SAGE2_V6.20_AEROSOL_O3_NO2_H2O_BINARY Readme File

1.0 Introduction

This ‘readme’ file provides information on the SAGE2_V6.20_AEROSOL_O3_NO2_H2O_BINARY data set. This data set includes aerosol extinction profiles at 1020, 525, 453, and 385 nanometers, number density profiles of ozone and nitrogen dioxide, plus molecular density and mixing ratio profiles of water vapor. It also includes aerosol surface area density and effective radius profiles (Thomason, L.W., L.R. Poole, and T.R. Deshler, "A Global Climatology Of Stratospheric Aerosol Surface Area Density As Deduced From SAGE II: 1984-1994", J. Geophys. Res., 102, 8967-8976; 1997.), and retrieved molecular density for the middle atmosphere (40-75 km). All profiles are at 0.5-km vertical resolution. These products are nearly global in coverage, with data spanning from 80 North to 80 South.

For more information on the SAGE II Project and a detailed description of the SAGE II Version 6.20 processing, visit the following web site http://www-sage2.larc.nasa.gov

If users have questions, please contact the Langley ASDC Science, Users and Data Services Office at:

Atmospheric Sciences Data Center
Science, Users and Data Services Office
Mail Stop 157D
2 South Wright Street
NASA Langley Research Center
Hampton, Virginia 23681-2199
U.S.A.
E-mail: larc@eos.nasa.gov
Phone: (757) 864-8656
FAX: (757) 864-8807
URL: http://eosweb.larc.nasa.gov

2.0 Data Set Description

The Stratospheric Aerosol and Gas Experiment (SAGE) II was launched aboard the Earth Radiation Budget Satellite in October 1984 during STS-41-G. The instrument continues to provide high quality measurements of ozone, nitrogen dioxide, water vapor, and multi-wavelength aerosol extinction from the mid-troposphere to as high as the lower mesosphere. The extended lifetime of this instrument and its measurement stability enhance its value in quantifying long-term trends and variability in its species ensemble. This data set spans the period October 1984 through the present. It contains profiles of aerosol extinction at 1020, 525, 453, and 385 nanometers(nm) and number density profiles of ozone, nitrogen dioxide, and molecular density, water vapor mixing ratio, and aerosol surface area and effective radius at a vertical resolution of 0.5km. It also includes retrieved molecular density from 40-75km on a 0.5km grid.

2.1 Instrument Description

The SAGE II instrument is a seven-channel Sun photometer using the Cassegrainian-configured telescope, holographic grating, and seven silicon photodiodes, some with interference filters, to define the seven spectral channel band passes. Solar radiation is reflected off a pitch mirror into the telescope with an image of the Sun formed at the focal plane. The instrument's instantaneous field-of-view, defined by an aperture in the focal plane, is a

0.5-by-2.5 arc-minute slit that produces a vertical resolution at the tangent point on the Earth's horizon of about 0.5 kilometers. Radiation passing through the aperture is transferred to the spectrometer section of the instrument containing the holographic grating and seven separate detector systems. The holographic grating disperses the incoming radiation into the various spectral regions centered at the 1020, 940, 600, 525, 453, 448, and 385 nanometer wavelengths. Slits on the Rowland circle of the grating define the spectral band pass of the seven spectral channels. The spectrometer system is inside the azimuth gimbal to allow the instrument to be pointed at the Sun without image rotation. The azimuth gimbal can be rotated over 370 degrees so that measurements can be made at any azimuth angle.

The operation of the instrument during each sunrise and sunset measurement is totally automatic. Prior to each sunrise or sunset encounter, the instrument is rotated in azimuth to its predicted solar acquisition position. When the Sun's intensity reaches a level of one percent of maximum in the Sun sensor, the instrument adjusts its azimuth position to lock onto the radiometric center of the Sun to within +/-45 arc-seconds and then begins acquisition of the Sun by rotating its pitch mirror in a predetermined direction depending on whether it is a sunrise or a sunset. When the Sun is acquired, the pitch mirror rotates back and forth across the Sun at a rate of about 15 arc-minutes per second. The radiometric channel data are sampled at a rate of 64 samples per second per channel, digitized to 12-bit resolution, and recorded for later transmission back to Earth.

Version 6.2
A new version of the SAGE II data products has been released. The primary change to the algorithm dealt with the improvement in the water vapor product. The SAGE II V6.1 data has not been publicly available after mid-2000 due to an altitude registration problem. This has been tracked down and corrected. An error in the interpolation of the NCEP Met data used to remove Rayleigh scattering from the transmission profiles has also been corrected. For a detailed description of all algorithm modifications, see the following SAGE II web site:
http://www-sage2.larc.nasa.gov/data/v6_data

2.3 Data Quality and Known Deficiencies
For a detailed description of all algorithm modifications, see the following SAGE II web site: http://www-sage2.larc.nasa.gov/data/v6_data

2.4 Science contact
Larry W. Thomason
NASA Langley Research Center
Atmospheric Sciences Division
Mail Stop 475
Hampton, VA 23681-2199
Phone: (757) 864-6842
FAX: (757) 864-2671
E-mail: l.w.thomason@nasa.gov

Joseph M. Zawodny
NASA Langley Research Center

Atmospheric Sciences Division
Mail Stop 475
Hampton, VA 23681-2199
Phone: (757) 864-2681
FAX: (757) 864-2671
E-mail: j.m.zawodny@nasa.gov

3.1 file Naming convention

SAGE II Version 6.20 files are named according to the following convention:
SAGE_II_INDEX_YYYYMM.6.20
SAGE_II_SPEC_YYYYMM.6.20
Where YYYY is the 4 digit year and MM is the 2 digit month.
The "INDEX" file contains the revision information and the "SPEC" file contains the Species profiles

4.0 Science Parameter Information

4.1 Altitude Range for Species

Species Range (km)
Ozone 5-60
NO2 15-60
Aerosol 1-45
Water Vapor MSL-40

4.2 Data File Contents

The following abbreviations have been used in the description of the file contents.
alt - altitude Lon - longitude
Arr - array Max - maximum
Char - character string Met - Meteorology
Ele - Element Min - minimum
ext - extinction NO2 - nitrogen dioxide
H20 - water vapor O3 - ozone
Int - Integer sr - sunrise
LaRC - Langley Research Center ss - sunset
Lat - latitude

4.2.1 Index File Contents

Revision Info

Field	Type	Description
Num_Prof	4-byte Int	Number of profiles (records) in file
Met_Rev_Date	4-byte Int	LaRC Met Model Rev Date (yyyymmdd)
Driver_Rev	8-byte Char	LaRC Driver version (eg. 6.20)
Transmission_Rev	8-byte Cha	LaRC Transmission version
Inversion_Rev	8-byte Char	LaRC Inversion version
Spectroscopy_Rev	8-byte Char	LaRC Inversion version
Eph_File_Name	32-byte Char	Ephemeris file name
Met_File_Name	32-byte Char	Met file name
Ref_File_Name	32-byte Char	Refraction file name
Trans_File_Name	32 -byte Char	Transmission file name
Spec_File_Name	32-byte Char	Species profile file name
FillVal	4-byte Real	Fill value

Altitude grid and range info

Grid_Size	4-byte Real	Altitude Grid spacing
Alt_Grid	4-byte Real Arr w200Ele	Geometric Alt
Alt_Mid_Atm	4-byte Real Arr w/70Ele	Geometric Alt for Dens_Mid_Atm
Range_Trans	4-byte Real Arr w/ 2 Ele	Transmission Min & Max alt
Range_O3	4-byte Real Arr w/ 2 Ele	Ozone Density Min & Max alt
Range_NO2	4-byte Real Arr w/ 2 Ele	NO2 Density Min & Max alt
Range_H2O	4-byte Real Arr w/ 2 Ele	H2O Density Min & Max alt
Range_Ext	4-byte Real Arr w/ 2 Ele	Extinction Min & Max alt
Range_Density	4-byte Real Arr w/ 2 Ele	Density Min & Max alt
Range_Surface	4-byte Real Arr w/ 2 Ele	Surface Area Min & Max alt

Event Specific Info useful for data subsetting:

YYYYMMDD	4-byte Int Arr w/930 Ele	Event Date (yyyymmdd) at 30 km
event_num	4-byte Int Arr w/930 Ele	The event number
HHMMSS	4-byte Int Arr w/930 Ele	Event Time (hhmmss) at 30 km
Day_Frac	4-byte Real Arr w/930 Ele	Time of Year (ddd.fraction)
Lat	4-byte Real Arr w/930Ele	Sub-tangent Lat at 30km
Lon	4-byte Real Arr w/930 Ele	Sub-tangent Lon at 30km
Beta	4-byte Real Arr w/930 Ele	Spacecraft Beta angle (degree
Duration	4-byte Real Arr w/930 Ele	Duration of event (seconds)
Type_Sat	2-byte Int Arr w/930 Ele	Instrument Event Type, 0=sr, 1=ss)
Type_Tan	2-byte Int Arr w/ 930 Ele	Event Type, Local (0=sr,1=ss)

Process Tracking Flag info:

Processing Success:

Dropped	4-byte Int Arr w/ 930 Ele	Value is non-zero if event is dropped
InfVec	4-byte Int Arr w/ 930 Ele	32 bits describing the event processing
Ephemeris:		
Eph_Cre_Date	4-byte Int Arr w/ 930 Ele	Record creation date (yyyymmdd)
Eph_Cre_Time	4-byte Int Arr w/ 930 Ele	Record creation time (hhmmss)
Met:		
Met_Cre_Date	4-byte Int Arr w/ 930 Ele	Record creation date (yyyymmdd)
Met_Cre_Time	4-byte Int Arr w/ 930 Ele	Record creation time (hhmmss)
Refraction:		
Ref_Cre_Date	4-byte Int Arr w/ 930 Ele	Record creation date (yyyymmdd)
Ref_Cre_Time	4-byte Int Arr w/ 930 Ele	Record creation time (hhmmss)
Transmission:		
TRANS_Cre_Date	4-byte Int Arr w/ 930 Ele	Record creation date (yyyymmdd)
TRANS_Cre_Time	4-byte Int Arr w/ 930 Ele	Record creation time (hhmmss)
Inversion:		
SPECIES_Cre_Date	4-byte Int Arr w/ 930 Ele	Record creation date (yyyymmdd)
SPECIES_Cre_Time	4-byte Int Arr w/ 930 Ele	Record creation time (hhmmss)

4.2.2 Species File Contents

Field	Type	Description
Tan_Alt	4-byte Real Arr w/ 8 Ele	Center-of-Sun Tangent Alt (km)
Tan_Lat	4-byte Real Arr w/ 8 Ele	Center-of-Sun Lat (deg)
Tan_Lon	4-byte Real Arr w/ 8 Ele	Center-of-Sun Lon (de
NMC_Pres	4-byte Real Arr w/ 14 Ele	Pressure (mb) (0.5-70km)
NMC_Temp	4-byte Real Arr w/ 140 Ele	Temperature (K), (0.5-70km)
NMC_Dens	4-byte Real Arr w/ 140 Ele	Density (molecules/cm3) (.5-

		70km)
NMC_Dens_Err	2-byte Int Arr w/ 140 Ele	Density Uncertainty(%x100)
Trop_Height	4-byte Real Arr w/ 1 Ele	Tropopause height in km
Wavelength	4-byte Real Arr w/ 7 Ele	Channel wavelengths
O3	4-byte Real Arr w/ 140Ele	O3 number density (cm-3)
NO2	4-byte Real Arr w/ 100Ele	NO2 number density (cm-3)
H2O	4-byte Real Arr w/ 100Ele	H2O number density (ppp)
Ext386	4-byte Real Arr w/ 80 Ele	386 nm aerosol extinction (1/km)
Ext452	4-byte Real Arr w/ 80 Ele	452 nm aerosol extinction (1/km)
Ext525	4-byte Real Arr w/ 80 Ele	525 nm aerosol extinction (1/km)
Ext1020	4-byte Real Arr w/ 80 Ele	1020 nm aerosol extinction (1/km)
Density	4-byte Real Arr w/ 140Ele	Molecular density (1/cm^3)
SurfDen	4-byte Real Arr w/ 80 Ele	Aerosol surface area density (micrometer^2/cm^3)
Radius	4-byte Real Arr w/ 80 Ele	Aerosol effective radius (micrometer)
Dens_Mid_Atm	4-byte Real Arr w/ 70 Ele	Middle atmosphere retrieved density(1/cm^3)
O3_Err uncertainty	2-byte Int Arr w/ 140 Ele	O3 number density (%x100)
NO2_Err uncertainty	2-byte Int Arr w/ 100 Ele	NO2 number density (%x100)
H2O_Err uncertainty	2-byte Int Arr w/ 100 Ele	H2O number density (%x100
Ext386_Err	2-byte Int Arr w/ 80 Ele	386 nm aerosol ext. uncertainty %x100)
Ext452_Err	2-byte Int Arr w/ 80 Ele	452 nm aerosol ext. uncertainty (%x100)
Ext525_Err	2-byte Int Arr w/ 80 Ele	525 nm aerosol ext. uncertainty (%x100)
Ext1020_Err	2-byte Int Arr w/ 80 Ele	1020 nm aerosol ext. uncertainty (%x100)
Density_Err	2-byte Int Arr w/ 140 Ele	Density uncertainty (%x100)
SurfDen_Err	2-byte Int Arr w/ 80 Ele	Aerosol surface area density uncertainty(%x100)
Radius_Err	2-byte Int Arr w/ 80 Ele	Aerosol effective radius uncertainty (%x100)
Dens_Mid_Atm_Err	2-byte Int Arr w/70 Ele	Middle atmosphere density uncertainty (%x100)
InfVec	2-byte Int Arr w/ 140 Ele	Bit-wise quality flags

5.0 Description of Sample Read Software

An Interactive Data Language (IDL) program is provided for reading the SAGE II Version 6.20 data files. Instructions and Fortran 90 modules that may be used to read the data are also available. The SAGE II team has provided both programs. The IDL program allows users to display graphically the data. This IDL package was designed for the "experienced" IDL user. There is a second IDL piece of code that allows the user to read one "event" or the entire file. This code converts the entire file into ASCII. Once in ASCII, the user is able to port the output to his/her favorite software. There is a Fortran 90 package available. This package contains modules to be used to read the data. This is NOT a complete sample read software package.

6.0 Implementation of the Sample Read Software

To run the IDL package, please refer to the "README" included in the package. To run the IDL code, sagetext_v6.20.pro, which converts the data from binary into ASCII, follow these instructions:

From the command line, type the following commands:

```
commandline> idl
IDL> .compile sagetext_v6.20.pro
% Compiled module: GETINDEXNAME.
% Compiled module: SAGETEXT.
IDL>sagetext
Please call sagetext using the filename of the file to read. Also provide the
record number to read, or answer the prompts. You may also use the output
keyword to specify the output file.
Usage examples:
1. Writes record 100 of SAGE_II_SPEC_199804.6.20 to sage.dat
sagetext,'SAGE_II_SPEC_199804.6.20',100
2. If the record number is left off, you will be prompted.
sagetext,'SAGE_II_SPEC_199804.6.20
3. In this example, output is written to 'output.dat'
sagetext,'SAGE_II_SPEC_199804.6.20',100,output='output.dat'
IDL>sagetext,'SAGE_II_SPEC_199804.6.20
Enter the starting record number (or Enter for first record of the file):
0
Enter the ending record number (or Enter for the last record of the file):
930
% Compiled module: READSTRUCTS.
% Compiled module: REVERSE.
% Compiled module: GETSTRUCTINFO.
% Compiled module: INDEXINFO_60D.
% Compiled module: INDEXINFO_610.
% Compiled module: TRANSINFO_600.
% Compiled module: REFRACTINFO_60D.
% Compiled module: METINFO.
% Compiled module: METINFO_610.
% Compiled module: EPHINFO_Y2K.
% Compiled module: SPECINFO_600.
% Compiled module: SPECINFO_610.
% Compiled module: SWAP_ENDIAN.
The specified record number are outside the range for this file.
Please try again with records between 0 and 559.
IDL>sagetext,'SAGE_II_SPEC_199804.6.20
Enter the starting record number (or Enter for first record of the file):
0
Enter the ending record number (or Enter for the last record of the file):
559
IDL>exit
```

Look into your working directory and you should have a file called sage.dat.

8.2 Bit Flag Meaning

Bit flags are used in both the index and species files to convey significant information about the inversion process. Index bit flags refer to an entire event while species bit flags are both species and altitude dependent. In general, severity increases with increasing value. Some flags are primarily kept as keys to the developers. A set bit flag does not necessarily indicate that an event should be considered flawed. The data set has been designed to indicate data validity through uncertainty estimates and, in the case of serious failure, missing data.

8.2.1 Index File Bit Flags

Name	BitNumber	Meaning
pmc_present	0	Polar Mesospheric Cloud (PMC) found in profile between 70 and 90 km
h2o_zero_found	1	Zero or negative mixing ratio inferred
h2o_slow_convergence	2	Water vapor retrieval required more than 20 iterations
h2o_ega_failure	3	Emissivity Curve-of-Growth Approximation (EGA) tool failure
default_nmc_temp_errors	4	A default uncertainty profile was used because no NCEP provided uncertainty were available
ch2_aero_model_A:	5	post-Chichon model for ch2 clearing
ch2_aero_model_B:	6	Pinatubo model for ch2 clearing
ch2_new_wavelength:	7	ch2 uses new filter function
incomplete_nmc_data	8	One or more mandatory levels were missing from NCEP data
mirror_model	15	Mirror reflectivity is modeled; insufficient high altitude data
twomey_non_conv_rayleigh	19	Twomey-Chahine (T-C) inversion routine failure for Rayleigh retrieval
twomey_non_conv_386_Aero	20	T-C inversion routine failure for 386 nm aerosol extinction retrieval
twomey_non_conv_452_Aero	21	T-C inversion routine failure for 452 nm aerosol extinction retrieval
twomey_non_conv_525_Aero	22	T-C inversion routine failure for 525 nm aerosol extinction retrieval
twomey_non_conv_1020_Aero	23	T-C inversion routine failure for 1020 nm aerosol extinction retrieval
twomey_non_conv_NO2	24	T-C inversion routine failure for NO2 retrieval
twomey_non_conv_ozone	25	T-C inversion routine failure for ozone retrieval
no_shock_correction	30	No correction for the electrical transient was

performed; usually a short event with too few extraterrestrial solar irradiance available

8.2.2 Species File Bit Flags

Name Bit	Number	Meaning
separation_method	0-2	Method used to separate between ozone, NO2, and aerosol
one_chan_aerosol_corr	3	Aerosol correction based on 1020nm aerosol only
no_935_aerosol_corr	4	No aerosol correction in the water vapor retrieval (based on the 935nm channel)
Large_1020_OD	5	1020-nm aerosol slant path optical depth is large at or above this level and may influence ozone retrieval
NO2_Extrap	6	Relevant to water vapor retrieval NO2 is extrapolated at this altitude based on a vertical profile that terminated at a higher altitude
Water_vapor_ratio:	7-10	Ratio of the water vapor slant path optical depth to the total optical depth in 940nm
filler_bit_8	8	
filler_bit_9	9	
filler_bit_10	10	
Cloud_Bit_1	11	Cloud bit: on is test successful, either aerosol (if 12 is off) or cloud (12 on), off is indeterminate or not tested
Cloud_Bit_2	12	Cloud bit: on is indeterminate (if 11 is off) or cloud (if 11 is on). off is not tested or aerosol
No_H2O_Corr	13	Ozone not corrected for water vapor
In_Troposphere	14	Altitude is below the NCEP-provided tropopause altitude

8.2.3 Species Separation Method Bit Flags

Name	Bit Number	Meaning
no_aerosol_method	0	Four channels used (3-6); Ozone, NO2, no aerosol inferred
trans_no_aero_to_five_chan	1	Transition
standard_method	2	Five channels used (1,3-6); Ozone, NO2, and aerosol (3)
trans_five_chan_to_low	3	Transition
four_chan_method	4	Four channels used (1,3-5); Ozone, aerosol (3)
trans_four_chan_to_three_chan	5	Transition
three_chan_method	6	Three channels used (1,3,4); Ozone, aerosol (2)
extension_method	7	Channel 1 only, aerosol (1)

READING SAGE II VERSION 6.20 FORMAT DATA USING FORTRAN 90.

Since different fortran compilers handle binary unformatted data in different ways, it is not possible to provide a reader for the SAGE II data that will work with all hardware systems and all compilers. But a number of general comments can be made. The Fortran 90 modules and sample output are available from the SAGE II data table (http://eosweb.larc.nasa.gov/PRODOCS/sage2/table_sage2.html).

The files are unformatted (native format) binary files written on a DEC Alpha by a program compiled using the DIGITAL Fortran 90 V5.2-705 compiler. The data consists of two and four byte little endian integers and single and double precision little endian IEEE floating point data. The following code fragments should be sufficient for reading the SAGE II data files on a DEC or PC system.

```
use specinfo   !the specinfo.f90 file is provided
implicit none
integer :: unit=201,lrspec,recnum=1   !this unit number is just an example
character (len=80) :: file = 'SAGE_II_SPEC_198410.6.20'   !example filename
type(speciesinfo) spec

!open the species structure file
inquire(iolength=lrspec) spec
open(unit=unit,file=file,action='read', &
form='unformatted',access='direct',recl=lrspec)

!read the first record
read(unit=unit,rec=recnum) spec

!close the file
close(unit)
```

A text file is provided which contains the contents of the first two records of the file SAGE_II_SPEC_198410.6.20. Also provided is a f90 subroutine called formatspec (in the file formatspec.f90), which can be used to print out one species record at a time with a standard format. The user can use these two file to test that the species structure files are being read correctly. Just call formatspec twice, once for each of the first two records of SAGE_II_SPEC_198410.6.20. The arguments are the file unit number to write to, and the entire record represented as a speciesinfo structure ("spec" in the above code fragment). The resulting text file should be exactly identical to SAGE_II_SPEC_198410.6.20.txt provided here. If it isn't, there are several possible causes...

BYTE SWAPPING
The SAGE II structure files contain little-endian data, that is, multiple byte data are stored with the least significant byte first. Certain hardware systems, such as Sun workstations, store data with the most significant byte first. Users of such systems will need to perform byte swapping on SAGE II data in order for it to be read correctly.

Some compilers provide various methods for performing byte-swapping automatically, although no such provision is given in the Fortran 90 standard. Look for a non-standard keyword to the open statement, such as "convert". Also look for ways to specify that byte-swapping should be performed using compiler options, run-time options, or system environment variables. Detailed information on these methods will be found in the documentation accompanying the fortran 90 compiler.

Note that some of these methods may not work for data in user-defined types. If that is the case, the read statement will have to be modified to read in each component of the species structure separately. That is, the statement

```
read(unit=unit,rec=recnum) spec
```

will have to be modified as

```
read(unit=unit,rec=recnum) spec%tan_alt,spec%tan_lat,spec%tan_lon, &
                           spec%nmc_pres,spec%nmc_temp,spec%nmc_dens, &
{etc.}
```

If the data still are not being read in correctly, proceed to the next section.

If no method of performing automatic byte-swapping is found in the compiler documentation, users will have to manually perform byte-swapping in the application program. Four byte integers and reals should have all four bytes reversed, so that what is read in as (byte-1)(byte-2)(byte-3)(byte-4) is changed to (byte-4)(byte-3)(byte-2)(byte-1). Two byte integers will simply have the even and odd bytes interchanged. One way to accomplish such swapping is using the fortran 90 intrinsic routine mvbits, as in the following example.

```
!four-byte swap
   bitperbyte = bit_size(tan_alt(1))/4
   btbd1 = 0
   btbd2 = bitperbyte
   btbd3 = bitperbyte*2
   btbd4 = bitperbyte*3

   call mvbits(tan_alt,btbd1,bitperbyte,output%tan_alt,btbd4)
   call mvbits(tan_alt,btbd2,bitperbyte,output%tan_alt,btbd3)
   call mvbits(tan_alt,btbd3,bitperbyte,output%tan_alt,btbd2)
   call mvbits(tan_alt,btbd4,bitperbyte,output%tan_alt,btbd1)
```

Note, however that mvbits accepts only integer arguments. One way around this limitation is to read in all the four-byte real data as four-byte integer data, swap the bytes, and re-write the data to a temporary file. The temporary file can then be read in the normal way.

TROUBLE-SHOOTING OTHER POSSIBLE PROBLEMS

Because there is nothing in the fortran 90 standard specifying how different compilers treat binary unformatted data, it is possible that other irregularities can cause difficulties in reading SAGE II data.

A user encountering such difficulties should first examine the expected record size. The following statement was included in the example code fragment given previously:

```
type(speciesinfo) spec
```

```
inquire(iolength=lrspec) spec
```

The result, lrspec, should be a multiple of 2137. If it is not, it may be true that the size of individual components is different from expected. The user can perform inquiry statements on arrays of integers of kind=2 and kind=4 and reals of kind=4 to determine if the following are true:

1. the size of integer(kind=4) should be the same as real(kind=4)
2. the size of integer(kind=4) should be twice the size of integer(kind=2)
3. the size of a variable of type speciesinfo should be 1482 times the size of a (kind=4) variable plus 1310 times the size of a (kind=2)

Note: the inquiry statement should be done on arrays rather than on scalars. The reason for this is that the "iolength" of a single (kind=2) integer may be the same as a single (kind=4) integer in order to ensure that data are aligned on natural byte boundaries. However, an array of 100 (kind=2) integers would be expected to be half the size of an array of 100 (kind=4) integers. Since two byte integers appear in the species structure only in arrays of even length, the iolength of an array of values is more relevant.

On some compilers, some of the given conditions may be false. For instance, the WorkShop f90 compiler that is used on many Sun systems reads and writes all integers as four bytes. In other words, the kind specification (or the "n" in "integer*n") controls only the precision and range of the integer, not the amount of memory used for the variable. For this reason, it is impossible to read two-byte integers in a straightforward way. One option for users of such compilers is to create a dummy structure which represents arrays of (kind=2) integers as arrays of (kind=4) integers of half the length. Then the intrinsic routine mvbits can be used to split the bytes apart.

The third condition specifies that the size of the speciesinfo record be equal to the sum of the sizes of its parts. If this is not true, then the compiler pads the structure in a way different from the way it was written. This is not expected to happen, because the speciesinfo type is already defined in such a way that the data are aligned on natural byte boundaries. A natural boundary is a multiple of the size of the data item. Alignment on natural byte boundaries improves the efficiency of data storage and many compilers ensure this alignment by padding misaligned data. If for some reason padding similar to this is being performed, the user can examine the documentation accompanying the compiler for a way to turn it off. Reading the data one component at a time, as described in the section on Byte-Swapping, would also most likely solve this problem.