ACES III User Guide

ACES III Documentation

Quantum Theory Project
University of Florida
Gainesville, FL 32605

Contributors to the software:
R. J. Bartlett, E. Deumens, N. Flocke, T. Hughes, N. Jindal,
V. F. Lotrich, A. Perera, J. M. Ponton, B. A. Sanders, T. Watson

Software version: 3.0.5 Oct 2010
Document version: 3.0.5 A Oct 2010
Document formatted:

November 8, 2010

Contents

1 Overview

ACES III is a program which implements much of the functionality of ACES II in parallel. The program is designed to run on a number of Unix-based platforms, including AIX, Altix, Cray, and a number of Linux clusters. Although it retains some features of ACES II, ACES III is a completely new program, based on the Super Instruction Architecture Language (SIAL for short, pronounced "sail") developed by ACES QC in conjunction with the DoD High Performance Computing Modernization Program. The program was designed to attain excellent performance and scalibility on up to 1000 processors, and beyond.

Using ACES III, the following types of calculations can be performed, on both closed shell and open shell molecular systems:

- SCF, MBPT(2), and CCSD energy and gradient calculations.
- CCSD(T) energy calculations.
- SCF and MBPT(2) analytic Hessians.

There is also the capability to perform MBPT(2) energy, gradient and Hessian calculations with an ROHF reference.

The serial ACES II contains the capability to do other types of calculations as well. (DFT, CCSDT, STEOM for example. For a complete list see the ACES II documentation). However, new functionality is being continually added to ACES III as the need arises.

2 Quick Start Guide for Running ACES III:

1. Build a run directory on the file system on which you wish to run the job. Most MPP systems have one or more scratch file systems set up specifically for user jobs. This is normally the directory on which to set up the run directory.
2. Two files must be present in the run directory before running the ACES III executable. These are the ZMAT and GENBAS. The GENBAS contains the data describing basis set information, exactly as in a serial ACES II run. The ZMAT is similar to the ACES II ZMAT and contains the user's input parameters. In addition to the standard ACES II ZMAT, it also may contain a $*$SIP namelist section with additional parameters specific to ACES III. These parameters are described in the Parameter Description section below. If the default ACES III-specific parameter values are acceptable, there is no need to code the $*$SIP section, and the ZMAT is identical to that of a normal ACES II run.
3. A run script should be created to run the job. This script normally contains parameters for the batch queuing system of the computer platform. It also must sets an environmental variable ACES_EXE_PATH to the path of the ACES III executable program, xaces3. Then the script should run the xaces3 executable. Under many systems this involves execution of the mpirun command.

Example:
This is an example of a script used on cobalt, an Altix system at NCSA. This example shows the following :

1. The PBS parameters set the job name, runtime limit, number of processors, and job queue to use.

2. Some environment variables are set up, including ACES_EXE_PATH. They must be exported so that each processor spawned by the mpirun command receives the values.

3. A run directory is created ($TMPD), and the ZMAT and GENBAS files are copied into it. On most systems, it is preferable to also copy the xaces3 file into the run directory for performance reasons, as is done here.

4. Note the mpirun command. It runs xaces3 on 4 processors, using

   $TESTROOT/$atom/$type/CH4_AO_S.out as the stdout file. The "dplace -s1 -c0-3" is specific to Altix systems, telling the system to "pin" the memory to specific processors instead of allowing it to migrate from one processor to another.

```
#!/bin/ksh

#PBS -S /bin/ksh
#PBS -N CH4_AO_S
#PBS -j oe
#PBS -l ncpus=4
#PBS -l walltime=04:00:00
#PBS -q standard

##########################
## PORTING VARIABLES ##
##########################

tag=CH4_AO_S
atom=CH4
type=AO_S
nprocs=4
out=$tag.out

#############################
# Set up run environment #
#############################

export WORKDIR=/scratch/users/ponton
export ACES_EXE_PATH=/u/ac/ponton/ACESII/bin
export TESTROOT=/u/ac/ponton/xaces3_tests
```

```
export MPI_REQUEST_MAX=100000

TMPD=$WORKDIR/$tag
rm -rf $TMPD
mkdir -p $TMPD

cd $TMPD
cp $TESTROOT/$atom/$type/ZMAT .
cp $TESTROOT/GENBAS .
cp $ACES_EXE_PATH/xaces3 .

mpirun -np $nprocs dplace -s1 -c0-3 ./xaces3 >$TESTROOT/$atom/$type/CH4_AO_S.out
```

Here is the ZMAT file for this job:

```
CH4
H .431 -.762 -.739
H -.467 .762 -.426
H .778 .198 .739
H -.778 -.688 .587
C -.008 -.122 .040

*ACES2
!restart,symmetry=off
basis=CC-PVTZ,SPHERICAL=OFF # basis options
coordinate=cartesian,cc_conv=10,scf_conv=8
ref=uhf,calc=ccsd

*SIP
COMPANY = 1 1 3 0
IOCOMPANY = 2 1 1 0
MAXMEM = 900
SIAL_PROGRAM = scf_uhf_isymm_diis10.sio
SIAL_PROGRAM = tran_uhf_ao_sv1.sio
SIAL_PROGRAM = ccsd_uhf_ao_sv1_diis5.sio
SIAL_PROGRAM = lambda_uhf_ao_sv1_diis5.sio
SIAL_PROGRAM = one_grad_uhf_ao_sv1_diis5.sio
SIAL_PROGRAM = two_grad_uhf_ao_sv1.sio
```

Please note the following:

1. The *ACES2 section is a normal setup for a serial ACES II CCSD gradient calculation.
2. In the *SIP section, the COMPANY and IOCOMPANY parameters divide the 4 processors into "companies" of 3 worker processes and 1 server process (more about this later).

3. The "MAXMEM=900" forces the program to use 900 Mbytes of memory per processor. Different platforms may allow more or less memory per processor than this.

4. A number of SIAL_PROGRAM parameters are coded. These identify a sequence of SIAL programs that will be executed in order to calculate the CCSD gradient.

3 Super Instruction Architecture

ACES III was developed using the Super Instruction Architecture. This architecture views all data as a set of multi-dimensional blocks, usually with between 10000 and 250000 floating point numbers per block. A run-time system, called SIP, was developed to manipulate these blocks efficiently. Also, a high-level programming language, called SIAL, was implemented, allowing computational chemists to implement algorithms like SCF and CCSD fairly quickly and efficiently as SIAL programs. ACES III runs one or more SIAL programs to achieve its desired computational task. The SIAL programs are coded in the ∗SIP section of the ZMAT file as shown above. If the ∗SIP section is not present, ACES III will use the appropriate defaults to perform the calculation.

The Super Instruction Architecture divides the set of processors into a master process, and a number of worker and server processes. The master (which is also a worker) performs initialization and clean up chores for each SIAL program. The workers perform the actual computations (tensor contractions, matrix diagonalizations, etc.). The servers do nothing except store and serve data transferred to them by the workers. Each server process has a number of scratch files, which are created in the run directory. These files are used to hold data until the server receives a request from a worker process for the data. The scratch files may be identified as SCR∗, and are retained only for the duration of the SIAL program.

Workers and servers are configured by the COMPANY and IOCOMPANY parameters in the ∗SIP section of the ZMAT file (described below). Usually, a 3-to-1 ratio of workers to servers has been found to achieve good levels of performance.

A portion of the serial ACES II code, joda, is linked into ACES III, and executed at the beginning of each new SIAL program. The joda code uses data such as the gradient calculated within the individual SIAL programs to determine criteria for convergence of geometry optimizations and transition state searches, as well as vibrational frequency calculations. If the necessary parameters are set in the ∗ACES2 section to do geometry optimization, transition state search, or vibrational frequency calculations, the master process continues looping through all the SIAL programs until joda sets flags which indicate the job should be halted. At this time, scratch files are cleaned up and the job is terminated by the master.

ACES III has a "coarse grained" restart capability. Restart is performed at the SIAL program level. Each SIAL program passes data to the next SIAL program in the sequence by writing the data to a BLOCKDATA file in the run directory. If a job should terminate due to timeout (or some other reason), the run directory contains all necessary information to

restart the job. A user can simply modify the run script so as not to disturb any files in the run directory (except for removing all the SCR∗ files), and resubmit the job. The master process will restart the job at the beginning of the SIAL program which was in progress at the time the previous job died.

At the beginning of the job, a dryrun pass is made to estimate if there are enough processors to run the job. Each SIAL program is scanned in the dryrun to determine memory requirements. If the dryrun fails, the user should check the printout, which will give an estimate of the minimum number of processors required.

4 ∗SIP Parameter Description

The following are the parameters which may be coded in the ∗SIP section. There are no longer any required parameters. To override any parameter value, a ∗SIP section must be added to the ZMAT file, with a line specifying the parameter to be overridden. Examples are given at the end of this document.

- SIP_MX_SEGSIZE Blocksize for the atomic orbital dimension of each data block.
- SIP_MX_OCC_SEGSIZE Blocksize for the occupied orbital dimension (both alpha and beta spin) of each data block.
- SIP_MX_VIRT_SEGSIZE Blocksize for the virtual orbital dimension (both alpha and beta spin) of each data block.
- MAXMEM Amount of RAM per process, in Mbytes. Default is 900.
- COMPANY Description of the company of worker processes. This consists of 4 parameters, the company descriptor, platoon descriptor, number of workers in the company, and memory per worker. Only the number of workers is currently used by the program, the other fields are required, but not used. The default sets the number of workers at 3/4 of the total number of processors.
- IOCOMPANY Description of the company of I/O server processes. This parameter requires the same 4 fields as described under COMPANY above. The default sets the number of I/O servers at 1/4 of the total number of processors.
- DEBUG_PRINT Set DEBUG_PRINT=YES to obtain useful debugging information. Warning: This could generate a large print file.
- TIMERS Set TIMERS=YES to obtain extensive timing data for each SIAL program.
- AUTO_SEG_ALGORITHM This parameter controls the algorithm used to generate segment sizes when the parameters SIP_MX_SEGSIZE, SIP_MX_OCC_SEGSIZE, and SIP_MX_VIRT_SEGSIZE are not coded. The possible values are MEMORY_OPTIMIZED, which attempts to minimize memory usage, and SEGMENT_OPTIMIZED, which attempts to reduce the number of segments into which each coordinate axis is sub-divided. The default is SEGMENT_OPTIMIZED.

- LOCAL_PATH The Unix directory path in which to open the server's SCR∗ files. The default is the directory in which the program is started.
- MASTER_IS_WORKER Obsolete parameter. In an earlier implementation, the master process was not required to also be a worker. In the current implementation, the master is always a process in the worker company.
- INTEGRAL_PACKAGE Must take on the values ERD or GAMESS. Although ACES III was developed with the ability to use either GAMESS or ERD integrals, the GAMESS integrals do not work for any programs using 2nd derivative calculations. ACES III now uses only ERD integrals.
- FAST_ERD_MEMCALC This parameter determines whether ERD integral package memory requirements are estimated by a "fast" algorithm or actually pre-calculated directly. For large systems using 2nd derivative integral calculations (i. e. Hessians), the direct calculation can take a significant amount of time. To bypass this calculation and use the estimation technique, set FAST_ERD_MEMCALC = YES.
- NWORKTHREAD The number of internal memory buffers used by the workers for inter-company communication. Defaults to 20.

The following parameters only apply when using specific SIAL programs. These parameters are provided for use in "batch-parallel" jobs, in which the natural parallelism of specific types of calculations may be used to break a large problem into a number of smaller ones, and the results combined after all the jobs have completed.

- IHESS iatom1 iatom2
- JHESS jatom1 jatom2
  - Used in all Hessian SIAL codes.
  - The Hessian can be written as a 4-dimensional array hess(iatom,ix,jatom,jx), where ix,jx = 1, 3. IHESS and JHESS represent the range of atoms for which the Hessian is computed in the current job. The default values run the entire calculation in one job.
  - Since correlated Hessian calculations can be quite expensive it is useful to allow the user to partition the calculation among several jobs. These jobs may be submitted into a batch queue system, and scheduled more efficiently by the queuing system. Instead of running a large Hessian calculation over a large number of processors, and having to wait a long time for the processors to become available, it may be preferable to use IHESS and JHESS to reduce the problem to a number of smaller jobs, each one running on a smaller number of processors.
  - Another reason for dividing the job using IHESS and JHESS is that ACES III currently does not allow restarts within a SIAL program. So, instead of running the Hessian calculation as one long-running job, partitioning the job in this way allows the calculation to be done in a number of shorter jobs. If one of these small jobs aborts due to a hardware problem, the entire amount of time is not lost.

  - In order to use IHESS and JHESS correctly care must be taken to run the proper sial codes AND sum the results correctly. Users wishing to perform such calculations should contact the authors as this is a nonstandard application.

- ITRIP ITRIPS ITRIPE
  - ITRIP may be used in any of the SIAL codes for CCSD(T) calculations. It is used to partition a large job performing CCSD(T) calculations into smaller ones, similar to the way IHESS and JHESS are used to partition Hessian calculations.
  - Since the perturbative triples calculation can be written as $E = \sum_i \sum_{abc,jk} Z(abc,ijk) * T(abc,ijk)$ the contributions $E(i) = \sum_{abc,jk} Z(abc,ijk) * T(abc,ijk)$ can naturally be computed seperately. ITRIP defines the range of i, and has the default ITRIP = 1 max(NOCCA,NOCCB). This has the effect of performing the entire calculation as one job.

- SUB SUBB SUBE
  - SUB is used in the CCSD(T) SIAL codes. It represents the range of occupied data to be held within the program's distributed memory. This is used to improve disk performance, and has no effect on the results of the calculations.
  - Ideally SUB and ITRIP coincide. Note that care must be used if the ITRIP parameter is used to insure that the energy is properly summed. In order to most effectively use the SUB parameter the authors should be contacted.

5 Example ZMAT Files

The following are some example ZMAT files, showing how to set up the parameters to run different types of ACES III jobs. For a complete list of current SIAL programs, please see the SIAL Program Inventory document.

5.1 Geometry optimization jobs:

5.1.1 SCF(UHF) on Ar_6 cluster

```
Ar6 IN aug-cc-pvtz basis
AR 2.5 2.5 0.0
AR -2.5 2.5 0.0
AR 2.5 -2.5 0.0
AR -2.5 -2.5 0.0
AR 0.0 0.0 2.5
AR 0.0 0.0 -2.5

*ACES2
!restart,symmetry=off
GEOM_OPT=FULL
basis=AUG-CC-PVTZ,SPHERICAL=ON # basis options
```

UNITS=BOHR
coordinate=cartesian
ref=uhf,calc=scf

*SIP
SIP_MX_SEGSIZE = 30
SIP_MX_OCC_SEGSIZE = 27
SIP_MX_VIRT_SEGSIZE = 27
COMPANY = 1 1 24 0
IOCOMPANY = 2 1 8 0
MAXMEM = 900
SIAL_PROGRAM = scf_uhf_isymm_diis10.sio
SIAL_PROGRAM = gradscf.sio

5.1.2 MP2(UHF) on Ar_6 cluster

Ar6 IN aug-cc-pvtz basis
AR 2.5 2.5 0.0
AR -2.5 2.5 0.0
AR 2.5 -2.5 0.0
AR -2.5 -2.5 0.0
AR 0.0 0.0 2.5
AR 0.0 0.0 -2.5

*ACES2
!restart,symmetry=off
GEOM_OPT=FULL
basis=AUG-CC-PVTZ,SPHERICAL=ON # basis options
UNITS=BOHR
coordinate=cartesian
ref=uhf,calc=mbpt(2)

*SIP
SIP_MX_SEGSIZE = 30
SIP_MX_OCC_SEGSIZE = 27
SIP_MX_VIRT_SEGSIZE = 27
COMPANY = 1 1 24 0
IOCOMPANY = 2 1 8 0
MAXMEM = 900
SIAL_PROGRAM = scf_uhf_isymm_diis10.sio
SIAL_PROGRAM = mp2grad_uhf_sv1.sio

5.1.3 CCSD(UHF) on Ar_6 cluster

```
Ar6 IN aug-cc-pvtz basis
AR 2.5 2.5 0.0
AR -2.5 2.5 0.0
AR 2.5 -2.5 0.0
AR -2.5 -2.5 0.0
AR 0.0 0.0 2.5
AR 0.0 0.0 -2.5

*ACES2
!restart,symmetry=off
GEOM_OPT=FULL
basis=AUG-CC-PVTZ,SPHERICAL=ON # basis options
UNITS=BOHR
coordinate=cartesian
ref=uhf,calc=ccsd

*SIP
SIP_MX_SEGSIZE = 30
SIP_MX_OCC_SEGSIZE = 27
SIP_MX_VIRT_SEGSIZE = 27
COMPANY = 1 1 24 0
IOCOMPANY = 2 1 8 0
MAXMEM = 900
SIAL_PROGRAM = scf_uhf_isymm_diis10.sio
SIAL_PROGRAM = tran_uhf_ao_sv1.sio
SIAL_PROGRAM = ccsd_uhf_ao_sv1_diis5.sio
SIAL_PROGRAM = lambda_uhf_ao_sv1_diis5.sio
SIAL_PROGRAM = one_grad_uhf_ao_sv1_diis5.sio
SIAL_PROGRAM = two_grad_uhf_ao_sv1.sio
```

5.2 Transition state search for Dimethylmethylphosphate:

cccc DMMP NTS1 OPT cccc
H
C 1 r2
O 2 r3 1 a3
P 3 r4 2 a4 1 d4
O 4 r5 3 a5 2 d5
C 5 r6 4 a6 3 d6
C 4 r7 3 a7 2 d7
O 4 r8 3 a8 2 d8
H 2 r9 1 a9 3 d9
H 2 r10 1 a10 3 d10
H 6 r11 5 a11 4 d11
H 6 r12 5 a12 4 d12
H 6 r13 5 a13 4 d13
H 7 r14 4 a14 3 d14
H 7 r15 4 a15 3 d15
H 7 r16 4 a16 3 d16
O 2 r17 4 a17 5 d17
H 17 r18 2 a18 4 d18

r2 = 1.2245104875
r3 = 1.4168956473
a3 = 109.4716141695
r4 = 1.6381450698
a4 = 119.9794951257
d4 = 62.3758493867
r5 = 1.6055092397
a5 = 102.2875376988
d5 = -139.0505800237
r6 = 1.4505339198
a6 = 118.5968228514
d6 = 73.3164625255
r7 = 1.7939415544
a7 = 105.1606322982
d7 = 115.2130868519
r8 = 1.4932730157
a8 = 112.3632012565
d8 = -12.8575710693
r9 = 1.0866209638
a9 = 106.0498560366
d9 = -116.2403495711
r10= 1.0897248583
a10= 106.2524931192

d10= 122.5676785489
r11= 1.0892520412
a11= 109.9657836383
d11= 62.4962398535
r12= 1.0862503872
a12= 105.6295098879
d12= -178.3609771686
r13= 1.0898070256
a13= 110.1634587241
d13= -59.2262253696
r14= 1.0896659037
a14= 109.1681841289
d14= 173.1829774789
r15= 1.0901180999
a15= 108.8704671880
d15= -67.6638946514
r16= 1.0892878937
a16= 110.0145490901
d16= 52.6755615836
r17= 2.4893506911
a17= 86.0854081748
d17= -76.7143005164
r18= 0.9793571499
a18= 88.9637963996
d18= -33.3244903596

*ACES2
!restart,symmetry=off
basis=6-31++G**,mult=2,spherical=on
ref=uhf,calc=ccsd
METHOD=TS
GEOM_OPT=FULL

*SIP
SIP_MX_SEGSIZE = 30
SIP_MX_OCC_SEGSIZE = 27
SIP_MX_VIRT_SEGSIZE = 27
COMPANY = 1 1 24 0
IOCOMPANY = 2 1 8 0
MAXMEM = 900
SIAL_PROGRAM = scf_uhf_isymm_diis10.sio
SIAL_PROGRAM = tran_uhf_ao_sv1.sio
SIAL_PROGRAM = ccsd_uhf_ao_sv1_diis5.sio
SIAL_PROGRAM = lambda_uhf_ao_sv1_diis5.sio
SIAL_PROGRAM = one_grad_uhf_ao_sv1_diis5.sio

SIAL_PROGRAM = two_grad_uhf_ao_sv1.sio

5.3 Vibrational frequency calculation for the water ion:

H2O(-1) in CC-PVQZ basis
O 0.0 0.0 0.1173
H 0.1 0.7572 -0.4692
H 0.0 -0.7572 -0.4692

∗ACES2(CALC=MBPT(2),BASIS=CC-PVQZ,REF=UHF,SPHERICAL=ON
CC_CONV=8,SCF_CONV=8
VIB_FINDIF=EXACT
charge=1,multiplicity=2
COORDINATES=CARTESIAN
SYMMETRY=OFF)

∗SIP
MAXMEM= 900
COMPANY = 1 1 3 0
IOCOMPANY = 2 1 1 0
SIP_MX_SEGSIZE = 29
SIP_MX_OCC_SEGSIZE = 5
SIP_MX_VIRT_SEGSIZE = 28
SIAL_PROGRAM = scf_uhf_init.sio
SIAL_PROGRAM = scf_uhf_finish.sio
SIAL_PROGRAM = hess_uhf_mp2_seg.sio

5.4 Hessian:

5.4.1 SCF(UHF) for water ion

H2O(-1) in CC-PVQZ basis
O 0.0 0.0 0.1173
H 0.1 0.7572 -0.4692
H 0.0 -0.7572 -0.4692

ACES2(CALC=SCF,BASIS=CC-PVQZ,REF=UHF,SPHERICAL=ON
CC_CONV=8,SCF_CONV=8
charge=1,multiplicity=2
COORDINATES=CARTESIAN
SYMMETRY=OFF)

*SIP
MAXMEM= 900
COMPANY = 1 1 3 0
IOCOMPANY = 2 1 1 0
SIP_MX_SEGSIZE = 29
SIP_MX_OCC_SEGSIZE = 5
SIP_MX_VIRT_SEGSIZE = 28
SIAL_PROGRAM = scf_uhf_init.sio
SIAL_PROGRAM = scf_uhf_finish.sio
SIAL_PROGRAM = hess_uhf_scf.sio

5.4.2 MP2(UHF)

H2O(-1) in CC-PVQZ basis
O 0.0 0.0 0.1173
H 0.1 0.7572 -0.4692
H 0.0 -0.7572 -0.4692

∗ACES2(CALC=MBPT(2),BASIS=CC-PVQZ,REF=UHF,SPHERICAL=ON
CC_CONV=8,SCF_CONV=8
charge=1,multiplicity=2
COORDINATES=CARTESIAN
SYMMETRY=OFF)

∗SIP
MAXMEM= 900
COMPANY = 1 1 3 0
IOCOMPANY = 2 1 1 0
SIP_MX_SEGSIZE = 29
SIP_MX_OCC_SEGSIZE = 5
SIP_MX_VIRT_SEGSIZE = 28
SIAL_PROGRAM = scf_uhf_init.sio
SIAL_PROGRAM = scf_uhf_finish.sio
SIAL_PROGRAM = hess_uhf_mp2_seg.sio

5.5 Single point energy CCSD(T) calculation for the Ar_6 cluster:

5.5.1 RHF

Ar6 IN aug-cc-pvtz basis
AR 2.5 2.5 0.0
AR -2.5 2.5 0.0
AR 2.5 -2.5 0.0
AR -2.5 -2.5 0.0
AR 0.0 0.0 2.5
AR 0.0 0.0 -2.5

∗ACES2
!restart,symmetry=off
basis=AUG-CC-PVTZ,SPHERICAL=ON # basis options
UNITS=BOHR
coordinate=cartesian
ref=rhf,calc=ccsd(t)

∗SIP
SIP_MX_SEGSIZE = 30
SIP_MX_OCC_SEGSIZE = 27
SIP_MX_VIRT_SEGSIZE = 27
COMPANY = 1 1 96 0
IOCOMPANY = 2 1 32 0
MAXMEM = 900
SIAL_PROGRAM = scf_rhf_isymm_diis10.sio
SIAL_PROGRAM = tran_rhf_ao_sv1.sio
SIAL_PROGRAM = ccsd_rhf_ao_sv1_diis5.sio
SIAL_PROGRAM = ccsdpt_rhf_pp.sio

5.5.2 UHF

Ar6 IN aug-cc-pvtz basis
AR 2.5 2.5 0.0
AR -2.5 2.5 0.0
AR 2.5 -2.5 0.0
AR -2.5 -2.5 0.0
AR 0.0 0.0 2.5
AR 0.0 0.0 -2.5

∗ACES2
!restart,symmetry=off
basis=AUG-CC-PVTZ,SPHERICAL=ON # basis options
UNITS=BOHR
coordinate=cartesian
ref=uhf,calc=ccsd(t)

∗SIP
SIP_MX_SEGSIZE = 30
SIP_MX_OCC_SEGSIZE = 27
SIP_MX_VIRT_SEGSIZE = 27
COMPANY = 1 1 96 0
IOCOMPANY = 2 1 32 0
MAXMEM = 900
SIAL_PROGRAM = scf_uhf_isymm_diis10.sio
SIAL_PROGRAM = tran_uhf_ao_sv1.sio
SIAL_PROGRAM = ccsd_uhf_ao_sv1_diis5.sio
SIAL_PROGRAM = ccsdpt_uhf_pp.sio

5.6 Single point CCSD gradient(UHF) using DROPMO on the CH_4 molecule:

```
CH4
H .431 -.762 -.739
H -.467 .762 -.426
H .778 .198 .739
H -.778 -.688 .587
C -.008 -.122 .040

*ACES2
!restart,symmetry=off
dropmo=1-2/30-35
basis=CC-PVDZ,SPHERICAL=OFF # basis options
coordinate=cartesian,cc_conv=8,scf_conv=8
FOCK=AO
ref=uhf,calc=ccsd

*SIP
SIP_MX_SEGSIZE = 20
SIP_MX_OCC_SEGSIZE = 13
SIP_MX_VIRT_SEGSIZE = 20
COMPANY = 1 1 3 0
IOCOMPANY = 2 1 1 0
MAXMEM = 900
SIAL_PROGRAM = scf_uhf_isymm_diis10.sio
SIAL_PROGRAM = tran_uhf_ao_sv1.sio
SIAL_PROGRAM = ccsd_uhf_dropmo.sio
SIAL_PROGRAM = lambda_uhf_dropmo.sio
SIAL_PROGRAM = expand_cc.sio
SIAL_PROGRAM = tran_uhf_expanded.sio
SIAL_PROGRAM = one_grad_uhf_ao_sv1_dropmo_diis5.sio
SIAL_PROGRAM = two_grad_uhf_ao_sv1_dropmo.sio
```

Any of the previous examples should also work if the *SIP sections are ommitted. For example, example 5.6 could be rewritten as follows:

```
CH4
H .431 -.762 -.739
H -.467 .762 -.426
H .778 .198 .739
H -.778 -.688 .587
C -.008 -.122 .040
```

*ACES2
!restart,symmetry=off
dropmo=1-2/30-35
basis=CC-PVDZ,SPHERICAL=OFF # basis options
coordinate=cartesian,cc_conv=8,scf_conv=8
ref=uhf,calc=ccsd

In this case, the program determines the segmentation parameters, the worker/server configuration, and even which .sio files to run, all based on the REF, CALC, and DROPMO parameters. Caution: Some type of calculations performed by the ACES II serial code are not yet supported in ACES III. An example of this is ECP. If the program cannot determine the type of calculation from the *ACES2 parameters, and no *SIP section is provided, an error message is printed and the program will abort.

If a user desires to run a different program than that determined by default, a *SIP section must be provided, and all .sio files must be specified, not just the one that is different from the default. For example, suppose someone wished to run a new integral transformation program, called test_tran.sio. It might seem that you could rewrite the previous example as follows:

CH4
H .431 -.762 -.739
H -.467 .762 -.426
H .778 .198 .739
H -.778 -.688 .587
C -.008 -.122 .040

*ACES2
!restart,symmetry=off
dropmo=1-2/30-35
basis=CC-PVDZ,SPHERICAL=OFF # basis options
coordinate=cartesian,cc_conv=8,scf_conv=8
ref=uhf,calc=ccsd

*SIP
SIAL_PROGRAM = test_tran.sio

However, there is no way to determine where in the sequence of .sio programs that test_tran.sio should be placed. All SIAL_PROGRAM parameters must be specified to correctly override the defaults. The correct ZMAT coding is as follows:

CH4
H .431 -.762 -.739
H -.467 .762 -.426
H .778 .198 .739

H -.778 -.688 .587
C -.008 -.122 .040

∗ACES2
!restart,symmetry=off
dropmo=1-2/30-35
basis=CC-PVDZ,SPHERICAL=OFF # basis options
coordinate=cartesian,cc_conv=8,scf_conv=8
ref=uhf,calc=ccsd

∗SIP
SIAL_PROGRAM = scf_uhf_isymm_diis10.sio
SIAL_PROGRAM = test_tran.sio
SIAL_PROGRAM = ccsd_uhf_dropmo.sio
SIAL_PROGRAM = lambda_uhf_dropmo.sio
SIAL_PROGRAM = expand_cc.sio
SIAL_PROGRAM = tran_uhf_expanded.sio
SIAL_PROGRAM = one_grad_uhf_ao_sv1_dropmo_diis5.sio
SIAL_PROGRAM = two_grad_uhf_ao_sv1_dropmo.sio