the input dataset. Note that if an ENDFILE block is detected before LBLK is reached, input processing will cease, so that block LBLK should be regarded as the highest block number reachable. Further 'input file definition lines' may be presented, the output generated by each line being appended to the already generated EDIT output. The data sequence is terminated by a line whose first data field contains the character string END. The routine will then append an ENDFILE block to the output dataset.

Example

```
EDIT MT4 * 60
ED2 1 0
ED3 1 0
END
```

The MAINFILE held in two sections starting at block 1 of ED2 and ED3 is edited to an file assigned to MT4, where it will be held in the form of one section terminated by an ENDFILE

block. The editing threshold of 10**(-60) ensures that all integrals are transcribed.

Example

```
EDIT ED2 1 7
ED3 1 0
END
```

The dataset assigned to ED3 will be edited to the dataset assigned as ED2, both files commencing at block 1, the threshold being 10^{-7}. Editing will terminate when an ENDFILE block is detected on ED3. The EDIT routine is here being used to reduce the size of the MAINFILE, at the cost of the loss of some accuracy, thereby reducing subsequent SCF iteration times. It is suggested that an editing threshold of 10^{-7} is used, SCF convergence to tolerance of 10^{-4} is all that is reasonable.

3.9 The ENDFILE Utility

Data for the ENDFILE directive consists of a single line read to variables TEXT, DDOUT, JBLK using format (2A,I).

- TEXT should be set to the character string ENDFILE.
- DDOUT specifies the LFN assigned to the output dataset.
- JBLK is an integer used to specify the block number where an ENDFILE block is to be written. Alternatively, the '*' or '* + m' or '* - m' formats may be used, as described in LIST directive.

Example

```
ENDFILE MT0 *
```

An ENDFILE block is written to a dataset assigned to MT0, at the current position.

3.10 The COPYDUMP Utility

Data for the COPYDUMP directive consists of a single dataline read to variables TEXT, DDIN, IBLK, DDOUT, JBLK using format (2A,I,A,I).

- TEXT should be set to the character string COPYDUMP.
- DDIN specifies the file name of the input dataset.
- IBLK is an integer used to specify the starting block of the Dumpfile resident on the input dataset. Alternatively, the formats '*' or '* + m' or '* - m' may be used, as outlined in the LIST directive.

- DDOUT specifies the LFN of the output dataset.
- JBLK is an integer used to specify the starting block for the output dataset. As before formats '*' or '* + m' or '* - m' may be employed.

The Dumpfile resident on the dataset nominated by the DDIN parameter, and starting at block IBLK is copied to the dataset nominated by DDOUT starting at block JBLK.

Example

```
COPYDUMP ED2 1 MT2 *
```

The Dumpfile starting at block 1 of ED2 is copied to MT2 starting at the current position.

3.11 The FINDDUMP Utility

Data for FINDDUMP consists of one line, read to variables TEXT, DDIN, IBLK, NBLK, TEST using format (2A,2I,A).

- TEXT should be set to the character string FINDDUMP.
- DDIN specifies the file name for the input dataset.
- IBLK is an integer used to specify the starting block number of the input file. Alternatively, '*' or '* + m' or '* - m' may be used, as described in the LIST directive.
- NBLK is an integer used to specify the number of blocks to be read. NBLK may not be set to zero.
- TEST should be set to one of the character strings HIGH or LOW, if omitted the default is LOW.

The routine scans over the nominated blocks, seeking out evidence of valid Dumpfiles. Upon detection of a valid Dumpfile the program will print a summary of the Dumpfile, the scale of the printing being controlled by the setting of variable TEST. If TEST=HIGH, a fairly complete listing of the Dumpfile will be issued, if TEST=LOW, only a low level of information will be printed. The FINDDUMP facility will skip blocks containing invalid checksum words.

Example

```
FINDDUMP ED3 1 600 HIGH
```

will cause the first 600 blocks of the dataset assigned to ED3 to be scanned for Dumpfiles. If Dumpfiles are detected, a complete listing will be printed.

3.12 The STOP Directive

Data for the STOP directive consists of one dataline with either the character string STOP or EXIT in the first datafield. All data sets will be closed, and execution ended. A 'STOP' dataline must be presented last in the datastream.

4 SERVEC - Vector Service module

This module was originally written to provide vector (orbital) manipulation capabilities to the ATMOL package. Sometimes also in GAMESS more flexible user-controlled orbital manipulations are required, e.g. for the VB package. Since SERVEC shares many routines with VB, it is included in that set of programs. The package works exclusively in a non-symmetry adapted basis and can write vector-sections of arbitrary dimensions. These are not recognised by the normal GAMESS routines. If however the keyword 'gamess' is used on the write directive or the number of vectors equals their dimension a normal gamess vector-section is written, which may be back symmetry adapted using a vectors getq in GAMESS. The order of directives is very important for the result and an auxiliary dumpfile to store intermediate vectors may be required.

It is invoked by specifying the keyword SERVEC (all 6 characters to distinguish it from SERV), either at the beginning of the input (like SERV), when no 1-electron integrals are available, or after the basis and geometry specification, in which case the one-electron integrals are available. The restart-like situation is created, allowing one to startup from the vectors section generated by servec, without a preceding gamess-job This package has always been a ad-hoc toolchest and can be adapted easily to suit ones needs.

If one for example would like a gamess calculation to start on a (Lowdin) orthogonalised set of unit vectors, one would have to use (a default basis will not do)

```
title
h2co - 3-21g - closed shell SCF
adapt off
zmatrix angstrom
c
o 1 1.203
h 1 1.099 2 121.8
h 1 1.099 2 121.8 3 180.0
end
basis 3-21g
servec
extra
1 to 22 end
s
lowdin
1 to 22 end
write ed3 1 9
finish servec
vectors getq ed3 1 9
enter
```

4.1 Notation Conventions

```
     .. istring   =  string of integer numbers finished by 'end'
                     'to' convention is allowed i.e. 2 to 5 = 2 3 4 5
     .. tapnam    =  atmol file
                     next parameters are iblock (start-block
                     for dumpfile) and isect (section-number)
     .. 'a'/'b'   =  choice between textstrings 'a' and 'b'
     .. 'flop'    =  text-string  (directive is always text)
     .. ('aa')    =  optional text-string
     ..  **       =  after ** comment is given
     ..  ex.      =  ex. denotes an example (starting in column 1)
 ** note  tapnam etc. must be on same card                          **
 **       istring may extend over many lines                       **
```

4.2 The TITLE Directive

The next line is a title that will be used in all the following writes

```
      ex.  title
      ex.  this is the title that will appear on vector-files
```

4.3 The READ Directive

```
 read   tapnam  ('notran')   ** read vectors
```

If 'notran' is specified the vectors are not transformed to the original basis. All ctrans (symmetry adaption) information is lost however (**not retained**)

```
      ex.   read ed3  1 1

 read  'input' ndim nmdim    ** read vectors from input free format
       ex    read input 3 2
 read  'free' ndim nmdim     ** read vectors from input using read *

 read  'format'    ** read vectors and occupations etc. formatted
```

For the exact form of input *see* the print vectors 'format' directive. this option is used to transport vectors as a text-file.

```
      ex.   read format
      ex.   ......................
      ex.   ............................
```

4.4 The WRITE Directive

```
 write tapnam  ('gamess') ** write vectors (see read)
                          ** if 'gamess' is given vectors are gamess-compatible
```

4.5 The INIT Directive

```
init   tapnam         ** produce new (empty) dumpfile
      ex.   init  ed3 1
```

4.6 The S Directive

```
s  'print'             ** read overlap integrals
  or
s tapnam (without section) ndims 'print' ** read s from foreign dumpfile
                       **  if print specified the s-matrix is printed
      ex.   s
      ex.   s ed5 1 122
```

4.7 The SCREEN Directive

```
 screen 'crit'   ** make all s-matrix elements < crit exactly zero
                 ** if crit is omitted the current criterion is used
```

4.8 The PRINT Directive

```
 print  'on'/'off'    ** switches print flag
 print  'vectors'  nv  ('occ'/'eig')  ** print first nv vectors
                             **  default nv = nmdim
                             **  if 'occ' => give occupations
                             **  if 'eig' => give orbital energies
      ex.   print off
     2ex.   print vectors 7

 print  'vectors'  nv  'format' format  ** print vectors formatted
          ** according to format(format). nv=0 means nv = nmdim
          **  default format =  5f16.9
          ** this option is used to transport vectors as text-file
          ** see also *read format* / the output looks as follows
          *** 'vectors' ndim  nv     format
          *** ((vc(i,j),i=1,ndim),j=1,nv)  (format(format))
          *** 'occ'  nn         ** nn = 0 for no occupations
          *** (occ(i),i=1,nn)   (format(format))
          *** 'eig'  nn         ** nn = 0 for no eigen-values
          *** (eig(i),i=1,nn)   (format(format))
          *** 'end of format-print'
          ** the input for read format should look the same

     ex.   print vectors 0 format   5e16.10
```

4.9 The SCHMIDT Directive

```
 schmidt  istring     ** schmidt orthogonalize the orbitals specified
      ex.   schmidt   1 5 7 3 end
```

```
schmidt  'set' ('norm'/'nonorm') istring istring
                  ** schmidt first istring onto second
                  ** (no internal orthogonalisations)
                  ** if 'norm' is specified the set is normalised
                  == The default is nonorm (the string may be omitted)
     ex.   schmidt  set  1 4 end 2 3 end
     ** as accuracy of schmidt crit is used
```

4.10 The LOWDIN Directive

```
lowdin   istring     ** lowdin orthogonalise the orbitals specified
     ex.   lowdin    2 3 4 5  end
     ** as accuracy of lowdin crit*10 is used
```

4.11 The NORMALISE Directive

```
normalise istring   ** normalise the orbitals specified
```

4.12 The CHECK Directive

```
check  ('print')    **  check vectors on orthonormality  within crit.
                    **  if in the overlap-matrix of the vectors an
                    **  element > crit is found the matrix is printed
                    **  if 'print' is selected it is printed anyway
```

4.13 The SDIAG Directive

```
sdiag ('set') ('istring') **  diagonalise s-matrix and leave its eigen-
                          **  vectors as current vector-set, ordered by
                          **  decreasing eigen-value / needs s-matrix
                          **  see  's' and 'crit' directives
                          **  ** if set is specified the s-matrix
                          **  for that set of unit vectors is diago-
                          **  nalised and the eigenvectors are of
                          **  the original dimension
```

4.14 The LOCMP Directive

```
locmp iset nset     ** localise within the set of nset orbitals
                    ** starting at orbital iset.
                    ** v. magnasco, a. perico jcp 47,971 (1967)
istring             ** a string of numbers indicating
                    ** the ao' to localise an mo on. when all mo's
                    ** to be localised are specified, give 'end'
'end'               ** # mo's to be localised  .le. nset
                    ** an moperm may be useful before applying locmp
                    **   *** locmp needs s-integrals and vectors so
                    **   *** the 's' and a 'read' directive must
     ex.   locmp 3  7
```

```
     ex.   1 4 5  0
     ex.   1 2 3  0
     ex.   end
```

4.15 The MOPERM Directive

```
moperm   istring     ** permute mo's
     ex.   moperm   1 to 10 13 12  end
```

4.16 The AOPERM Directive

```
aoperm   istring     ** permute ao's
     ex.   aoperm  17 18 end
```

4.17 The COMBINE Directive

```
combine  tapnam  tapnam ** identical to atmolscf combine option
                        ** except that no orthogonalisation is done
                        ** combine first and second vector - set
      ** next cards define the origin of the ao's **
         ifr ito 'a'/'b' ii ** ao's ifr to ito (inclusive) in new set
         ... ...   ...   .. ** are ao's ii onwards of vectorset a/b
         ... ...   ...   .. ** go on until all ao's are assigned
         'end'
      ** next cards define the origin of the mo's **
         ifr ito 'a'/'b' ii ** same as for ao's / finish with 'end'
         'end'
     ex.   combine ed6 1 1 ed4 200 3
     ex.   1 5 a 1
     ex.   6 10 b 1
     ex.   11 20 a 6
     ex.   21 26 b 6
     ex.   end
     ex.   1  4  a 1
     ex.   5  7  b 1
     ex.   8 18  a 5
     ex.   19 26  b 4
     ex.   end
      **  Alternatively a simplified form may be used just specifying quantities
      **  and on the same line
     ex.  combine  ed6 1 1 ed4 200 3
     ex.  a 5 b 5  a 10  b 6  end
     ex.  a 4 b 3  a 11  b 8  end
```

4.18 The EXTRA Directive

```
extra    istring     ** add ao's and corresponding unit-vectors
                     ** (in existing vectors zeros are inserted)
                     ** warning  : ndim **must** equal nmdim
         ** e.g. one performed a dz-calculation on h2 using ao's
```

```
         ** 1,2 and 3,4 . now one adds polarisation functions (p)
         ** on each h-atom (p's will be 3,4,5 and 8,9,10). the start-
         ** orbitals for this new calculation are produced (starting
         ** from the old 4*4 set) by putting :
     ex.   extra 3 4 5 8 9 10 end
```

4.19 The SUMMARY Directive

```
summary              ** print summary of files and parameters
```

4.20 The TRANSFORM Directive

```
transform tapnam ('notran')   ** transform nmdim*ndim vectors in core
                              ** with a nmdim*nmdim set on tapnam
                              ** notran as in read
```

4.21 The OCCUP or OCC Directive

```
occup   nval (occ(i),i=1,nval) ** read occupation numbers (free form)
    ex.    occup 3  2.0 2.0 1.5
    ex2.   occ 2  2 1
         ** the occupation-numbers will appear on any tape4 or dump-
         ** file written , following this directive
```

4.22 The EIGEN Directive

```
eigen   nval              ** read nval orbital-energies on following
        (eig(i),i=1,nval) ** according to format(5e15.5)
    ex.    eigen 2
    ex.     -20.5673456     -11.43456
```

4.23 The CRIT Directive

```
crit    acr  npower ** set criterion to acr.10**(-npower)
                    ** (default  1.0e-14)
    ex.    crit    1.0   20
```

4.24 The NDIM Directive

```
ndim    nd   ** reduce ao-dimension to nd (loose last ao's)
    ex.    ndim    22
```

4.25 The NMDIM Directive

```
nmdim   nmd  ** reduce mo-dimension to nmd (loose last mo's)
```

4.26 The LOG Directive

```
log  'on'/'of' ** switch printflag for printing of directive-cards
               ** in interactive sessions log on is more forgiving
```

4.27 The ADDVEC Directive

```
addvec  tapnam ** add vectors (same dimension as current vectors)
               ** from specified file ,as last orbitals
    ex.    addvec ed3 1 2
addvec  'unit'  istring .. .. ** add specified unit vectors
                           ** (i.e. with 1.0 in positions specified by istring
                           **  respectively )
    ex.    addvec unit 11 7 1
```

4.28 The SYMMETRY Directive

```
symmetry  nao nirr n1 n2 n3 .. ** 'symmetry' selection of mo's
         ** nao = number of ao's in each mo to look at (max 8)
         ** nirr= number of representations to select  (max 12)
         ** n1,n2,n3 etc : dimensions of subsets within mo's
          ** next cards define selection (nirr cards) **
             (iao(i),ich(i),i=1,nao) ** iao = number of ao
             .. .. .. .. .. .. .. .. ** ich = 1,0 or -1 = sign of ao
             .. .. .. .. .. .. .. .. **       in mo  (parity is ok)
     ex.   symmetry 3 2  3 5
     ex.   1 1    3 0    12  -1
     ex.   1 0    4 0    10   1

symmetry 'h'            ** the symmetry is determined from the
                        ** (1-electron) h-matrix
        ** following directives may be issued / finish with 'end'
    'sets'  is1 is2 ... ** dimensions of subsets in orbital-space
   'rename' ir1 ir2 ... ** new numbers of the representations
    'end'               **
     ex.   symmetry h tape3
     ex.   order 1 4 2 3
     ex.   sets  1 4 8 1
     ex.   end
    ** this sorts the orbitals for water in a dz basis , keeping
    ** the 1s and the 1s* , the occupied and the virtual spaces
    ** apart / the a2 symmetry does not occur in the occupied
    ** space ,is therefore originally assigned # 4 and is reordered
    ** to position 2

*****************************************************************
**** a directive : crit 5 5 : is recommended before symmetry   ****
**** selection, since orbitals are usually only accurate to    ****
**** 5 figures                                                 ****
*****************************************************************
```

4.29 The MAXDIM Directive

```
maxdim 'low'/'medium'/'high' / set maximum dimension  (default low)
                            ** low : all actions possible
                            ** medium : no orthogonalisation possible
                            ** high  : and no combine possible
     ex.   maxdim   high
```

4.30 The ATOMMO Directive

```
atommo 'atom' istring 'mo' istring / sorts the mo's listed in the order of the atoms
    ex. atommo atom 1 2 3 end mo 1 2 3 4 5  end
```

4.31 The DUMPFILE Directive

```
dumpfile tapnam (without section)  / set the standard dumpfile to tapnam
             ** the dumpfile must exist
             ** default the GAMESS dumpfile is used
             ** if not available the first dumpfile vectors were read from

    ex. dumpfile ed4 1
```

4.32 The STOP / FINISH Directive

```
stop/finish     **  ends calculation
     ex.   stop
```

5 OPTBS - Single variable Basis Set optimisation

The program optimises a set of even tempered outer exponents, taking the existing basis as a start. Currently only a single parameter optimisation of the exponent ratio is available. Optimisation is possible for mSCF,MP2 and CASSCF and DFT; Dft can give problematic convergence though.

Data for OPTBS consists of one line, read to variables TEXT, OUTEXP,NEXP using format (A,2I). In addition to these required parameters, STEP and TOL may be optionally specified, using The TEXT,VARIABLE format (A,F). Also a PRINT (format(A)) may be specified.

- TEXT should be set to the character string OPTBS.
- OUTEXP specifies the index of the outer core exponent, i.e. the last one not changed.
- NEXP is the number of exponents to be optimised.
- STEP specified the stepsize taken in changing the exponent ratio (default 0.1)
- TOL is the termination criterion (default 0.01)

- PRINT gives output of all SCF-type calculations

Example

```
SERVER
OPTBS 15 2 STEP 0.1 TOL 0.01
....... GAMESS - INPUT .........
ENTER
```

Starting from the GAMESS input it will optimise exponents 16 and 17.