

# speciessetup

November 24, 2009  
13:41

## Contents

<b>1</b>	<b>Routines used to set up species</b>	<b>1</b>
<b>2</b>	<b>References</b>	<b>6</b>
<b>3</b>	<b>INDEX</b>	<b>7</b>

# 1 Routines used to set up species

```
$Id: speciessetup.web,v 1.23 2000/04/01 17:34:44 dstotler Exp $
```

Inputs for species are contained in the file with the symbolic name `species_infile` identified in the `degas2.in` file. Each species is an aggregate of the elements and is represented by two lines in this file. The first line provides the full name of the species. The second line is a blank-delimited sequence of entries

```
symbol count-1 element-1 count-2 element-2 ...
```

An isotopic equivalence of this species to a previously-defined generic one is indicated with a delimiter “=” followed by the symbol of the generic species.

For example,

```
geometry
0 1 0
electron
e 1 e
atomic hydrogen
H 1 H
molecular hydrogen
H2 2 H
hydrogen ion
H+ 1 H -1 e
atomic deuterium
D 1 D = H
deuterium ion
D+ 1 D -1 e = H+
molecular hydrogen-deuterium
HD 1 H 1 D = H2
molecular deuterium
D2 2 D = H2
molecular hydrogen ion
H2+ 2 H -1 e
molecular hydrogen-deterium ion
HD+ 1 H 1 D -1 e = H2+
molecular deuterium ion
D2+ 2 D -1 e = H2+
helium
He
```

These routines are called by the program `datasetup`. `datasetup` reads the `species_infile`, translates the information into DEGAS 2 data, and generates the netCDF file corresponding to the symbolic name `speciesfile`.

```
"speciessetup.f" 1 ≡
@m FILE 'speciessetup.web'
```

The unnamed module.

"`speciessetup.f`" 1.1 ≡  
⟨Functions and Subroutines 1.2⟩

Read in data from `species_infile`.

⟨Functions and Subroutines 1.2⟩ ≡

```

subroutine read_species
  implicit none f77
  sp_common    // Common
  el_common
  rf_common
  implicit none f90
  st_decls
  ⟨Memory allocation interface 0⟩

  external get_multiplicity    // External
  integer get_multiplicity

  integer length, p, b, e, i    // Local
  character*LINELEN line
  character*FILELEN tempfile    // local

  sp_num = 0
  species_version = 'unknown'
  tempfile = filenames_array_species_infile
  assert(tempfile ≠ char_undef)
  open(unit = disk_in, file = tempfile, form = 'formatted', status = 'old')
  assert(read_string(disk_in, line, length))
  assert(length ≤ len(line))
  species_version = line(: length)
loop1: continue
  if (¬read_string(disk_in, line, length))
    goto eof    /* Add next species */
  sp_num = sp_num + 1
  var_realloca(species_name)
  var_realloca(species_sy)
  var_realloca(species_z)
  var_realloca(species_m)
  var_realloca(species_ncomp)
  var_realloca(species_generic)
  var_realloca(species_multiplicity)
  var_realloca(species_el)
  var_realloca(species_count)    /* First line contains species name */
  length = parse_string(line(: length))
  species_name_sp_num = line(: length)
    /* Second line contains symbol and breakdown into "elements" */
  assert(read_string(disk_in, line, length))
  length = parse_string(line(: length))    /* Extract species "symbol" from this line */
  p = 0
  assert(next_token(line, b, e, p))
  sp_sy(sp_num) = read_text(line(b : e))
  sp_ncomp(sp_num) = 0
  sp_z(sp_num) = 0
  sp_m(sp_num) = zero
  do i = 1, sp_ncomp_max
    sp_el(sp_num, i) = int_uninit
    sp_count(sp_num, i) = int_uninit
  end do    /* Loop over individual "elements" */

```

```

loop2: continue
  if ( $\neg$ next_token(line, b, e, p)) then // Generic species
    sp_generic(sp_num) = sp_num
    sp_multiplicity(sp_num) = get_multiplicity(sp_num)
    goto loop1
  else if (line(b : e)  $\equiv$  '=') then // Non-generic species
    assert(next_token(line, b, e, p))
    sp_generic(sp_num) = sp_lookup(line(b : e))
    sp_multiplicity(sp_num) = get_multiplicity(sp_num)
    goto loop1
  else // Not done reading elements
    /* Increment number of "components"  $\leftrightarrow$  "elements" */
    sp_ncomp(sp_num) = sp_ncomp(sp_num) + 1
    assert(sp_ncomp(sp_num) > 0  $\wedge$  sp_ncomp(sp_num)  $\leq$  sp_ncomp_max)
    /* Extract number of this "component" in the species */
    sp_count(sp_num, sp_ncomp(sp_num)) = read_integer(line(b : e))
    /* Get the symbol representing the "element"; store the index used to represent it */
    assert(next_token(line, b, e, p))
    sp_el(sp_num, sp_ncomp(sp_num)) = el_lookup(line(b : e))
    assert(el_check(sp_el(sp_num, sp_ncomp(sp_num))))
    /* Compute net charge of species. This is = 0 by default; nonzero values are arrived at by
    incorporating the appropriate number (positive or negative) of electrons. */
    if (el_z(sp_el(sp_num, sp_ncomp(sp_num)))  $\equiv$  -1) then // Only the electron is charged
      sp_z(sp_num) = sp_z(sp_num) + sp_count(sp_num, sp_ncomp(sp_num)) * el_z(sp_el(sp_num,
        sp_ncomp(sp_num)))
    end if /* Accumulate species mass based on "count" and the "element" mass */
    sp_m(sp_num) = sp_m(sp_num) + sp_count(sp_num,
      sp_ncomp(sp_num)) * atomic_mass_unit * el_m(sp_el(sp_num, sp_ncomp(sp_num)))
  end if
  goto loop2
eof: continue
close(unit = diskio)
var_reallocb(species_name)
var_reallocb(species_sy)
var_reallocb(species_z)
var_reallocb(species_m)
var_reallocb(species_ncomp)
var_reallocb(species_generic)
var_reallocb(species_multiplicity)
var_reallocb(species_el)
var_reallocb(species_count)

return
end

```

See also sections 1.3 and 1.4.

This code is used in section 1.1.

Write out data into netcdf file `species.nc`

⟨Functions and Subroutines 1.2⟩ +=

```

subroutine nc_write_species
  implicit_none_f77
  sp_common    // Common
  rf_common
  implicit_none_f90
  nc_decls    // Local
  integer fileid

  sp_ncdecl
  st_decls
  character*LINELEN description, program_version
  character*FILELEN tempfile    // local

  program_version = '$Id: speciessetup.web,v1.23,2000/04/01,17:34:44,dstotler,Exp$'

  tempfile = filenames_array_speciesfile
  assert(tempfile ≠ char_undef)
  fileid = nccreate(tempfile, NC_CLOBBER, nc_stat)

  description = 'Data_for_species_in_degas2'
  call ncattputc(fileid, NC_GLOBAL, 'description', NC_CHAR, string_length(description),
    description, nc_stat)

  call ncattputc(fileid, NC_GLOBAL, 'data_version', NC_CHAR, string_length(species_version),
    species_version, nc_stat)

  call ncattputc(fileid, NC_GLOBAL, 'program_version', NC_CHAR,
    string_length(program_version), program_version, nc_stat)

  sp_ncdef(fileid)
  call ncendef(fileid, nc_stat)
  sp_ncwrite(fileid)
  call ncclose(fileid, nc_stat)

  return
end

```

Compute multiplicity of a given species. This is a temporary version which treats hydrogen as the only species with multiple isotopes. A generalization may be possible or desirable if the concept of a “generic element” analogous to the present “genericspecies” is introduced.

⟨Functions and Subroutines 1.2⟩ +≡

```

function get_multiplicity(sp_dummy(species))
  implicit_none_f77
  sp_common
  el_common
  implicit_none_f90

  integer get_multiplicity    // Function

  sp_decl(species)    // Input

  integer i, num_h, num_d, num_t, num_tot    // Local
  integer fact0:6
  data fact/1, 1, 2, 6, 24, 120, 720/

  num_h = 0
  num_d = 0
  num_t = 0
  do i = 1, sp_ncomp(sp_num)
    if (el_sy(sp_el(sp_num, i)) ≡ 'H') then
      num_h = sp_count(sp_num, i)
    else if (el_sy(sp_el(sp_num, i)) ≡ 'D') then
      num_d = sp_count(sp_num, i)
    else if (el_sy(sp_el(sp_num, i)) ≡ 'T') then
      num_t = sp_count(sp_num, i)
    end if
  end do
  num_tot = num_h + num_d + num_t
  assert(num_tot ≤ 6)    // Limit of factorial table
  get_multiplicity = factnum_tot / (factnum_h * factnum_d * factnum_t)

  return
end

```

## 2 References

### 3 INDEX

*assert*: 1.2, 1.3, 1.4.  
*atomic\_mass\_unit*: 1.2.

*b*: [1.2](#).

*char\_undef*: 1.2, 1.3.

*datasetup*: 1.  
*description*: [1.3](#).  
*diskin*: 1.2.

*e*: [1.2](#).  
*el\_check*: 1.2.  
*el\_common*: 1.2, 1.4.  
*el\_lookup*: 1.2.  
*el\_m*: 1.2.  
*el\_sy*: 1.4.  
*el\_z*: 1.2.  
*eof*: 1.2.

*fact*: [1.4](#).  
*file*: 1.2.  
*FILE*: [1](#).  
*fileid*: [1.3](#).  
*FILELEN*: 1.2, 1.3.  
*filenames\_array*: 1.2, 1.3.  
*form*: 1.2.

*get\_multiplicity*: [1.2](#), [1.4](#).

*i*: [1.2](#), [1.4](#).  
*implicit\_none\_f77*: 1.2, 1.3, 1.4.  
*implicit\_none\_f90*: 1.2, 1.3, 1.4.  
*int\_uninit*: 1.2.

*len*: 1.2.  
*length*: [1.2](#).  
*line*: [1.2](#).  
*LINELEN*: 1.2, 1.3.  
*loop1*: 1.2.  
*loop2*: 1.2.

*NC\_CHAR*: 1.3.  
*NC\_CLOBBER*: 1.3.  
*nc\_decls*: 1.3.  
*NC\_GLOBAL*: 1.3.  
*nc\_stat*: 1.3.  
*nc\_write\_species*: [1.3](#).  
*ncattputc*: 1.3.  
*ncclose*: 1.3.  
*nccreate*: 1.3.  
*ncendef*: 1.3.  
*next\_token*: 1.2.  
*num\_d*: [1.4](#).

*num\_h*: [1.4](#).  
*num\_t*: [1.4](#).  
*num\_tot*: [1.4](#).

*p*: [1.2](#).  
*parse\_string*: 1.2.  
*program\_version*: [1.3](#).

*read\_integer*: 1.2.  
*read\_species*: [1.2](#).  
*read\_string*: 1.2.  
*read\_text*: 1.2.  
*rf\_common*: 1.2, 1.3.

*sp\_common*: 1.2, 1.3, 1.4.  
*sp\_count*: 1.2, 1.4.  
*sp\_decl*: 1.4.  
*sp\_dummy*: 1.4.  
*sp\_el*: 1.2, 1.4.  
*sp\_generic*: 1.2.  
*sp\_lookup*: 1.2.  
*sp\_m*: 1.2.  
*sp\_multiplicity*: 1.2.  
*sp\_ncdecl*: 1.3.  
*sp\_ncdef*: 1.3.  
*sp\_ncomp*: 1.2, 1.4.  
*sp\_ncomp\_max*: 1.2.  
*sp\_ncwrite*: 1.3.  
*sp\_num*: 1.2, 1.4.  
*sp\_sy*: 1.2.  
*sp\_z*: 1.2.

*species*: 1.4.  
*species\_count*: 1.2.  
*species\_el*: 1.2.  
*species\_generic*: 1.2.  
*species\_infile*: 1.2.  
*species\_m*: 1.2.  
*species\_multiplicity*: 1.2.  
*species\_name*: 1.2.  
*species\_ncomp*: 1.2.  
*species\_sy*: 1.2.  
*species\_version*: 1.2, 1.3.  
*species\_z*: 1.2.  
*speciesfile*: 1.3.  
*st\_decls*: 1.2, 1.3.  
*status*: 1.2.  
*string\_length*: 1.3.  
*tempfile*: [1.2](#), [1.3](#).

*unit*: 1.2.

*var\_realloca*: 1.2.  
*var\_reallocb*: 1.2.

*zero*: 1.2.

⟨Functions and Subroutines 1.2, 1.3, 1.4⟩ Used in section 1.1.

⟨Memory allocation interface 0⟩ Used in section 1.2.

**COMMAND LINE:** "fweave -f -i! -W[ -ykw700 -ytw40000 -j -n/  
/u/dstotler/degas2/src/speciesetup.web".

**WEB FILE:** "/u/dstotler/degas2/src/speciesetup.web".

**CHANGE FILE:** (none).

**GLOBAL LANGUAGE:** FORTRAN.